Unité Mixte de Recherche 5104 CNRS - INPG - UJF
Centre Equation
2, avenue de VIGNATE
F-38610 GIERES
tel : +33 456 52 03 40
fax : +33 456 52 03 50
http://www-verimag.imag.fr

Causal Semantics for the Algebra of Connectors

Simon Bliudze and Joseph Sifakis

Technical Report n^o TR-2008-4

March 13, 2008

Causal Semantics for the Algebra of Connectors

Simon Bliudze and Joseph Sifakis

March 13, 2008

Abstract

The Algebra of Connectors $\mathcal{AC}(P)$ is used to model structured interactions in the BIP component framework. Its terms are *connectors*, 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*.

In a previous paper, we have 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 father 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*. This representation is used for their manipulation and simplification as well as for synthesizing connectors.

How to cite this report:

@techreport {TR-2008-4,
title = {Causal Semantics for the Algebra of Connectors},
authors = {Simon Bliudze and Joseph Sifakis},
institution = {VERIMAG},
number = {TR-2008-4} ,
year = {2008}
}

Contents

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.

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 dispersed 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 [BWH+03, BGK+06, EJL+03]. 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 [BS07]. 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, while pushing to the second plan or completely abstracting the data flow.

Control flow connectors are often specified in an operational setting, usually a process algebra. In [BCD00], 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 [SG03], 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 [Fia04] .

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 [BLM06], an algebra of connectors is developed that allows, in particular, an algebraic translation of the categorical approach used in CommUnity. This algebra allows to construct *stateless* connectors from a number of basic ones.

Reo [Arb04, Arb05] 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 [BLM06], as it focuses on stateless connectors in a control flow setting. We consider connectors as relations between ports with synchronization types, which allows to describe complex coordination patterns with an extremely small set of basic primitives.

In a previous paper [BS07], we have studied an *interaction semantics* for the Algebra of Connectors $\mathcal{AC}(P)$, which is used to model interactions in the BIP component framework [BBS06, Sif05]. 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*.

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*

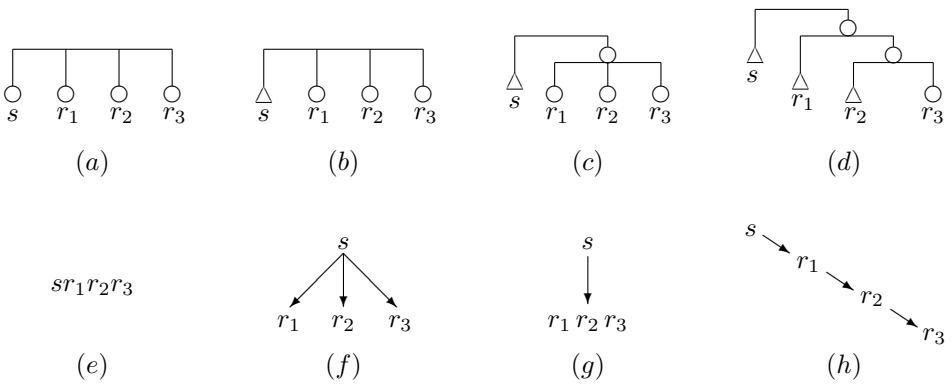

Figure 1: *Connectors and causal trees representing a rendezvous* (a, e)*, a broadcast* (b, f)*, an atomic broadcast* (c, g)*, and a causal chain* (d, h)*.*

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 transmitting port is a trigger, receiving ports are synchrons, and possible interactions are those containing the trigger. Connectors, representing these two protocols for a sender s and receivers r_1, r_2, r_3, are shown in Figure $1(a, b)$. Triangles represent triggers, and circles represent synchrons.

Hierarchical connectors are expressions composed of typed ports and/or typed sub-connectors. Figure $1(c)$ shows a connector realizing an atomic broadcast from a port s to ports r_1, r_2, r_3. The sender port s is a trigger, and the three receiver ports are strongly synchronized in a sub-connector itself typed as a synchron. The corresponding $\mathcal{AC}(P)$ term is $s'[r_1 r_2 r_3]$, 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 Figure $1(d)$ 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 [BS07], 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. Figure $1(e - h)$ 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 $\mathcal{AC}(P) \to \mathcal{CT}(P)$. 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. We also provide a method for synthesizing a set of connectors realizing any boolean constraint on variables from P.

The paper is structured as follows. Section 2 provides a succinct presentation of the basic semantic model for BIP and in particular, its composition parameterized by interactions. Section 3 presents the Algebra of Connectors and its global interaction semantics. Section 4 presents the Algebra of Causal Trees and its properties as well as causal semantics for $\mathcal{AC}(P)$. The last section studies causal rules for representing causal trees and computing their intersection.

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 from BIP programs, C++ code executable on a dedicated platform (see [BBS06, bip]).

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

Definition 2.1. 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 2.2. A transition system is a triple $B = (Q, P, \to)$, where Q is a set of *states*, P is a set of *ports*, and $\to \subseteq Q \times 2^P \times Q$ is a set of *transitions*, each labelled by an interaction.

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

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

In BIP, a system can be obtained as the composition of n components, each modelled by a transition system $B_i = (Q_i, P_i, \to_i)$, 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 $\gamma \subset 2^P$ is the transition system $B = (Q, P, \to_\gamma)$, where $Q = \bigotimes_{i=1}^n Q_i$ and $\to_\gamma$ is the least set of transitions satisfying the rule

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

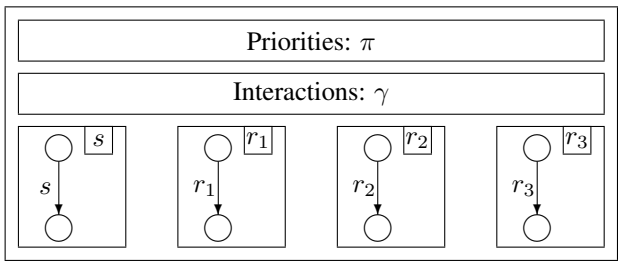

Figure 2: *A system with four atomic components.*

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 [bip, BS07]. Throughout this paper, whenever two interactions, a and a', such that $a \subset a'$, are possible, we always choose a'.

Example 2.3 (Sender/Receivers)**.** Figure 2 shows a component $\pi\,\gamma(S, R_1, R_2, R_3)$ 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 and priorities π. 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 2.4 (Modulo-8 counter)**.** Figure 3 shows a model for the Modulo-8 counter presented in [MR01], 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.

Notice that the composition operator can express usual parallel composition operators [BS07], such as the ones used in CSP [Hoa85] and CCS [Mil89]. By enforcing maximal progress, priorities allow to express broadcast.

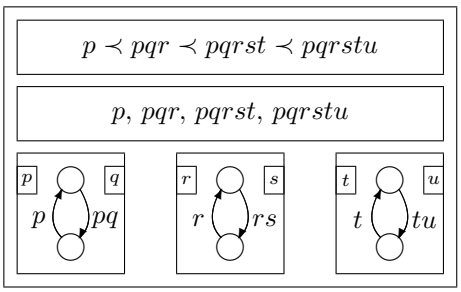

Figure 3: *Modulo-8 counter.*

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 [BBS06]. 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. [BS07]).

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 (Definition 2.1) 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

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

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

$$\begin{array}{lcl} \|0\| = \emptyset, & \|1\| = \{\emptyset\}, & \|p\| = \Big\{\{p\}\Big\}, \\ \|x_1 + x_2\| & = & \|x_1\| \cup \|x_2\|, \\ \|x_1 \cdot x_2\| & = & \Big\{a_1 \cup a_2 \,\Big|\, a_1 \in \|x_1\|,\ a_2 \in \|x_2\|\Big\}, \\ \|(x)\| & = & \|x\|, \end{array} \tag{3}$$

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.

Note 3.1. Interactions, in the sense of Definition 2.1, correspond to singleton subsets $\{a\} \subset 2^P$ with $a \neq \emptyset$. In the following, we lift this restriction to include the singleton subset $\{\emptyset\} \subset 2^P$ represented by $1 \in \mathcal{AI}(P)$. The term $0 \in \mathcal{AI}(P)$ corresponds to an empty subset of 2^P and, consequently, does not represent any interaction. Thus interactions correspond to non-zero monomial terms of $\mathcal{AI}(P)$.

Proposition 3.2 ([BS07])**.** *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 3.3 (Sender/Receiver continued)**.** The second column of Table 1 shows the representation in $\mathcal{AI}(P)$ for the four interaction schemes of Example 2.3.

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	$s\,(1 + r_1\,(1+$ $+ r_2\,(1 + r_3)))$	$s'\,[r_1'\,[r_2'\, r_3]]$	$s \to r_1 \to r_2 \to r_3$

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

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

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:

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

for $p_{i_1}, \dots, p_{i_k} \in P$, 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.

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 $p \in P$, $p = true$ 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. Note 3.1 and Table 2).

Definition 3.4. 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

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

Note 3.5. 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 3.6. *An interaction belongs to the set described by an expression $x \in \mathcal{AI}(P)$ if and only if it satisfies $\beta(x)$:*

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

Note 3.7. As $\|0\| = \emptyset$, according to Definition 3.4, it satisfies all formulae in $\mathbb{B}[P]$, and in particular $0 \models false$. This is the only term in $\mathcal{AI}(P)$ satisfying the constant predicate $false$. Recall (Note 3.1) 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 3.8. 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

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

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

Notation 3.10. 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) $\Big[[x]^\alpha\Big]^\beta = [x]^\beta$.

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

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

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

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

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 3.11. 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 3.12 ([BS07])**.** *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 3.13 (Sender/Receiver continued)**.** The third column of Table 1 shows the connectors for the four interaction schemes of Example 2.3.

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 3.14 (Modulo-8 counter continued)**.** In the model shown in Figure 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 Figure 3.

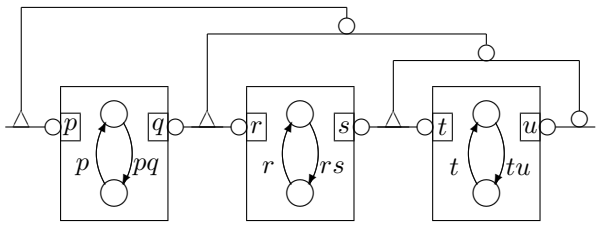

Figure 4: *Modulo-8 counter.*

Definition 3.15. 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 3.16. 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 3.17 ([BS07])**.** *For $x, y \in \mathcal{AC}(P)$, we have*

$$x \cong y \Longleftrightarrow \begin{cases} x \simeq y \\ x \cdot 1' \simeq y \cdot 1' \\ \#x > 0 \Leftrightarrow \#y > 0\,, \end{cases} \tag{12}$$

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 3.18. *For $x, y \in \mathcal{AC}(P)$, holds $[x]'\,[y]' \cong \left[[x]'\,[y]'\right]'$.*

4 Causal semantics for connectors

In this section, we propose a new *causal* semantics for $\mathcal{AC}(P)$ connectors. This allows 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 to define a subset $\mathcal{AC}_c(P) \subset \mathcal{AC}(P)$ 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 [BS07], we have shown 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 the complex enumeration of the interactions of connectors entailed by the method in Section 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. Section 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

$$t ::= a \,|\, (t \to t) \,|\, (t \oplus t)\,, \tag{13}$$

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

$$|a| = a\,, \tag{14}$$
$$|a \to t| = a\Big(1 + |t|\Big)\,, \tag{15}$$
$$|t_1 \oplus t_2| = |t_1| + |t_2| + |t_1|\,|t_2|\,, \tag{16}$$

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 4.1 (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 (15), 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 4.2. *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 $|t_1| = |t_2|$.*

Proof. This proposition is proved by verifying that the semantics of left- and right-hand sides coincide for all axioms above. For most axioms, this trivially follows from the properties of $\mathcal{AI}(P)$, such as, in particular, the idempotence of both union and synchronization. Let us show this, for instance, for axiom (3b).

By axioms (3a) and (3c) and the induced semantic rules, it is sufficient to consider the case $t_1 = a$, for some interaction $a \in 2^P$. We compute the semantics of both sides:

$$|a \to (t_2 \oplus t_3)| \ = \ a\Big(1 + |t_2 \oplus t_3|\Big) \ = \ a\Big(1 + |t_2| + |t_3| + |t_2|\,|t_3|\Big)\,,$$

and

$$\begin{aligned} |a \to t_2 \oplus a \to t_3| &= |a \to t_2| + |a \to t_3| + |a \to t_2|\,|a \to t_3| \ = \\ &= a\Big(1 + |t_2|\Big) + a\Big(1 + |t_3|\Big) + a\Big(1 + |t_2|\Big)\,a\Big(1 + |t_3|\Big) \\ &= a\Big(1 + |t_2| + |t_3| + |t_2|\,|t_3|\Big)\,, \end{aligned}$$

where the last equation follows from the idempotence of operations on $\mathcal{AI}(P)$. □

Note 4.3. According to the axioms of $\mathcal{CT}(P)$ any causal tree can be represented as a parallel composition of its causal chains (see Figure 5). Thus an interaction belonging to a causal tree is a synchronization of any number of prefixes (cf. Example 4.1) of the corresponding causal chains, i.e. branches of this tree.

Example 4.4 (Sender/Receiver continued)**.** The fourth column of Table 1 shows the causal trees for the four interaction schemes of Example 2.3.

Definition 4.5. Two causal trees $t_1, t_2 \in \mathcal{CT}(P)$ are *equivalent*, denoted $t_1 \sim t_2$, iff $|t_1| = |t_2|$.

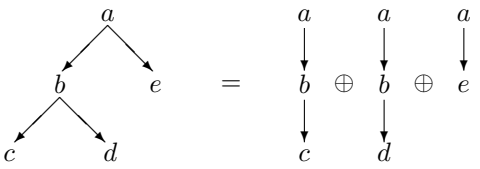

Figure 5: *A causal tree is the parallel composition of its causal chains.*

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

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

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 with a connector a causal tree (the following three equations are sufficient by Corollary 3.18):

$$\tau\left([x_1]'\,[x_2]'\prod_{i=1}^{n} y_i\right) = \tau\left([x_1]'\prod_{i=1}^{n} y_i\right) \oplus \tau\left([x_2]'\prod_{i=1}^{n} y_i\right), \tag{17}$$

$$\tau\left([x]'\prod_{i=1}^{n}[y_i]\right) = \tau(x) \to \bigoplus_{i=1}^{n}\tau(y_i)\,, \tag{18}$$

$$\tau\left(\prod_{i=1}^{n}[y_i]\right) = \bigoplus_{j=1}^{m}\left(a_j \to \bigoplus_{i=1}^{n} rest\Big(\tau(y_i)\Big)\right), \tag{19}$$

where $x, x_1, x_2, y_1, \ldots, y_n \in \mathcal{AC}(P)$, and, in (19), a_j are such that

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

We also define the function $\sigma : \mathcal{CT}(P) \to \mathcal{AC}(P)$, which associates with a causal tree a connector:

$$\sigma(a) = a\,, \tag{20}$$

$$\sigma(a \to t) = [a]'\,[\sigma(t)]\,, \tag{21}$$

$$\sigma(t_1 \oplus t_2) = [\sigma(t_1)]'\,[\sigma(t_2)]'\,. \tag{22}$$

Proposition 4.6. *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* $\forall x \in \mathcal{AC}(P),\ \sigma(\tau(x)) \simeq x$*).*

Sketch of the proof. The first three properties can be demonstrated by comparing definitions (8)–(10) and (14)–(16) of the semantic function $|\cdot|$ and (17)–(22) for functions τ and σ. The fourth property then follows trivially from the first two. □

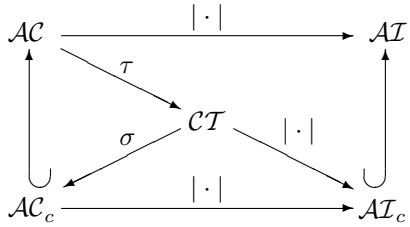

Figure 6: *A diagram relating the algebras.*

The above proposition says that the diagram shown in Figure 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, $\mathcal{AC}_c(P) \subset \mathcal{AC}(P)$ 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, $\mathcal{AI}_c(P) \subset \mathcal{AI}(P)$ 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 4.7. *The set of the causal interactions is closed under synchronization, that is* $x \in \mathcal{AI}_c(P)$ *iff* $\forall a, b \in \|x\|,\ ab \in \|x\|$.

Proof. Consider $x \in \mathcal{AI}_c(P)$ and interactions $a, b \in \|x\|$. There exists $t \in \mathcal{CT}(P)$ such that $x = |t|$. Hence, according to Note 4.3, both a and b can be represented as unions of a number of prefixes of branches of t, which implies automatically that ab can also be represented in this form, and therefore $ab \in \|x\|$.

To prove that the condition of the proposition is sufficient, consider $x \in \mathcal{AI}(P)$ satisfying this property, and take $t = \bigoplus_{a \in \|x\|} a$. Clearly, $|t| = x$, which, by definition, implies $x \in \mathcal{AI}_c(P)$. □

Proposition 4.8. $\forall t_1, t_2 \in \mathcal{CT}(P),\ t_1 \sim t_2 \Rightarrow \sigma(t_1) \cong \sigma(t_2)$.

Proof. This proposition follows directly from Proposition 4.6(4) and Theorem 3.17. □

Corollary 4.9. *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* $x_1 \cong x_2$.

Proof. By definition of $\mathcal{AC}_c(P)$, there exist $t_1, t_2 \in \mathcal{CT}(P)$ such that $x_1 = \sigma(t_1)$ and $x_2 = \sigma(t_2)$. By Proposition 4.6(1,3), $t_1 \sim t_2$, which, by Proposition 4.8, implies $x_1 \cong x_2$. □

4.3 Boolean representation of connectors

Definition 4.10. 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, consequently, are conjunctions of dual Horn clauses, i.e. disjunctions of variables whereof at most one is negative.

In line with Definition 3.4, 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 4.11. 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 $true \Rightarrow a$, it is necessary that it contain a.
2. Rules of the form $p \Rightarrow true$ are satisfied by all interactions.
3. An interaction can satisfy the rule $p \Rightarrow false$ only if it does not contain p.

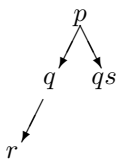

Figure 7: *Graphical representation of the causal tree* $t = p \rightarrow (q \rightarrow r \ \oplus \ qs)$.

Note 4.12. Notice that $a_1 \vee a_1\, a_2 = a_1$, and therefore causal rules can be simplified accordingly:

$$(p \Rightarrow a_1 \vee a_1\, a_2) \ \rightsquigarrow \ (p \Rightarrow a_1)\,. \tag{23}$$

We assume that all the causal rules are simplified by using (23).

Definition 4.13. A *system of causal rules* is a set $R = \{p \Rightarrow x_p\}_{p \in P^t}$, where $P^t \stackrel{def}{=} P \cup \{true\}$. 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

$$R(t) \stackrel{def}{=} \{p \Rightarrow c_p(t)\}_{p \in P^t}\,, \tag{24}$$

where, for $p \in P^t$, the function $c_p : \mathcal{CT}(P) \rightarrow \mathbb{B}[P]$ 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

$$\begin{aligned}
c_p(0) &= false\,, && (25)\\
c_p(p \rightarrow t) &= true\,, && (26)\\
c_p(pa \rightarrow t) &= a\,, && (27)\\
c_p(a \rightarrow t) &= a\, c_p(t)\,, && (28)\\
c_p(t_1 \oplus t_2) &= c_p(t_1) \vee c_p(t_2)\,, && (29)
\end{aligned}$$

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

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

Note 4.14. It is important to observe that, for any $t \in \mathcal{CT}(P)$, the system of causal rules $R(t)$, defined by (24), 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 $p \Rightarrow false$. For ports that do not have any causality constraints, the rule is $p \Rightarrow true$.

Proposition 4.15. *For any causal tree* $t \in \mathcal{CT}(P)$, $|t| = |R(t)|$.

Proof. This proposition follows from Note 4.3 and a similar observation for the system $R(t)$. Indeed, according to the rules (25)–(29) and the simplification rule (23), the cause in a causal rule $p \Rightarrow c_p(t)$ is the union of all the shortest prefixes in t, containing p. □

Example 4.16. Consider the causal tree $t = p \rightarrow (q \rightarrow r \oplus qs)$ shown in Figure 7. The associated system $R(t)$ of causal rules is

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

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

The corresponding boolean formula is then

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

5 Constructing causal trees from boolean functions

5.1 Expressing boolean functions as causal rules

In the previous section, we have introduced the causal rules providing a straightforward way for computing boolean representations of connectors (and causal trees). This section deals with the reverse procedure: given a boolean function on P, we construct a causal tree model (and consequently an $\mathcal{AC}(P)$ connector). We proceed in two steps: given a boolean formula, we translate it into an equivalent one, which is the disjunction of the corresponding causal rules, from which we construct an equivalent causal tree. This translation leads, in particular, to the definition of a normal form for causal trees. We also derive a simple algorithm for computing the intersection of causal trees directly on the corresponding systems of causal rules.

In order to compute the causal rules for a given boolean function $\varphi \in \mathbb{B}[P]$, we take its conjunctive normal form (CNF)

$$\varphi = C_1 \wedge C_2 \wedge \ldots \wedge C_n$$

with, for $k \in [1, n]$,

$$C_k = \bigvee_{i \in I_k} p_i \vee \bigvee_{j \in J_k} \overline{p_j}\,,$$

where $I_k \cap J_k = \emptyset$, and $p_i, p_j \in P$ for all $i \in I_k$ and $j \in J_k$. We can now rewrite every clause C_k, with $J_k \neq \emptyset$, as a disjunction of dual Horn clauses.

$$C_k = \bigvee_{j \in J_k} \left(\overline{p_j} \vee \bigvee_{i \in I_k} p_i \right)\,.$$

By distributivity, we obtain a representation of φ as a disjunction of dual Horn formulae and, after combining the clauses with the same negative variable,

$$\varphi = R_1 \vee R_2 \vee \ldots \vee R_m \tag{30}$$

with, for $k \in [1, m]$,

$$R_k = \bigwedge_{i \in \widetilde{I}_k} \left(\overline{p_i} \vee \bigvee_{j \in \widetilde{J}_{k,i}} a_j \right) = \bigwedge_{i \in \widetilde{I}_k} \left(p_i \Rightarrow \bigvee_{j \in \widetilde{J}_{k,i}} a_j \right)\,,$$

where, for all $i \in \widetilde{I}_k$, $p_i \in P^t$ and, for all $j \in \widetilde{J}_{k,i}$, a_j is $false$, $true$, or a conjunction of positive variables. Recall (Example 4.11) that for a positive clause C_k we have $C_k = (true \Rightarrow C_k)$, whereas $\overline{p} = (p \Rightarrow false)$. Thus, each R_k in (30) is a system of causal rules as defined in Section 4.3.

Proposition 5.1. *For a given boolean function $\varphi \in \mathbb{B}[P]$, the representation (30) is defined uniquely.*

The next section presents an algorithm for constructing a causal tree for a system of causal rules, thus completing the chain of transformations necessary for constructing the causal tree corresponding to a boolean function.

This algorithm also allows to normalize and compute intersections of causal trees by transforming them into systems of causal rules and back. These two operations are also presented in the subsequent sections.

5.2 Constructing causal trees from causal rules

Definition 5.2. A system of causal rules $\{p_i \Rightarrow x_i\}_{i=1}^n$ is *saturated* iff, for all $i \in [1,n]$, $x_i = x_i[x_j/p_j]$, where $x_i[x_j/p_j]$ is obtained by substituting x_j for p_j in x_i, for all $j \neq i$. We denote by $\mathcal{CR}(P)$ the set of saturated systems of causal rules over P.

For a given system of causal rules $R = \{p_i \Rightarrow x_i\}_{i=1}^n$, we denote by $R_{sat} = \{p_i \Rightarrow x_i^*\}_{i=1}^n$, the saturated system of rules, where $\{x_i^*\}_{i=1}^n$ is the unique fixpoint iteratively computed by

$$x_i^0 = x_i, \quad x_i^{k+1} = x_i^k[(p_j x_j^k)/p_j], \quad \text{for } i = 1, \ldots, n\,.$$

Clearly, this computation terminates within a bounded number of iterations.

Lemma 5.3. *Let R be a system of causal rules, and R_{sat} be the corresponding saturated system. Then $|R| = |R_{sat}|$.*

Proof. This lemma follows directly from the observation that the substitution, used to compute the fixpoint in the definition of saturation, preserves boolean equivalence of systems of causal rules. □

Example 5.4. Consider the system of causal rules $\{p \Rightarrow a_p,\ q \Rightarrow pa_q\}$, where $p, q \in P$ are two ports, and $a_p, a_q \in 2^P$. To saturate it, we substitute pa_p for p in the second rule to obtain $\{p \Rightarrow a_p,\ q \Rightarrow pa_pa_q\}$.

Clearly, the corresponding boolean formulae are equivalent:

$$\begin{aligned}(p \Rightarrow a_p) \wedge (q \Rightarrow pa_q) \;&=\; (\overline{p} \vee a_p) \wedge (\overline{q} \vee pa_q) \;= \\ &= \overline{p}\,\overline{q} \vee a_p\overline{q} \vee pa_pa_q \;=\; (\overline{p} \vee a_p) \wedge (\overline{q} \vee pa_pa_q) \;=\; (p \Rightarrow a_p) \wedge (q \Rightarrow pa_pa_q)\,.\end{aligned}$$

Note 5.5. Observe that, for any $t \in \mathcal{CT}(P)$, the system of causal rules $R(t)$, defined by (24), is saturated.

Lemma 5.6. *Let $X = \{p \Rightarrow x_p\}_{p \in P^t}$ be a saturated system of causal rules simplified by absorption (23), with $x_p = \bigvee_{i=1}^{m_p} a_i^p$. The set $Y = \{pa_i^p \,|\, p \in P, i \in [1, m_p]\} \cup \{a_i^{true} \,|\, i \in [1, m_{true}]\}$ consists exactly of all interactions satisfying X and minimal in the following sense:*

1. *Any interaction a, such that $a \models X$, can be decomposed as $a = b_1 \ldots b_k$, with $b_1, \ldots, b_k \in Y$.*
2. *No $a \in Y$ can be further decomposed in this way, i.e. $a = b_1 \ldots b_k$, with $1 \neq b_j \in Y$ for $j \in [1,k]$, implies $k = 1$.*

Proof. 1. Consider $a \models X$, and a port $p_1 \in a$. We then have $a \models \bigwedge_{p \in P^t}(p \Rightarrow \bigvee_{i=1}^{m_p} a_i^p)$, and therefore, for some $i_1 \in [1, m_{p_1}]$, $a_{i_1}^{p_1} \subseteq a$. Hence, $a = b_1 a_1$, with $b_1 = p_1 a_{i_1}^{p_1}$ and $a_1 = a \setminus b_1$. Idempotence of synchronization in $\mathcal{AI}(P)$ allows us to proceed by picking some $p_2 \in a_1$ and applying the same reasoning to obtain $a = b_1 b_2 a_2$, where $b_j = p_j a_{i_j}^{p_j}$, for $j = 1, 2$, and $a_2 = a_1 \setminus b_2$, and so on. As at each step we select $p_j \in a_{j-1}$, we have $a_j \subsetneq a_{j-1}$, and therefore, for some k, $a_k = 1$, which implies $a = b_1 \ldots b_k$, with $b_j = p_j a_{i_j}^{p_j} \in Y$, for $j \in [1,k]$.

2. Consider $a = b_1 \ldots b_k \in Y$. As $a \in Y$, for some $p \in P^t$, we have $a = pa_p$, where a_p is a summand in x_p. If $k > 1$, there exists $l \in [1,k]$ such that $p \in b_l$ and $b_l \subsetneq a$. As $b_l \in Y$, we have $b_l = qa_q$, for some $q \in P^t$ and a_q a summand in x_q. The assumption that all rules are simplified by absorption (23) implies that $p \neq q$. As X is saturated, we have

$$x_q = x_q\Big[(px_p)/p\Big] = \bigvee_{i=1}^{m_q}\bigvee_{j=1}^{m_p} a_i^q[(pa_j^p)/p] = \bigvee_{i \in I}\left(\bigvee_{j=1}^{m_p} a_i^q a_j^p\right) \vee \bigvee_{i \notin I} a_i^q\,,$$

where $I \subseteq [1, m_q]$ is the subset indexing the summands of x_q containing p, i.e. $i \in I$ iff $p \in a_i^q$. As a_q is a summand in x_q and $p \in a_q$, there exist $i \in [1, m_q]$ and $j \in [1, m_p]$, such that $a_q = a_i^q a_j^p$. Hence, $a_j^p \subseteq a_q \subsetneq a_p$ (recall that $q \in a_p$, but $q \notin a_q$). However, both a_j^p and a_p are summands in x_p, which contradicts the assumption that all rules are simplified by absorption (23). □

Theorem 5.7. *Given a saturated system of causal rules $X = \{p \Rightarrow x_p\}_{p \in P^t}$, the algorithm shown in Figure 8, constructs an equivalent causal tree t.*

Input: A saturated system of causal rules $X = \{p \Rightarrow x_p\}_{p \in P^t}$ with $x_p = \bigvee_{i=1}^{m_p} a_i^p$.
Output: A causal tree $t \in \mathcal{CT}(P)$ equivalent to X.

1. $Y := \{pa_i^p \,|\, p \in P, i \in [1, m_p]\} \cup \{a_i^{true} \,|\, i \in [1, m_{true}]\}$;
2. $Z := \min Y$; // the subset of interactions of minimal cardinality in Y
3. $t := \bigoplus Z$;
4. $W := \min(Y \setminus Z)$;
5. for each $w \in W$ and $z \in Z$
6. if $z \subset w$
   // add a son labelled by w to the node z in t
7. replace the subtree $z \to t_z$ of t rooted in z by $z \to (t_z \oplus w)$;
8. $Y := Y \setminus Z$;
9. $Z := W$;
10. if $Y \neq \emptyset$ goto Step 4;
11. clean-up: starting from the leaves of t, for all nodes $a_1 \to a_2$ replace a_2 by $a_2 \setminus a_1$.

Figure 8: *Algorithm for constructing a causal tree from a saturated system of causal rules.*

Proof. Consider the set Y defined in Lemma 5.6. By Lemma 5.6(2), the elements of Y correspond exactly to all the prefixes in the causal tree constructed by the algorithm shown in Figure 8. Thus, denoting by $\|X\|$ the set $\{a \in 2^P \,|\, a \models X\}$, Lemma 5.6(1) implies $\|X\| \subseteq \|t\|$. As X is saturated, $Y \subset \|X\|$. Therefore, by Proposition 4.7, $\|t\| \subseteq \|X\|$, which finalizes the proof. □

5.3 Normal form for causal trees

Lemma 5.8. *Let R_1, R_2 be two saturated systems of causal rules. Then $\|R_1\| = \|R_2\|$ implies $R_1 = R_2$ (equal as sets of causal rules that cannot be simplified by absorption).*

Proof. Suppose that $R_1 \neq R_2$, and all rules are simplified by absorption (23). Then there exists $p \in P$ such that the rules $(p \Rightarrow \bigvee_{i=1}^{n} a_i) \in R_1$ and $(p \Rightarrow \bigvee_{j=1}^{m} b_j) \in R_2$ do not coincide. Without loss of generality, for some $i \in [1, n]$, we have $b_j \not\subseteq a_i$ simultaneously for all $j \in [1, m]$. This implies that the interaction pa_i does not satisfy the rule $(p \Rightarrow \bigvee_{j=1}^{m} b_j) \in R_2$, and therefore $pa_i \notin \|R_2\|$. At the same time $pa_i \in \|R_1\|$, as R_1 is saturated. □

Corollary 5.9. *Let $t_1, t_2 \in \mathcal{CT}(P)$ be two equivalent causal trees. The corresponding systems of causal rules $R(t_1)$ and $R(t_2)$ (cf. (24)) are identical.*

Proof. $t_1 \sim t_2$ implies $\|R(t_1)\| = \|t_1\| = \|t_2\| = \|R(t_2)\|$. Hence, by Lemma 5.8, $R(t_1) = R(t_2)$. □

Definition 5.10. Let $t \in \mathcal{CT}(P)$ be a causal tree. The *normal form* of t is the causal tree obtained by applying the algorithm in Figure 8 to the system $R(t)$.

Proposition 4.15, Theorem 5.7 and Corollary 5.9 guarantee that the definition above is well founded and, indeed, defines a normal form.

Example 5.11. Consider again the causal tree $t = p \to (q \to r \,\oplus\, qs)$ from Example 4.16 (also shown in Figure 9(a)). We have seen that the associated system $R(t)$ of causal rules is

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

The set, defined in Step 1 of the algorithm in Figure 8, is therefore $Y = \{p, pq, pqr, pqs\}$. The sets of minimal cardinality interactions (represented by Z) at subsequent iterations of Steps 2–10 are then respectively $\{p\}$, $\{pq\}$, and $\{pqr, pqs\}$. Thus, the intermediate tree constructed in Steps 2–10 is that shown in Figure 9(b), whereas the final tree is shown in Figure 9(c).

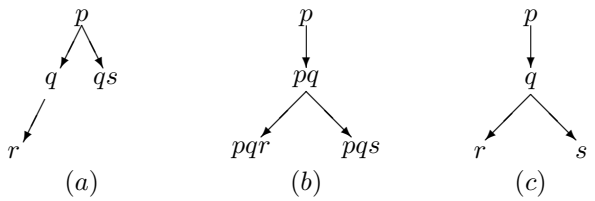

Figure 9: *Construction of the normal form of causality trees example: initial* (a)*, intermediate* (b)*, and normal form* (c) *trees.*

Input:	Causal trees $t_1, t_2 \in \mathcal{CT}(P)$.
Output:	A causal tree $t \in \mathcal{CT}(P)$ such that $\|t\| = \|t_1\| \cap \|t_2\|$.

1. compute systems of causal rules $R(t_1)$ and $R(t_2)$ defined by (24);
2. $X := \{p \Rightarrow c_p(t_1) \wedge c_p(t_2) \,|\, p \in P^t\}$;
3. compute X_{sat} by saturating X;
4. apply the algorithm in Figure 8 to X_{sat};

Figure 10: *Algorithm for computing an intersection of two causal trees.*

6 Examples

For the examples of this section we will need the notions of *interconnected systems* and *multi-shot semantics* introduced in [BS07].

6.1 Multi-shot semantics

Definition 6.1. An *interconnected system* is a pair $(\{B_i\}_{i=1}^n, \{C_j\}_{j=1}^m)$, where $B_i = (Q_i, P_i, \rightarrow_i)$ with $\rightarrow_i \subseteq Q_i \times 2^{P_i} \times Q_i$, are components, and $C_j \in \mathcal{AC}(P)$ with $P = \bigcup_{i=1}^n P_i$.

For an integer parameter $0 < d \leq m$, the *d-shot semantics* of the interconnected system $(\{B_i\}_{i=1}^n, \{C_j\}_{j=1}^m)$ is the system $\gamma_d(B_1, \ldots, B_n)$ defined by applying the rule (1) with $\gamma = \gamma_d$, where $\gamma_d = \sum \prod_{i \in I} [C_i]'$, where the summation is performed over all subsets $I \subseteq [1, m]$ of cardinality d.

Multi-shot semantics corresponds to the case, where d is maximal (i.e. $d = m$), and, in particular, $\gamma_m = \prod_{i=1}^m [C_i]'$.

Notice that, for an interconnected system, d-shot semantics entails simultaneous firing of interactions from at most d connectors.

The application of rule (1) for the d-shot semantics with $d > 1$, requires the nontrivial computation of all the possible interactions. For this the following proposition can be used.

Proposition 6.2. *Let $S = (\{B_i\}_{i=1}^n, \{C_j\}_{j=1}^m)$ be an interconnected system. For $i \in [1, n]$, we denote by $G_i = \sum_{q_i \in Q_i} G_{q_i}$, with $G_{q_i} = \sum_{q_i \xrightarrow{a}} a$, the set of all interactions offered by the component i alone. Let $G = \prod_{i=1}^n [G_i]'$. The set of the possible interactions for d-shot semantics of S is $G \cap \gamma_d$.*

Notice that $G = \prod_{i=1}^n [G_i]'$ is the set of all the interactions offered by the components, whereas γ_d is the set of the interactions allowed by d-shot semantics. Therefore, the intersection of the two sets characterizes all the possible interactions in the system.

To compute efficiently the intersection of two causal connectors we use their boolean representation. It follows trivially from Proposition 4.7 that an intersection of two causal sets of interactions is itself causal. To simplify presentation, we reason on causal trees. The mapping between $\mathcal{AC}(P)$ and $\mathcal{CT}(P)$ is given by functions σ and τ defined in Section 4.2.

Lemma 6.3. *For $t_1, t_2 \in \mathcal{CT}(P)$, the algorithm in Figure 10 computes the unique causal tree in normal form $t \in \mathcal{CT}(P)$ (denoted $t_1 \cap t_2$), such that $\|t\| = \|t_1\| \cap \|t_2\|$.*

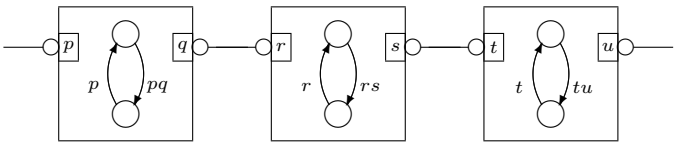

Figure 11: Multi-shot modulo-8 counter

Table 3: *Causal trees and rules for Example 6.5.*

G	γ_m
$(p \rightarrow q) \oplus (r \rightarrow s) \oplus (t \rightarrow u)$	$p \oplus qr \oplus st \oplus u$
$true \Rightarrow p + r + t$	$true \Rightarrow p + qr + st + u$
$p \Rightarrow true \quad r \Rightarrow true \quad t \Rightarrow true$	$p \Rightarrow true$
$q \Rightarrow p \quad s \Rightarrow r \quad u \Rightarrow t$	$q \Rightarrow r \quad r \Rightarrow q \quad s \Rightarrow t \quad t \Rightarrow s$

Example 6.4. Consider two causal trees with opposite causal relations: $t_1 = p \rightarrow q$ (possible interactions: p and pq) and $t_2 = q \rightarrow p$ (possible interactions: q and pq). Let us compute $t_1 \oplus t_2$ and $t_1 \cap t_2$.

The systems of causal rules corresponding to t_1 and t_2 are respectively $\{true \Rightarrow p, p \Rightarrow true, q \Rightarrow p\}$ and $\{true \Rightarrow q, q \Rightarrow true, p \Rightarrow q\}$.

1. To compute $t_1 \oplus t_2$, we apply (29) from the definition of the function c_p to obtain the system of causal rules $\{true \Rightarrow p \vee q,\ p \Rightarrow true \vee q,\ q \Rightarrow true \vee p\}$, which is simplified to $\{true \Rightarrow p \vee q,\ p \Rightarrow true,\ q \Rightarrow true\}$ by absorption (23). This corresponds to the causal tree $p \oplus q$.

2. To compute $t_1 \cap t_2$, we apply Lemma 6.3 to obtain the system $\{true \Rightarrow pq, p \Rightarrow q, q \Rightarrow p\}$, which saturates to $\{true \Rightarrow pq, p \Rightarrow pq, q \Rightarrow pq\}$, and produces therefore the causal tree with a single node pq.

Example 6.5 (Modulo-8 counter continued)**.** Consider the interconnected system in Figure 11. The set of the possible interactions for multi-shot semantics is the intersection of $G = [p'q]'\,[r's]'\,[t'u]'$ and $\gamma_m = p'\,[qr]'\,[st]'\,u'$, and corresponds to the modulo-8 counter from Example 2.4. As in Proposition 6.2, G represents the interactions offered by the components, whereas γ_m represents those offered by atomic connectors. The causal trees for G and γ_m are shown in Table 3, as well as the corresponding systems of causal rules.

Applying Lemma 6.3 and absorption (23), we compute the system of causal rules for $G \cap \gamma_m$:

$$\{\, true \Rightarrow p + qr + st + ru + tu\,, \quad p \Rightarrow true\,, \quad q \Rightarrow pr\,, \quad r \Rightarrow q\,, \quad s \Rightarrow rt\,, \quad t \Rightarrow s\,, \quad u \Rightarrow t\,\}\,.$$

After saturation, we obtain

$$\{\, true \Rightarrow p\,, \quad p \Rightarrow true\,, \quad q \Rightarrow pqr\,, \quad r \Rightarrow pqr\,, \quad s \Rightarrow pqrts\,, \quad t \Rightarrow pqrts\,, \quad u \Rightarrow pqrts\,\}\,.$$

By applying the algorithm of Figure 8, we obtain the causal tree $p \rightarrow qr \rightarrow st \rightarrow u$, which represents the connector of Example 3.14.

6.2 Two tasks with preemption

Let T_1 and T_2 be two tasks running on a single processor. We assume that each one of these tasks can preempt the other. No interactions other than preemption are possible.

Tasks can be modelled by the generic atomic component shown in Figure 12(a). Its behavior has three states: 1 – the task is running, 2 – the task is waiting to begin computation, and 3 – the task has been preempted and is waiting to resume computation. The transitions are labelled b, f, p, and r for *begin*, *finish*, *preempt*, and *resume* respectively, and can be synchronized with external events through the corresponding ports of the behavior.

Mutual preemption is described by two statements:

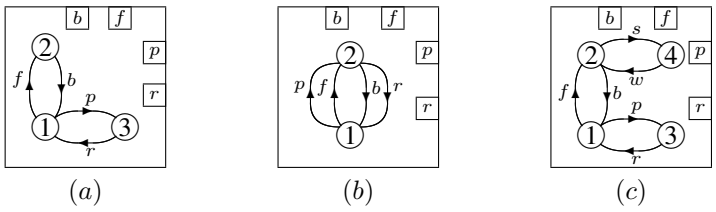

Figure 12: *Three behaviors modelling a preemptable task.*

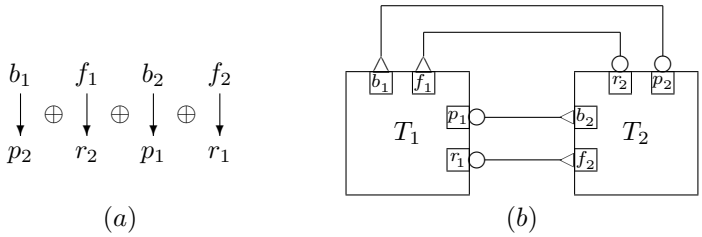

Figure 13: *A causal tree* (a) *and an interconnected system* (b) *modelling two mutually preempting tasks.*

1. A running task is preempted, when the other one begins computation.
2. A preempted task resumes computation, when the other one finishes.

In order to compute the connectors ensuring these interactions, we rewrite these statements as causal rules on $\{b_i, f_i, p_i, r_i\}_{i=1,2}$:

$$\begin{aligned} &true \Rightarrow b_1 \vee f_1 \vee b_2 \vee f_2, \\ &p_1 \Rightarrow b_2, \qquad p_2 \Rightarrow b_1, \\ &r_1 \Rightarrow f_2, \qquad r_2 \Rightarrow f_1\,. \end{aligned} \tag{31}$$

The first rule in (31) means that at any moment at least one task must execute a begin or finish action. It can be easily verified that this system of causal rules is saturated. Applying the algorithm in Figure 8, we obtain the causal tree shown in Figure 13(a), and, furthermore,

$$\sigma\Big(b_1 \to p_2 \;\oplus\; f_1 \to r_2 \;\oplus\; b_2 \to p_1 \;\oplus\; f_2 \to r_1\Big) = \Big[b_1' p_2\Big]' \Big[f_1' r_2\Big]' \Big[b_2' p_1\Big]' \Big[f_2' r_1\Big]'\,. \tag{32}$$

Figure 13(b) shows an interconnected system consisting of two tasks and four connectors forming the right-hand side of (32). The multi-shot semantics of this system realizes the desired interaction model.

Different behaviors can be used to model a preemptable task. In addition to behavior in Figure 12(a), other possible behaviors for tasks are given in Figure 12(b, c).

1. The behavior in Figure 12(b) has two states: 1 – the task is running, 2 – the task is waiting, with the four transitions labelled by b, f, p, and r.
2. The behavior in Figure 12(c) has four states. States 1–3 are the same as those of the behavior in Figure 12(a), whereas in state 4 the task is sleeping, and the two additional transitions s and w correspond respectively to actions *sleep* and *wake-up*.

Independently of which behavior in Figure 12 is used to model the tasks, the interactions offered by each task are $G_i = b_i + f_i + p_i + r_i$, for $i = 1, 2$ (cf. Proposition 6.2). Thus, the interactions possible in the multi-shot semantics of the system in Figure 13(b) are described by

$$\Big[b_1 + f_1 + p_1 + r_1\Big]' \Big[b_2 + f_2 + p_2 + r_2\Big]' \cap \Big[b_1' p_2\Big]' \Big[f_1' r_2\Big]' \Big[b_2' p_1\Big]' \Big[f_2' r_1\Big]' \simeq\; b_1' p_2 + f_1' r_2 + b_2' p_1 + f_2' r_1\,.$$

Hence, the multi-shot and single-shot semantics of this system coincide.

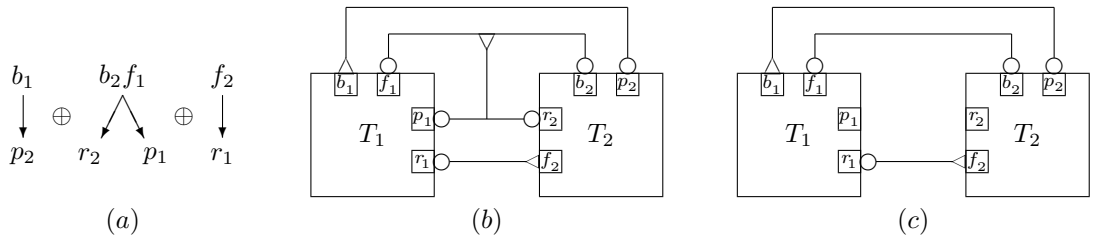

Figure 14: *A causal tree* (a) *and two interconnected systems* (b, c) *modelling a sequential execution of two mutually preempting tasks.*

Notice, however, that the connectors computed above define the set of allowed interactions. The actual interactions, as well as the order of their execution, depend on the behavior that is used to model the tasks. For example, when the behavior in Figure 12(b) is used, the following trace is acceptable in the composed system

$$b_1(b_2p_1)(b_1p_2)(b_2p_1)\dots,$$

whereas, when the behavior in Figure 12(a) is used, an interaction f_2r_1 must follow b_2p_1:

$$b_1(b_2p_1)(f_2r_1)b_1(b_2p_1)\dots.$$

6.3 Sequential execution of two tasks

In a setting, similar to that of the previous section — that is two tasks T_1 and T_2 running on a single processor and mutually preempting each other —, we now additionally require that each execution of T_1 be followed by one of T_2, and, conversely, each execution of T_2 be preceded by one of T_1. In other words, we impose the sequential execution $T_1; T_2$.

The corresponding causal rules can therefore be obtained by adding to (31) the boolean constraint $b_2 = f_1$, expressed as the conjunction of two causal rules $f_1 \Rightarrow b_2$ and $b_2 \Rightarrow f_1$:

$$\begin{aligned} &true \Rightarrow b_1 \vee f_1 \vee b_2 \vee f_2,\\ &p_1 \Rightarrow b_2, \qquad p_2 \Rightarrow b_1,\\ &r_1 \Rightarrow f_2, \qquad r_2 \Rightarrow f_1,\\ &f_1 \Rightarrow b_2, \qquad b_2 \Rightarrow f_1\,, \end{aligned}$$

which saturates to

$$\begin{aligned} &true \Rightarrow b_1 \vee b_2f_1 \vee f_2,\\ &p_1 \Rightarrow b_2f_1, \qquad p_2 \Rightarrow b_1,\\ &r_1 \Rightarrow f_2, \qquad r_2 \Rightarrow b_2f_1,\\ &f_1 \Rightarrow b_2, \qquad b_2 \Rightarrow f_1\,. \end{aligned} \tag{33}$$

The causal tree computed by the algorithm in Figure 8 is shown in Figure 14(a), and the corresponding $\mathcal{AC}(P)$ connector is

$$\sigma\Big(b_1 \to p_2 \;\oplus\; b_2f_1 \to (r_2 \oplus p_1) \oplus\; f_2 \to r_1\Big) = \Big[b_1'p_2\Big]'\Big[[b_2f_1]'r_2p_1\Big]'\Big[f_2'r_1\Big]'\,,$$

which corresponds to the multi-shot semantics of the interconnected system (with tree connectors) in Figure 14(b). (Notice that the ports b_2 and r_2 are interchanged in this figure, as compared to Figure 13(b).)

The above systems of causal rules represent boolean constraints on the interactions of the composed system, derived from the required interaction model (i.e. sequential execution with preemption). Other boolean constraints can also be considered. When we model the two tasks as atomic components given in Figure 12, we can derive additional constraints from their behavior: two ports from the same component cannot participate together in the same interaction. Hence one can, for instance, add to the system (33) the

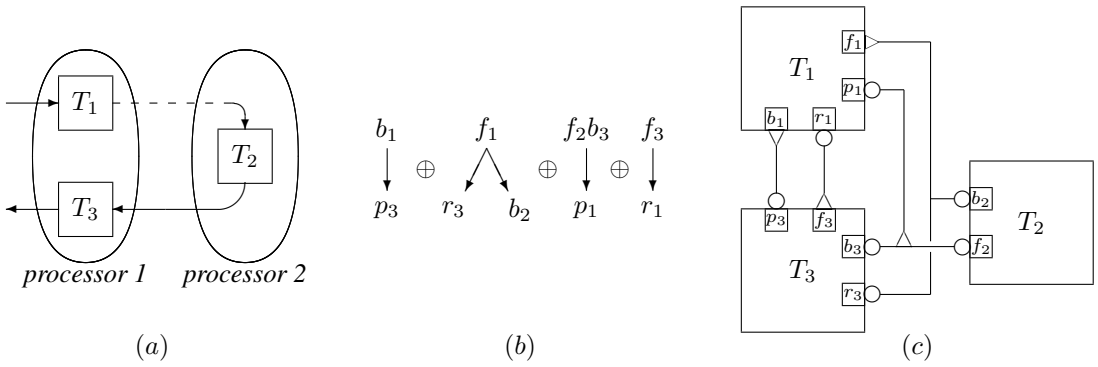

Figure 15: *An illustration* (a)*, a causal tree* (b)*, and an interconnected system* (c) *for the example of three sequential tasks.*

boolean constraints $p_1 \Rightarrow \overline{b_1}\,\overline{f_1}\,\overline{r_1}$ and $r_2 \Rightarrow \overline{b_2}\,\overline{f_2}\,\overline{p_2}$, which modifies the existing causal rules for p_1 and r_2, giving $p_1 \Rightarrow false$ and $r_2 \Rightarrow false$. Thus, the resulting causality tree is $b_1 \rightarrow p_2 \,\oplus\, b_2 f_1 \,\oplus\, f_2 \rightarrow r_1$, and the corresponding $\mathcal{AC}(P)$ connector is (see Figure 14(c))

$$\sigma\Big(b_1 \rightarrow p_2 \,\oplus\, b_2 f_1 \,\oplus\, f_2 \rightarrow r_1\Big) = \Big[b_1' p_2\Big]' \Big[b_2 f_1\Big]' \Big[f_2' r_1\Big]' .$$

6.4 Three sequential tasks running on two processors

We now further develop the example of the previous sections, by introducing one more task, although running on a different processor. Thus our system is composed of three tasks T_1, T_2, and T_3, with T_1 and T_3 running on the same processor and preempting each other as in Section 6.2 (cf. Figure 15(a)). The second task is running on a separate processor and, consequently, cannot be preempted. Therefore ports p and r are irrelevant for T_2, and will not be considered in the sequel.

We want to ensure the following execution protocol:

1. Each execution of T_2 must be immediately preceded by an execution of T_1. However, T_1 can be executed without being immediately followed by an execution of T_2 (dashed arrow in Figure 15(a)). (One can assume, for example, that T_1 represents a producer serving multiple consumers, whereof only one, represented by T_2, has to be considered. Thus several executions of T_1 can happen before an execution of T_2 is triggered.)

2. Each execution of T_2 must be immediately followed by an execution of T_3, and, conversely, each execution of T_3 must be immediately preceded by an execution of T_2 (solid arrow in Figure 15(a)).

As in the previous sections, we represent the constraints characterizing this protocol as causal rules on ports of the three components:

$$\begin{aligned}
&true \Rightarrow b_1 \vee f_1 \vee b_2 \vee f_2 \vee b_3 \vee f_3,\\
&p_1 \Rightarrow b_3, \qquad p_3 \Rightarrow b_1,\\
&r_1 \Rightarrow f_3, \qquad r_3 \Rightarrow f_1,\\
&b_2 \Rightarrow f_1,\\
&f_2 \Rightarrow b_3, \qquad b_3 \Rightarrow f_2\,.
\end{aligned}$$

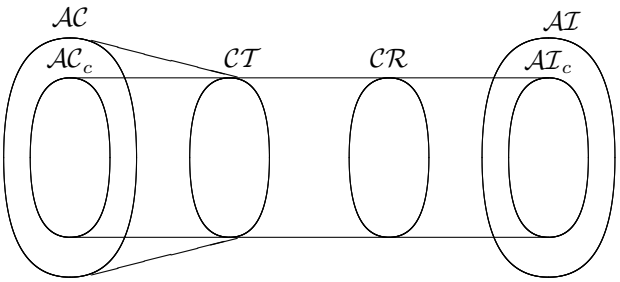

Figure 16: *A graphical representation of the relations between different algebras.*

By saturating this system, we obtain

$$
\begin{aligned}
&true \Rightarrow b_1 \vee f_1 \vee f_2 b_3 \vee f_3,\\
&p_1 \Rightarrow f_2 b_3, \qquad p_3 \Rightarrow b_1,\\
&r_1 \Rightarrow f_3, \qquad r_3 \Rightarrow f_1,\\
&b_2 \Rightarrow f_1,\\
&f_2 \Rightarrow b_3, \qquad b_3 \Rightarrow f_2\,.
\end{aligned}
$$

The corresponding causal tree is shown in Figure 15(b) and the interconnected system in Figure 15(c).

7 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 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 Figure 16.

$\mathcal{CT}(P)$ breaks with the reductionist view of interaction semantics as it distinguishes between symmetric and asymmetric interaction. It allows structuring 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. [Now06]. This opens new possibilities for unifying asynchronous and synchronous semantics.

$\mathcal{CT}(P)$ 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. The examples provided in the previous section show that causal rules can be used for the specification of interactions from which connectors can be synthesized by using the algorithm given in Figure 8. Synthesis of connectors is similar to synthesis of circuits from boolean specifications. This idea seems very interesting and deserves further investigation.

References

[Arb04] Farhad Arbab. Reo: a channel-based coordination model for component composition. *Mathematical Structures in Computer Science*, 14(3):329–366, 2004. 1

[Arb05] Farhad Arbab. Abstract behavior types: a foundation model for components and their composition. *Sci. Comput. Program.*, 55(1-3):3–52, 2005. 1

[BBS06] Ananda Basu, Marius Bozga, and Joseph Sifakis. Modeling heterogeneous real-time components in BIP. In 4^{th} *IEEE International Conference on Software Engineering and Formal Methods (SEFM06)*, pages 3–12, September 2006. Invited talk. 1, 2, 3

[BCD00] Marco Bernardo, Paolo Ciancarini, and Lorenzo Donatiello. On the formalization of architectural types with process algebras. In *SIGSOFT FSE*, pages 140–148, 2000. 1

[BGK+06] K. Balasubramanian, A.S. Gokhale, G. Karsai, J. Sztipanovits, and S. Neema. Developing applications using model-driven design environments. *IEEE Computer*, 39(2):33–40, 2006. 1

[bip] BIP. `http://www-verimag.imag.fr/ async/BIP/bip.html` 2, 2

[BLM06] Roberto Bruni, Ivan Lanese, and Ugo Montanari. A basic algebra of stateless connectors. *Theor. Comput. Sci.*, 366(1):98–120, 2006. 1

[BS07] Simon Bliudze and Joseph Sifakis. The algebra of connectors — Structuring interaction in BIP. In *Proceeding of the EMSOFT'07*, pages 11–20, Salzburg, Austria, October 2007. ACM SigBED. 1, 1, 2, 2, 3, 3.2, 3.12, 3.17, 2, 6

[BWH+03] F. Balarin, Y. Watanabe, H. Hsieh, L. Lavagno, C. Passerone, and A.L. Sangiovanni-Vincentelli. Metropolis: An integrated electronic system design environment. *IEEE Computer*, 36(4):45–52, 2003. 1

[EJL+03] J. Eker, J.W. Janneck, E.A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong. Taming heterogeneity: The Ptolemy approach. *Proceedings of the IEEE*, 91(1):127–144, 2003. 1

[Fia04] José Luis Fiadeiro. *Categories for Software Engineering*. Springer-Verlag, April 2004. 1

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall International Series in Computer Science. Prentice Hall, April 1985. 2

[Mil89] Robin Milner. *Communication and Concurrency*. Prentice Hall International Series in Computer Science. Prentice Hall, 1989. 2

[MR01] Florence Maraninchi and Yann Rémond. Argos: an automaton-based synchronous language. *Computer Languages*, 27:61–92, 2001. 2.4

[Now06] David Nowak. Synchronous structures. *Inf. Comput.*, 204(8):1295–1324, 2006. 7

[SG03] Bridget Spitznagel and David Garlan. A compositional formalization of connector wrappers. In *ICSE*, pages 374–384. IEEE Computer Society, 2003. 1

[Sif05] Joseph Sifakis. A framework for component-based construction. In 3^{rd} *IEEE International Conference on Software Engineering and Formal Methods (SEFM05)*, pages 293–300, September 2005. Keynote talk. 1