Complete Axioms for Stateless Connectors*

Roberto Bruni, Ivan Lanese, and Ugo Montanari

Computer Science Department, University of Pisa, Italy.
{bruni,lanese,ugo}@di.unipi.it

Abstract. The conceptual separation between computation and coordination in distributed computing systems motivates the use of peculiar entities commonly called *connectors*, whose task is managing the interaction among distributed components. Different kinds of connectors exist in the literature, at different levels of abstraction. We focus on a basic algebra of connectors which is expressive enough to model, e.g., all the architectural connectors of CommUnity. We first define the operational, observational and denotational semantics of connectors, then we show that the observational and denotational semantics coincide and finally we give a complete normal-form axiomatization.

Introduction

The advent of modern communication technologies shifted the focus of computer science researchers from isolated computing systems to distributed communicating systems, in which *interaction* plays the prominent role. In Milner's words [21], "computing has grown into informatics and Turing's logical computing machines are matched by a logic of interaction". In this perspective, the analysis of global computing systems is facilitated by approaches, techniques and paradigms that exploit a clean conceptual separation between *computation* and *coordination*. This is much evident at several levels of abstraction (architecture, software, processes), where issues like reusability, maintenance, heterogeneity call for modular specifications, theories and models.

When separating coordination from computation, the notion of a *connector* emerges in different contexts, with slightly different meaning, expressiveness and functionalities. The common trait is the role of a connector: a component that mediates the interaction of other computational components and connectors. In particular, connectors have been studied within both *algebraic* and *categorical approaches* to system modeling.

The algebraic approach [14,20] models systems a elements of a suitable term algebra, with constants modeling basic components that can be composed via the other operators, e.g., parallel composition and name restriction. Operational and abstract semantics are usually based on a labelled transition system defined by structural induction.

The categorical approach [13] models systems as objects in a category, with morphisms defining relations such as subsystem or refinement. Complex software architectures can be modeled as diagrams in the category, with universal constructions, such as colimit, building an object in the same category that behaves as the whole system

* Research supported by the FET-GC Project IST-2001-32747 AGILE and by the project HPRN-CT-2002-00275 SEGRAVIS.

and that is uniquely determined up to isomorphism. The use of architectural connectors within the categorical approach is well exemplified by CommUnity [11,10].

Having rigorous mathematical foundations is crucial for the analysis of coordinated distributed systems. Several different kinds of connectors have been studied in the literature, studying e.g. the observational semantics of process contexts [17,24,12,16], or analysing suitable equational theories and reduction to normal forms [25,6,3,15].

In this paper, we concentrate on the algebraic approach by promoting a small algebra of connectors, for which we define suitable operational, observational and denotational semantics. The first one is expressed using the *Tile Model* [12]. The observational semantics we select is *tile bisimilarity*, that also coincides with *tile trace equivalence* for the algebra under inspection. The denotational semantics is original to this contribution and it is based on (an algebra of) suitable boolean matrices, called tick-tables. We first show that the observational and denotational semantics coincide and then give a complete normal-form axiomatization for them, which is the main result of the paper.

Our connectors are rather simple: they essentially model basic synchronization, mutual exclusion and hiding and they are all stateless. Nevertheless, we think that the analysis of these connectors is quite interesting, since they allow to build a wide range of coordination connectors. For instance, they are expressive enough to model the multiple-action synchronization mechanism of CommUnity which uses morphisms and complex architectural connectors. This is shown in the previous work [2], where an encoding from CommUnity into the Tile Model is defined. One of the main results of [2] is that the translation of a diagram is tile bisimilar to the translation of its colimit.

The above mentioned main result of this paper, namely the complete axiomatization of abstract semantics, improves the work in [2] by showing that, for the part of action coordination, tile bisimilarity can be axiomatized as a suitable equational theory, where equivalence classes have standard representatives. While in the algebraic approach equivalence classes are usually abstract entities, having a normal form gives a concrete representation that matches a nice feature of the categorical approach, namely that the colimit of a diagram is its best concrete representative.

The research initiated in [2] and extended in this paper is a first step towards a more general reconciliation between the categorical and the algebraic approach, of which CommUnity and the Tile Model are just intended to be two selected representatives.

With respect to other approaches to synchronization connectors existing in the literature [3,5,25,6,15], our main contribution is the introduction of the mutual exclusion connector, which allows to specify a wider range of possible synchronization policies. Furthermore, the semantics based on matrices is new and it provides a clean mathematical definition of connectors. Finally, we characterize the classes of matrices that can be specified, both with and without mutual exclusion.

Structure of the paper. § 1 contains some background on symmetric monoidal categories and on the Tile Model. § 2 presents syntax and semantics of connectors, showing the correspondence between the observational and denotational semantics. § 3 contains the main results of the paper, namely the axiomatization of connectors and the theorems for semantic equivalence and normal form: we consider the case without mutual exclusion first (§ 3.1) and the general case later (§ 3.2). Conclusion and future work are in § 4. Due to space constraints, proofs are just sketched: full proofs can be found in [4].

1 Background

Symmetric monoidal categories for connectors. It has been shown in the literature that distributed systems can be conveniently modeled as graphs [8,22,9] that straightforwardly account for the network distribution of processes, mobile agents, etc. The advantage representing configuration graphs as (freely generated) symmetric monoidal categories is three-fold. First, they introduce a suitable notion of (observable) interfaces for configurations. Second, they introduce two key operations for composing graphs, namely sequential and parallel compositions. Third, the natural isomorphism defined by symmetries allows to take graphs up to interface-preserving graph isomorphism.

We recall that a *(strict) monoidal category* [18] $(\mathcal{C},\otimes,e)$ is a category $\mathcal{C}$ together with a functor $\otimes\colon \mathcal{C}\times\mathcal{C}\to\mathcal{C}$ called the *tensor product* and an object e called the *unit*, such that for any arrows $\alpha_1,\alpha_2,\alpha_3\in\mathcal{C}$ we have $(\alpha_1\otimes\alpha_2)\otimes\alpha_3=\alpha_1\otimes(\alpha_2\otimes\alpha_3)$ and $\alpha_1\otimes id_e=\alpha_1=id_e\otimes\alpha_1$. Note that, by functoriality of $\otimes$ we have, e.g., $\alpha_1\otimes\alpha_2=\alpha_1\otimes id_{a_2};id_{b_1}\otimes\alpha_2=id_{a_1}\otimes\alpha_2;\alpha_1\otimes id_{b_2}$ for any $\alpha_i\colon a_i\to b_i, i\in\{1,2\}$.

Definition 1.1. *A* symmetric (strict) monoidal category $(\mathcal{C},\otimes,e,\gamma)$ *is a (strict) monoidal category $(\mathcal{C},\otimes,e)$ together with a family of arrows, called* symmetries, $\{\gamma_{a,b}\colon a\otimes b\to b\otimes a\}_{a,b}$ *indexed by pairs of objects in $\mathcal{C}$ such that for any two arrows $\alpha_1,\alpha_2\in\mathcal{C}$ with $\alpha_i\colon a_i\to b_i$, we have $\alpha_1\otimes\alpha_2;\gamma_{b_1,b_2}=\gamma_{a_1,a_2};\alpha_2\otimes\alpha_1$ (that is γ is a natural isomorphism) that satisfies the coherence equalities (for any objects a,b,c):*

$$\gamma_{a,b};\gamma_{b,a}=id_{a\otimes b} \qquad\qquad \gamma_{a\otimes b,c}=id_a\otimes\gamma_{b,c};\gamma_{a,c}\otimes id_b.$$

The categories we are interested in are those freely generated from an unsorted (hyper)signature Σ, i.e., from a ranked family of operators $\sigma\colon n\to m$. The objects are just natural numbers expressing the arities of the interfaces, i.e., the number of "attach points", with $n\otimes m=n+m$ and $e=0$. The operators $\sigma\in\Sigma$ are seen as basic arrows with source and target defined accordingly to the arity of σ. Symmetries can be always expressed in terms of the basic symmetry $\gamma_{1,1}\colon 2\to 2$. Intuitively, symmetries can be used to rearrange the input-output interfaces of graph-like configurations. We call *permutation* any composition of identities and symmetries. A generic arrow can always be expressed as a suitable composition of id_1, $\gamma_{1,1}$ and $\sigma\in\Sigma$.

Lemma 1.2. *Any arrow α can be decomposed as $id_{n_1}\otimes\sigma_1\otimes id_{m_1};\ldots;id_{n_k}\otimes\sigma_k\otimes id_{m_k}$ for some natural numbers $k,n_1,\ldots,n_k,m_1,\ldots,m_k$ and $\sigma_1,\ldots,\sigma_k\in\{\gamma_{1,1}\}\cup\Sigma$.*

An arrow expressed using only identities and (possibly multiple instances of) one particular $\sigma\in\{\gamma_{1,1}\}\cup\Sigma$ is called a *layer of* σ. For example, a permutation is a layer of $\gamma_{1,1}$.

Tile Model. In this paper, we choose the Tile Model for defining the operational and observational semantics of connectors. In fact, tile configurations are suitable to represent connectors, which include input and output interfaces where actions can be observed and that can be used to compose configurations and to coordinate their behaviours.

The Tile Model [12] is a rule-based framework whose main ingredients are rewrite rules with side effects, called *basic tiles* that combine inspirations from SOS rules [23], *context systems* [17], *structured transition systems* [7] and *rewriting logic* [19].

$$\begin{tikzcd} \textit{initial input interface} & x \arrow[r, "s"] \arrow[d, "a"'] \arrow[dr, phantom, "A"] & y \arrow[d, "b"] & \textit{initial output interface} \\ \textit{final input interface} & z \arrow[r, "t"'] & w & \textit{final output interface} \end{tikzcd}$$

Fig. 1. Graphical representation of a tile A.

A tile $A : s \xrightarrow[b]{a} t$ is a rewrite rule stating that the *initial configuration* s can evolve to the *final configuration* t via A, producing the *effect* b; but the step is allowed only if the 'arguments' of s can produce a, which acts as the *trigger* of A (see Figure 1). Triggers and effects are called *observations* and tile vertices are called *interfaces*.

The operational semantics of concurrent systems can be expressed via tiles if system configurations form a monoidal category $\mathcal{H}$, and observations form a monoidal category $\mathcal{V}$ with the same set of objects as $\mathcal{H}$. Abusing the notation, we denote by $\_ \otimes \_$ both monoidal functors of $\mathcal{H}$ and $\mathcal{V}$ and by $\_;\_$ both sequential compositions in $\mathcal{H}$ and $\mathcal{V}$.

Definition 1.3. *A* tile system *is a tuple $\mathcal{R} = (\mathcal{H}, \mathcal{V}, N, R)$ where $\mathcal{H}$ and $\mathcal{V}$ are monoidal categories with the same set of objects $O_{\mathcal{H}} = O_{\mathcal{V}}$, N is the set of rule names and $R\colon N \to \mathcal{H} \times \mathcal{V} \times \mathcal{V} \times \mathcal{H}$ is a function such that for all $A \in N$, if $R(A) = \langle s, a, b, t \rangle$, then the arrows s, a, b, t can form a tile like in Figure 1.*

Tiles can be composed horizontally, in parallel, and vertically to generate larger steps. Horizontal composition $A;B$ coordinates the evolution of the initial configuration of A with that of B, yielding the 'synchronization' of the two rewrites. Horizontal composition is possible only if the initial configurations of A and B interact cooperatively: the effect of A must provide the trigger for B. The parallel composition $A \otimes B$ builds concurrent steps. Vertical composition $A * B$ is sequential composition of computations.

Tiles can be seen as sequents of *tile logic*: the sequent $s \xrightarrow[b]{a} t$ is *entailed* by the tile logic associated with $\mathcal{R}$, written $\mathcal{R} \vdash s \xrightarrow[b]{a} t$, if it can be obtained by composing some basic tiles in R (possibly using also auxiliary tiles, like identities $id \xrightarrow[a]{a} id$ propagating observations). The "borders" of composed sequents are defined in Figure 2.

The main feature of tiles is their double labeling with triggers and effects, that allows to observe the input-output behaviour of configurations. By taking $\langle \text{trigger}, \text{effect} \rangle$ pairs as labels one can see tiles as a labeled transition system. In this context, the usual notion of bisimilarity is called *tile bisimilarity*.

Definition 1.4. *Let $\mathcal{R} = (\mathcal{H}, \mathcal{V}, N, R)$ be a tile system. A symmetric relation $\sim_{\mathrm{t}}$ on configurations is called* tile bisimulation *if whenever $s \sim_{\mathrm{t}} t$ and $\mathcal{R} \vdash s \xrightarrow[b]{a} s'$, then t' exists such that $\mathcal{R} \vdash t \xrightarrow[b]{a} t'$ and $s' \sim_{\mathrm{t}} t'$.*

The maximal tile bisimulation is called *tile bisimilarity* and it is denoted by $\simeq_{\mathrm{t}}$. Note that $s \simeq_{\mathrm{t}} t$ only if s and t have the same input-output interfaces.

A syntactic property on tiles guaranteeing that $\simeq_{\mathrm{t}}$ is a congruence, i.e. that the semantics is compositional, is the so-called *basic source format* [12], which amounts to

$$\frac{s \xrightarrow[b]{a} t \quad h \xrightarrow[c]{b} f}{s;h \xrightarrow[c]{a} t;f}\ (\mathtt{hor}) \qquad \frac{s \xrightarrow[b]{a} t \quad h \xrightarrow[d]{c} f}{s \otimes h \xrightarrow[b\otimes d]{a\otimes c} t \otimes f}\ (\mathtt{par}) \qquad \frac{s \xrightarrow[b]{a} t \quad t \xrightarrow[d]{c} h}{s \xrightarrow[b;d]{a;c} h}\ (\mathtt{ver})$$

Fig. 2. Inference rules for tile logic.

require that $\mathcal{H}$ is generated from a (hyper)signature Σ and that the initial configuration of each basic tile consists of a basic operator in Σ.

We shall focus on tile systems of *stateless* connectors, meaning that in all basic tiles the final configuration is equal to the initial one. Operatively, this means that the behaviour of a connector is history independent. An easy consequence is that $\simeq_{\mathrm{t}}$ coincides with tile trace equivalence.

2 Algebra of connectors

We present here a rich algebra of connectors for action coordination. We have developed such an algebra to model systems where multiple actions can be executed at each time, either independently or synchronized. Connectors are used to guarantee the global consistency of local evolutions. For instance, in the translation of CommUnity [2], connectors are used in conjunction with other operators representing the computational entities. Roughly these have n attach points associated with actions and according to the computed action they emit 1 tick (action performed) and $n-1$ $\mathsf{unticks}$ (forced inactivity).

We remark that all structures that we are going to present are based on the symmetric strict monoidal structure given by symmetries γ, tensor product $\otimes$ and unit 0.

The complete list of connectors is in Figure 3. The ordinary basic connectors are in the leftmost part of the table, while their duals are on the right (symmetry is self-dual). The term mex stands for "mutual exclusion". We also speak about synch connectors (∇ and Δ), choice connectors ($\underline{\nabla}$ and $\underline{\Delta}$), hiding connectors ($!$ and $¡$) and inaction connectors ($\mathbf{0}$ and $\overline{\mathbf{0}}$). This set of connectors has been used in [2] to model action coordination in CommUnity, an architectural design language which has the extreme separation between computation and coordination as distinctive feature.

We now define the tile semantics for our connectors. As usual for tiles, we first fix the categories of configurations and of observations and then we give the basic tiles.

As explained in Section 1, the objects of our categories are natural numbers.

The horizontal category of configurations is the free symmetric (strict) monoidal category generated by the basic connectors. The basic connector γ is the symmetry $\gamma_{1,1}$. We call *connector* any arrow in the horizontal category. Given a connector $\alpha \colon n \to m$ we denote by $\alpha^c \colon m \to n$ its dual, defined in the obvious way for basic connectors (see Figure 3) and then inductively by $(\alpha;\beta)^c = \beta^c;\alpha^c$ and $(\alpha \otimes \beta)^c = \alpha^c \otimes \beta^c$.

The vertical category is the free monoidal category generated by the arrows $\mathsf{tick} : 1 \to 1$ and $\mathsf{untick} : 1 \to 1$.

The tiles defining the semantics of ordinary connectors are in Figure 4. The first rule specifies that a symmetry can accept any input pair which is swapped in the output. Then there are the two rules for duplicator, where the constraint is that all the actions

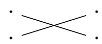
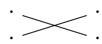
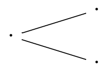
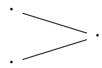
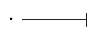
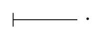
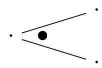
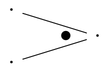
Ordinary structure			Dual structure		
name	symbolic	graphical	name	symbolic	graphical
symmetry	$\gamma\colon 2 \to 2$		**symmetry**	$\gamma\colon 2 \to 2$	
duplicator	$\nabla\colon 1 \to 2$		**coduplicator**	$\Delta\colon 2 \to 1$	
bang	$!\colon 1 \to 0$		**cobang**	$¡\colon 0 \to 1$	
mex	$\dot{\nabla}\colon 1 \to 2$		**comex**	$\dot{\Delta}\colon 2 \to 1$	
zero	$\mathbf{0}\colon 1 \to 0$		**cozero**	$\overline{\mathbf{0}}\colon 0 \to 1$	

Fig. 3. Syntax of basic connectors.

$$\gamma \xrightarrow[y\otimes x]{x\otimes y} \gamma \text{ where } x,y \in \{\mathsf{tick},\mathsf{untick}\} \qquad \nabla \xrightarrow[\mathsf{tick}\otimes\mathsf{tick}]{\mathsf{tick}} \nabla \qquad \nabla \xrightarrow[\mathsf{untick}\otimes\mathsf{untick}]{\mathsf{untick}} \nabla \qquad \mathbf{0} \xrightarrow[id_0]{\mathsf{untick}} \mathbf{0}$$

$$! \xrightarrow[id_0]{\mathsf{tick}} ! \qquad ! \xrightarrow[id_0]{\mathsf{untick}} ! \qquad \dot{\nabla} \xrightarrow[\mathsf{untick}\otimes\mathsf{untick}]{\mathsf{untick}} \dot{\nabla} \qquad \dot{\nabla} \xrightarrow[\mathsf{tick}\otimes\mathsf{untick}]{\mathsf{tick}} \dot{\nabla} \qquad \dot{\nabla} \xrightarrow[\mathsf{untick}\otimes\mathsf{tick}]{\mathsf{tick}} \dot{\nabla}$$

Fig. 4. Basic tiles for ordinary connectors.

must coincide. Last rule in the first row defines the only allowed behaviour for zero, which admits just untick on its interface. Rules in the second row specify the behaviour of bang, which hides any action on its interface, and mex: if the trigger is untick, then the effects are two unticks, otherwise the trigger tick is propagated to exactly one effect.

Dual connectors have symmetric tiles. For instance, the tiles for Δ are:

$$\Delta \xrightarrow[\mathsf{tick}]{\mathsf{tick}\otimes\mathsf{tick}} \Delta \qquad \Delta \xrightarrow[\mathsf{untick}]{\mathsf{untick}\otimes\mathsf{untick}} \Delta$$

From the tile system we can derive an observational semantics using tile bisimilarity. This semantics is compositional, as proved by the following theorem.

Theorem 2.1. *In all the tile systems built using only the above tiles for connectors, $\simeq_{\mathrm{t}}$ is a congruence (w.r.t. parallel and sequential composition).*

Proof. Trivial, since all the basic tiles satisfy the basic source property. □

It is worth noting that the axioms for symmetry "bisimulate", in the sense that the left hand side and the right hand side of each axiom are tile bisimilar.

The coordination policy of a connector $\alpha\colon n \to m$ can be represented as a $2^n \times 2^m$ tick-table whose cells contain boolean values. Each row (resp. column) represents a combination of tick/untick values (denoted as 1 or 0 in the tick-tables) for the n inputs

id	0	1
0	✓	
1		✓

γ	00	01	10	11
00	✓			
01			✓	
10		✓		
11				✓

∇	00	01	10	11
0	✓			
1				✓

!	$\varnothing$
0	✓
1	✓

$\bar{\nabla}$	00	01	10	11
0	✓			
1		✓	✓	

0	$\varnothing$
0	✓
1	

Fig. 5. Denotational semantics.

(resp. m outputs). If a cell is true (i.e., marked), then the corresponding combination of inputs and outputs is admissible, otherwise (the cell is false, i.e., empty, unmarked) the corresponding combination of inputs and outputs is forbidden. The tick-tables for basic connectors are in Figure 5 (dual basic connectors have transposed tables).

We denote with $T(\alpha)$ the tick-table associated to connector α. Furthermore, given a position $[i, j]$ in a tick-table T we denote with $d_T([i, j])$ its *domain*, that is the set of elements in its input and output interfaces on which tick actions are performed.

A connector $\alpha\colon n \to m$ can be seen as an hypergraph where basic connectors are edges and elements of interfaces are nodes. The solution of the network of constraints S associated with α is the set of consistent assignments of tick/untick values to all the nodes appearing in the graph denoted by α in such way that a corresponding "tiling" can be found. However this semantics is too *concrete* when one is not interested in knowing the way in which all constraints of the network are satisfied. A more abstract semantics of α is the solution of the network, where all the internal nodes have been existentially quantified. Thus tick-tables can be seen as the *denotational semantics* of connectors.

Next lemma shows the effect on tick-tables of the operations on connectors.

Lemma 2.2. *For any two connectors $\alpha\colon n \to h, \beta\colon h \to m$, $T(\alpha;\beta)$ is the product matrix $T(\alpha) \times T(\beta)$, i.e., $T(\alpha;\beta)[i, j] = \bigvee_k (T(\alpha)[i,k] \wedge T(\beta)[k, j])$. For any two connectors $\alpha\colon n \to h, \beta\colon l \to m$, $T(\alpha \otimes \beta)$ is obtained by refining each marked entry of $T(\beta)$ by a copy of $T(\alpha)$, and each unmarked entry of $T(\beta)$ by the empty table with the same dimension as $T(\alpha)$. Moreover, for any connector α, $T(\alpha^c)$ is the transposition of $T(\alpha)$.*

The denotational semantics of connectors given by tick-tables agrees with the observational semantics defined by tile bisimilarity, that is two connectors are tile bisimilar iff they have the same associated tick-table.

Theorem 2.3. *For each pair of connectors α and β, $\alpha \simeq_t \beta$ iff $T(\alpha) = T(\beta)$.*

Proof. Since all connectors are stateless, two connectors are tile bisimilar iff their allowed combinations of ticks and unticks on the interfaces are equal, i.e., iff their tables are equal. □

3 Normal form

We first show an axiomatization of connectors which is correct and complete w.r.t. their denotational semantics, and then we show an algorithm to derive a standard representative for each equivalence class, i.e., a normal form. From the categorical point of view, this corresponds to compute the colimit of a diagram (such as a CommUnity one).

Several axioms for connectors have been proposed, studied and applied in the literature, see e.g. [1,5,25,6,3]. Axioms over connectors are usually aimed at characterizing a category of *links between objects* as the equational term algebra freely generated from a restricted set of basic connectors. Usually the axioms have just to consider the few possible ways in which two or three basic connectors can be composed together. However, our algebra is very rich and thus a few more complex patterns need to be considered.

The consistency of all the axioms we are going to present w.r.t. the denotational semantics can be checked just by looking at the tables associated to each term. More precisely, for each axiom $\alpha = \beta$ that we propose, it is easy to check that $T(\alpha) = T(\beta)$.

Notation. Given a set of connectors S we denote with $\mathbf{CC}(S)$ the class of connectors generated by connectors in S. Note that symmetries are always included in $\mathbf{CC}(S)$, even when $S = \varnothing$.

Two edges are *adjacent* if they share a node in the graph representation of the connector. An edge is adjacent to any node in its interfaces. A *path* in the graph is a sequence of nodes $\{n_i | i \in \{1,\ldots,n\}\}$ such that for each $i \in \{1,\ldots,n-1\}$ n_i is an element of the input interface of a basic connector and n_{i+1} is an element of the output interface of the same basic connector if the connector is not a symmetry. As suggested by their graphical representation, for symmetries the path can only enter in the first element of the input interface and exit from the second one in the output interface, or enter from the second one and exit from the first one. The *components* of a path are all its nodes and all the edges traversed. We say that two components of a graph are *linearly connected* iff there exists a path of which they are both components. The relation of *connectedness* is the transitive closure of the relation of linear connectedness.

We let ∇^n denote the "tree" of ∇ connectors with n leaves, inductively defined as $\nabla^0 = !$ and $\nabla^{n+1} = \nabla; id \otimes \nabla^n$. Note that $\nabla^1 = id$. We also define connectors for structured objects in terms of connectors defined for smaller objects:

$$\nabla_0 = id_0 \qquad \nabla_{n+1} = \nabla \otimes \nabla_n; id \otimes \gamma_{1,n} \otimes id_n \qquad !_0 = id_0 \qquad !_{n+1} = ! \otimes !_n$$

Note that $\nabla_1 = \nabla$ and $!_1 = !$. Similar notations are used for the other connectors.

3.1 Connectors for synchronization

First, we focus on the class of connectors $\mathbf{CC}(\nabla, \Delta, !, \mathsf{i})$. The tick-tables associated to these connectors can be characterized as below.

Proposition 3.1. *Let* $\alpha \in \mathbf{CC}(\nabla, \Delta, !, \mathsf{i})$. *Then* $T(\alpha)$ *satisfies the following properties.*

- $T(\alpha)[\vec{0}, \vec{0}] = \checkmark$;
- *suppose* $T(\alpha)[i_1, j_1] = \checkmark$ *and* $T(\alpha)[i_2, j_2] = \checkmark$;
  - *if* $d_{T(\alpha)}([i,j]) = d_{T(\alpha)}([i_1,j_1]) \cup d_{T(\alpha)}([i_2,j_2])$ *then* $T(\alpha)[i,j] = \checkmark$
  - *if* $d_{T(\alpha)}([i,j]) = d_{T(\alpha)}([i_1,j_1]) \cap d_{T(\alpha)}([i_2,j_2])$ *then* $T(\alpha)[i,j] = \checkmark$
  - *if* $d_{T(\alpha)}([i,j]) = d_{T(\alpha)}([i_1,j_1]) \smallsetminus d_{T(\alpha)}([i_2,j_2])$ *then* $T(\alpha)[i,j] = \checkmark$
  - *if* $d_{T(\alpha)}([i,j]) = \overline{d_{T(\alpha)}([i_1,j_1])}$ *then* $T(\alpha)[i,j] = \checkmark$

Proposition 3.1 says that the cell with empty domain is always enabled and that table entries are closed under domain union, intersection, difference and complement.

Intuitively, the last four properties are true because connectors built of synch and hiding connectors individuate equivalence classes on the elements of the interfaces (connected elements are in the same class), different equivalence classes act independently and domains are unions of such classes. It is an easy consequence of the proposition that, for instance, for any $\alpha \in \mathbf{CC}(\nabla, \Delta, !, \text{¡})$, $T(\alpha)[\vec{1}, \vec{1}] = \checkmark$.

We call *synch-tables* the tables that satisfy these properties.

Definition 3.2 (Base). *Given a synch-table T its base $b(T)$ is the set of the domains of its marked cells that are minimal w.r.t. set inclusion.*

The synch-tables are uniquely identified by their bases.

Lemma 3.3. *Let T_1 and T_2 be any two synch-tables with the same dimension. Then $T_1 = T_2$ iff $b(T_1) = b(T_2)$.*

Proof. The if part is trivial. The only if part follows from Proposition 3.1. □

Analogous structures have been already studied in the literature [5,25,6,3]. If we inspect which equalities are satisfied among those in [3], then according to the terminology therein, we have a gs-monoidal structure $(\nabla, !)$, a cogs-monoidal structure $(\Delta, \text{¡})$, a match-share structure (∇, Δ) and a new-bang structure $(!, \text{¡})$. The whole structure is called a p-monoidal structure. Interestingly, the p-monoidal axioms characterize exactly tile bisimilarity and allow for normal-form reduction. This is explained below in detail.

As far as the gs-structure is concerned there are three axioms expressing the "associativity", "commutativity" and "unit" for the ∇ (with ! as "unit").

$$\nabla; (id \otimes !) = id \qquad \nabla; \gamma = \nabla \qquad \nabla; (\nabla \otimes id) = \nabla; (id \otimes \nabla)$$

A cogs-monoidal structure is just a gs-monoidal structure in the dual category. Therefore the axioms are obtained by reversing the order of composition.

The axioms of match-share categories have been proposed in [3], where the free algebra of match-share connectors has been shown to model partition relations between non-empty source and target objects. There are three match-share axioms:

$$\nabla; \Delta = id \qquad \Delta; \nabla = (id \otimes \nabla); (\Delta \otimes id) \qquad \Delta; \nabla = (\nabla \otimes id); (id \otimes \Delta)$$

The leftmost axiom essentially says that the multiplicity of connections between two objects is not important. The other two axioms (which are in fact equivalent, thus one of them can be dropped) say that the path connecting two objects is not important.

The new-bang categories just contain the axiom $\text{¡}; ! = id_0$ which represents garbage-collection of isolated nodes.

We want to use the axioms to reduce any connector in a suitable normal form. We start by defining a *sorted form* that forces a standard order on connector layers.

Definition 3.4 (Sorted form). *A connector $\alpha \in \mathbf{CC}(\nabla, \Delta, !, \text{¡})$ is in sorted form iff*

$$\alpha \equiv \alpha_{\text{¡}}; \alpha_\gamma; \alpha_\Delta; \beta_\nabla; \beta_\gamma; \beta_!$$

where α_σ and β_σ are layers of σ and $\equiv$ is syntactic identity.

Proposition 3.5. *Any connector $\alpha \in \mathbf{CC}(\nabla, \Delta, !, \text{¡})$ can be transformed in sorted form using the axioms.*

Proof. The proof is by induction on the construction of the connector. Essentially, we have to prove that given a connector α in sorted form, we can transform in sorted form any connector $id_{n_1} \otimes \sigma \otimes id_{n_2}; \alpha$. For each σ one can find axioms that make it to commute with all the other connectors that it must traverse to reach its final position. □

We want now to define for connectors a *normal form* which is strictly related to tick-tables. We first need an auxiliary definition.

Definition 3.6 (Central point). *A* central point *is any element of interface shared by layers* α_{Δ} *and* β_{∇}.

Definition 3.7 (Normal form). *A connector* $\alpha \in \mathbf{CC}(\nabla, \Delta, !, ¡)$ *is in normal form iff:*

1. *it is in sorted form;*
2. *hiding connectors have central points as interface;*
3. *each central point is linearly connected to at least an external interface.*

Theorem 3.8. *Any connector* $\alpha \in \mathbf{CC}(\nabla, \Delta, !, ¡)$ *can be transformed in normal form using the axioms.*

Proof. Trivial, using Proposition 3.5. □

The theorems below show the connection between normal forms and synch-tables.

Theorem 3.9. *For each synch-table T, we can build a connector* $\alpha \in \mathbf{CC}(\nabla, \Delta, !, ¡)$ *in normal form such that $T = T(\alpha)$. Moreover the construction is unique up to the axioms of symmetric monoidal categories and of associativity and commutativity of synch connectors.*

Proof. Let $b(T)$ be the base of T. We build α in the following way:

- we create a central point P_b for each element $b \in b(T)$;
- we build a tree Δ^n (resp. ∇^m) on the left (resp. right) of each central point P_b, where n (resp. m) is the number of elements in b that are in the left (resp. right) interface;
- we add permutations to connect the trees Δ^n and ∇^m to the corresponding elements of the interfaces. □

Theorem 3.10. *We have a bijective correspondence between synch-tables and connectors in* $\mathbf{CC}(\nabla, \Delta, !, ¡)$ *up to the axioms.*

Proof. The proof is done by showing that the function from synch-tables to connectors up to the axioms defined in Theorem 3.9 is bijective. □

3.2 Adding the mutual exclusion connector

As we have already seen, connectors in $\mathbf{CC}(\nabla, \Delta, !, ¡)$ allow to specify only a small class of tick-tables. In particular, we can express synchronization constraints but not mutual exclusion constraints. This is proved by the fact that the class $\mathbf{CC}(\nabla, \Delta, !, ¡)$ has limited expressiveness. For instance, it is not expressive enough to model all CommUnity connectors. To solve that problem we will add the mutual exclusion connector $\overline{\nabla} : 2 \to 1$.

$$
\begin{aligned}
\boldsymbol{\nabla};\boldsymbol{\Delta} &= id && (1)\\
\nabla;\boldsymbol{\Delta} &= \mathbf{0};\overline{\mathbf{0}} && (2)\\
\boldsymbol{\Delta};\nabla &= \nabla_2;\boldsymbol{\Delta}\otimes\boldsymbol{\Delta} && (3)\\
\boldsymbol{\Delta};\mathbf{0} &= \mathbf{0}\otimes\mathbf{0} && (4)\\
\nabla;id\otimes\mathbf{0} &= \mathbf{0};\overline{\mathbf{0}} && (5)\\
\Delta;\mathbf{0} &= \mathbf{0}\otimes\mathbf{0} && (6)\\
\boldsymbol{\nabla};!_2 &= ! && (7)\\
\nabla;\boldsymbol{\nabla}\otimes id &= \boldsymbol{\nabla};\nabla\otimes\nabla;id\otimes\boldsymbol{\Delta}\otimes id;id\otimes\gamma && (8)\\
\boldsymbol{\nabla};\nabla\otimes id &= \nabla;\boldsymbol{\nabla}\otimes\boldsymbol{\nabla};id\otimes\Delta\otimes id;id\otimes\gamma && (9)\\
\boldsymbol{\Delta};\boldsymbol{\nabla} &= \boldsymbol{\nabla}_2;\nabla\otimes\nabla\otimes\nabla\otimes\nabla;id\otimes\boldsymbol{\Delta}\otimes(\boldsymbol{\Delta};!)\otimes\boldsymbol{\Delta}\otimes id;\gamma\otimes\gamma;id\otimes(\boldsymbol{\Delta};!)\otimes id && (10)\\
¡;\boldsymbol{\nabla};\boldsymbol{\nabla}\otimes id &= ¡_3;\boldsymbol{\nabla}\otimes\boldsymbol{\nabla}\otimes\boldsymbol{\nabla};id\otimes\gamma\otimes\gamma\otimes id;\Delta\otimes\Delta\otimes\Delta && (11)\\
id_2 &= \boldsymbol{\nabla}\otimes\boldsymbol{\nabla};id\otimes\Delta\otimes id;id\otimes\nabla\otimes id;\boldsymbol{\Delta}\otimes\boldsymbol{\Delta} && (12)\\
id_2 &= \boldsymbol{\nabla}\otimes(¡;\boldsymbol{\nabla})\otimes\boldsymbol{\nabla};id\otimes\gamma\otimes\gamma\otimes id;\Delta\otimes\Delta\otimes\Delta;id\otimes\nabla\otimes id;\boldsymbol{\Delta}\otimes\boldsymbol{\Delta} && (13)\\
!_n &= id_n\otimes ¡;id_n\otimes\nabla^n;\boldsymbol{\Delta}_n;!_n && (14)
\end{aligned}
$$

Fig. 6. Axioms for mutual exclusion, textually.

Following the analogy with Section 3.1, one may think that also the dual connector $\boldsymbol{\Delta}$ must be explicitly introduced, but this is not strictly required since the complex term $id\otimes(¡;\nabla)\otimes id;id\otimes\boldsymbol{\nabla}\otimes id_2;id_2\otimes\gamma\otimes id;\Delta\otimes id\otimes\Delta;!\otimes id\otimes!$ exhibits the same behaviour as $\boldsymbol{\Delta}$. Similarly both inaction connectors $\mathbf{0}$ and $\overline{\mathbf{0}}$ can be derived as auxiliary connectors. In fact we have for instance $T(\mathbf{0}) = T(\boldsymbol{\nabla};\Delta;!)$.

One may start considering just the axiomatization of choice and inaction connectors, separately w.r.t. synch and hiding connectors. Thus one individuates a gs-monoidal structure $(\boldsymbol{\nabla},\mathbf{0})$, a cogs-monoidal structure $(\boldsymbol{\Delta},\overline{\mathbf{0}})$ and a new-bang structure $(\mathbf{0},\overline{\mathbf{0}})$. Unluckily no simple axiomatization can be found for $(\boldsymbol{\nabla},\boldsymbol{\Delta})$, since they form neither a match-share category since $T(\boldsymbol{\Delta};\boldsymbol{\nabla}) \neq T(\boldsymbol{\nabla}\otimes id;id\otimes\boldsymbol{\Delta})$ nor an r-monoidal [3] category since $T(\boldsymbol{\Delta};\boldsymbol{\nabla}) \neq T(\boldsymbol{\nabla}\otimes\boldsymbol{\nabla};id\otimes\gamma\otimes id;\boldsymbol{\Delta}\otimes\boldsymbol{\Delta})$.

Thus we resort to a complex axiomatization that deals with all the four classes of connectors at the same time. The axioms are textually written in Figure 6. For simplicity, dual axioms are omitted. Axioms 1–9 are quite simple. The other ones, which are more complex, are depicted in Figure 7 and commented below. The last one, which is actually an axiom scheme, is drawn only for $n = 3$. Axiom 10 deals with commutation of $\boldsymbol{\Delta}$ and $\boldsymbol{\nabla}$, but w.r.t. the conceptually similar axiom 3, we have to force mutual exclusion on all the paths. Axiom 11 shows that mutual exclusion on three actions can be enforced by imposing mutual exclusion separately on each pair of actions. Axiom 12 shows that given two independent actions we can freely add an action for their synchronized execution and in that case axiom 13 says that we can also force mutual exclusion on the two paths corresponding to the asynchronous execution of the two starting actions. Finally, axiom 14 means that if all the elements of the interfaces of a connector are adjacent to a node in the interface of a connector $\boldsymbol{\Delta};!$ (or of the dual form), then for each denotation we can obtain a concrete correct behaviour by performing an untick on each internal node, thus there is no real constraint on the behaviour of the elements of the interfaces, which can be considered disconnected and closed by an hiding connector.

We present here some useful equivalence lemmas.

Lemma 3.11. $\boldsymbol{\nabla}_n;\Delta^n \otimes \Delta^n = \Delta^n;\boldsymbol{\nabla}$.

Proof. The proof is by induction on n. □

Lemma 3.12. *For each connector* $\alpha : m \to n$ *let* $\alpha^c : n \to m$ *be its dual connector. We have* $\alpha \otimes id_n;\Delta_n;!_n = id_m \otimes \alpha^c;\Delta_m;!_m$.

Proof. The proof is by induction on the number of basic connectors in α. □

Also for $\mathbf{CC}(\nabla,\Delta,!,¡,\boldsymbol{\nabla})$ we can define a sorted form and a normal form.

Definition 3.13 (Sorted form). *A connector* $\alpha \in \mathbf{CC}(\nabla,\Delta,!,¡,\boldsymbol{\nabla})$ *is in sorted form iff:*

$$\alpha \equiv \alpha_{¡};\alpha_{\overline{\mathbf{0}}};\alpha_{\boldsymbol{\nabla}};\alpha_\gamma;\alpha_{\boldsymbol{\Delta}};\beta_\nabla;\beta_\gamma;\beta_{\boldsymbol{\Delta}};\beta_{\mathbf{0}};\beta_!$$

where α_σ *and* β_σ *are layers of* σ *and* $\equiv$ *is syntactic identity.*

Note that the definition of central point (Definition 3.6) can be applied also to this new sorted form. Central points can be linearly connected to both *free variables* (i.e., external interfaces) and *hidden variables* (i.e., interfaces of hiding connectors).

Proposition 3.14. *Any connector* $\alpha \in \mathbf{CC}(\nabla,\Delta,!,¡,\boldsymbol{\nabla})$ *can be transformed in sorted form using the axioms.*

Proof. The proof strategy is the same of Proposition 3.5, but here when moving connectors to their final layer, other basic connectors may be created, and in this case we have to check that the procedure indeed terminates.

In most cases the proof can be done by induction on the width of layer that must be traversed, that is on the maximum number of connectors in all the paths of the layer. The complex case is the one dealing with connectors ∇ and $\boldsymbol{\Delta}$.

The only risk to cycle is when a ∇ connector while traversing layer $\alpha_{\boldsymbol{\nabla}}$ creates a $\boldsymbol{\Delta}$ connector that while traversing layer $\alpha_{\boldsymbol{\Delta}}$ creates again a $\boldsymbol{\nabla}$ connector that goes back to the original ∇ connector. One can see that this may happen only if there are two different paths starting from the same ∇ connector, one for each element of its right interface, which arrive to the same Δ connector. These paths can be deleted by isolating either the connector $\nabla;\boldsymbol{\nabla}\otimes\boldsymbol{\nabla};id\otimes\Delta\otimes id$ or the connector $\boldsymbol{\nabla};\Delta$, which can be transformed using the axioms respectively into $\boldsymbol{\nabla};\nabla\otimes id;id\otimes\gamma$ and into $\mathbf{0};\overline{\mathbf{0}}$. □

Definition 3.15 (Normal form). *A connector* $\alpha \in \mathbf{CC}(\nabla,\Delta,!,¡,\boldsymbol{\nabla})$ *is in normal form if and only if:*

1. α *has the form* $\alpha_{¡};\alpha_{\mathbf{0}};\alpha_{\boldsymbol{\nabla}};\alpha_\gamma;\alpha_{\boldsymbol{\Delta}};\beta_\nabla;\beta_\gamma;\beta_{\boldsymbol{\Delta}};\beta_{\overline{\mathbf{0}}};\beta_!$*;*
2. *hiding connectors are adjacent to either roots of mex trees or central points;*
3. *there exists at most one path between a fixed central point and a fixed variable;*
4. *no two central points are linearly connected to exactly the same set of variables;*

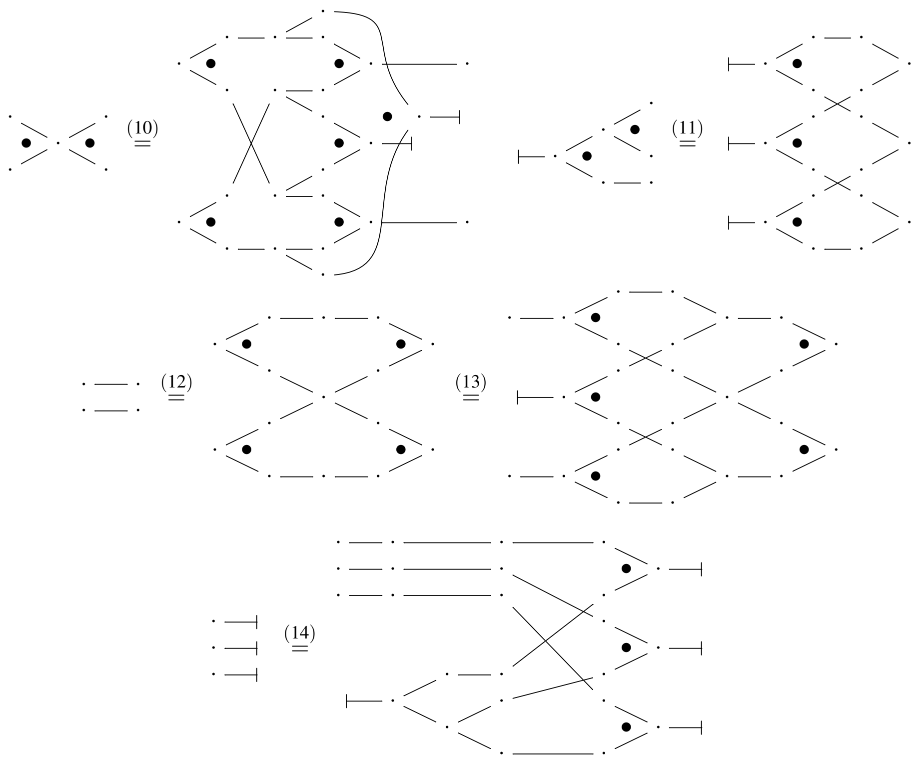


Fig. 7. Complex axioms for mutual exclusion, graphically.

5. *each central point is linearly connected to at least one free variable;*
6. *each hidden variable is linearly connected to at most two central points;*
7. *no two hidden variables are linearly connected to the same set of central points;*
8. *each pair of central points associated with disjoint sets of free variables is linearly connected to an hidden variable;*
9. *hidden variables are on the left of central points, unless they are adjacent to them.*

Theorem 3.16. *Any connector* $\alpha \in \mathbf{CC}(\nabla, \Delta, !, \text{!`}, \nabla\!\!\!\!\nabla)$ *can be transformed in normal form using the axioms.*

Proof. Thanks to Proposition 3.14, α can be transformed in sorted form. Then the conditions can be satisfied one at the time using the axioms. □

Again, there is a precise correspondence between normal forms and tick-tables.

Theorem 3.17. *For any* tick-*table* T *with* $T[\vec{0},\vec{0}] = \checkmark$*, we can build a connector* α *in normal form such that* $T(\alpha) = T$*. Moreover the construction is unique up to the axioms of symmetric monoidal categories and of associativity and commutativity of synch and choice connectors.*

Proof. Given a tick-table T, the connector α is realized in the following way:

– variables that always have value 0 are connected to $\mathbf{0}$ or $\overline{\mathbf{0}}$ connectors;
– for other input (resp. output) variables, we build a tree of $\nabla\!\!\!\!\nabla$ (resp. $\Delta\!\!\!\!\Delta$) with as many leaves as the number of checked cells that have that variable in the domain;
– for each pair of central points with disjoint sets of free variables, we create an hidden variable for them;
– for each checked cell in the table, we create a central point with two outgoing trees of synch connectors. The number of leaves is the number of free variables (input on the left, output on the right) in the domain, plus (on the left) the number of hidden variables associated to the central point;
– we connect leaves of synchronization trees with leaves of mutual exclusion trees using permutations, connecting each central point to the associated variables. □

The theorem below establishes a bijective correspondence between denotations and standard implementations of connectors.

Theorem 3.18. *We have a bijective correspondence between* tick-*tables T with* $T[\vec{0},\vec{0}] = \checkmark$ *and connectors in* $\mathbf{CC}(\nabla, \Delta, !, ¡, \nabla\!\!\!\!\nabla)$ *up to the axioms.*

Proof. Analogously to Theorem 3.10, the proof is done by showing that the function from tick-tables to connectors up to the axioms defined in Theorem 3.17 is bijective.
□

These results can be used to extend the research in [2], where a mapping from CommUnity to the Tile Model was presented, and the main result was that the translation of a CommUnity diagram is tile bisimilar to the translation of its colimit. Using the correspondence between observational semantics and connectors up to the axioms we can state that the (synchronization part of the) translation of a CommUnity diagram is equal up to the axioms to the (synchronization part of the) translation of its colimit. More in general, colimit computation in the categorical approach is now strongly related to normalization using suitable axioms in the algebraic approach.

4 Conclusion and future work

We have presented different classes of connectors and we have shown how they can be analyzed from different points of view: their concrete structures can be described by graphs, their operational and observational semantics are given using tiles and tile bisimilarity, while the representation based on tick-tables provides a denotational semantics. We have proved that there is a bijective correspondence among connectors up to axioms, classes of bisimilar connectors and denotations. This allows to extend the result of [2] proving that there is a correspondence between colimit computation in a categorical framework and normalization up to the axioms in an algebraic framework.

Our work leaves an open problem: we argue that the axiom schema 14 (see Figure 7) is needed for the completeness of the axiomatization, but we do not know whether a different finite axiomatization of $\mathbf{CC}(\nabla, \Delta, !, ¡, \nabla\!\!\!\!\nabla)$ exists or not.

As future work we plan to study the complexity of our reduction to normal form. Furthermore we want to generalize our connectors to a setting where we have a richer set *Act* of actions ruled by a synchronization algebra, instead of just two possible observations (tick/untick). Another interesting extension is given by probabilistic connectors.

References

1. J.A. Bergstra, C.A. Middelburg, and G. Stefanescu. Network algebra for asynchronous dataflow. *International Journal of Computer Mathematics*, 65:57–88, 1997.
2. R. Bruni, J.L. Fiadeiro, I. Lanese, A. Lopes, and U. Montanari. New insights on architectural connectors. In *Proc. IFIP TCS 2004*, pp. 367–379. Kluwer Academics, 2004.
3. R. Bruni, F. Gadducci, and U. Montanari. Normal forms for algebras of connections. *Theoret. Comput. Sci.*, 286(2):247–292, 2002.
4. R. Bruni, I. Lanese, and U. Montanari. Normal forms for stateless connectors. Tech. Rep. TR-05-11, Computer Science Department, University of Pisa, Italy.
5. V.E. Cazanescu and G. Stefanescu. Towards a new algebraic foundation of flowchart scheme theory. *Fundamenta Informaticae*, 13:171–210, 1990.
6. A. Corradini and F. Gadducci. An algebraic presentation of term graphs, via gs-monoidal categories. *Applied Categorical Structures*, 7:299–331, 1999.
7. A. Corradini and U. Montanari. An algebraic semantics for structured transition systems and its application to logic programs. *Theoret. Comput. Sci.*, 103:51–106, 1992.
8. P. Degano and U. Montanari. A model for distributed systems based on graph rewriting. *Journal of the ACM*, 34(2):411–449, 1987.
9. H. Ehrig, M. Pfender, and H. J. Schneider. Graph grammars: an algebraic approach. In *Proc. IEEE Conference on Automata and Switching Theory*, pp. 167–180, 1973.
10. J. L. Fiadeiro. *Categories for Software Engineering*. Springer, 2004.
11. J.L. Fiadeiro, A. Lopes, and M. Wermelinger. A mathematical semantics for architectural connectors. In *Generic Programming*, *LNCS* 2793, pp. 190–234. Springer, 2003.
12. F. Gadducci and U. Montanari. The tile model. In *Proof, Language and Interaction: Essays in Honour of Robin Milner*, pp. 133–166. MIT Press, 2000.
13. J. A. Goguen. Categorical foundations for general systems theory. In *Advances in Cybernetics and Systems Research*, pp. 121–130. Transcripta Books, 1973.
14. C.A.R. Hoare. *CSP – Communicating Sequential Processes*. International Series in Computer Science. Prentice-Hall, 1985.
15. P. Katis, N. Sabadini, and R.F.C. Walters. Bicategories of Processes. *Journal of Pure and Applied Algebra*, 115:141–178, 1997.
16. Y. Lafont. Interaction combinators. *Inform. and Comput.*, 137(1):69–101, 1997.
17. K.G. Larsen and L. Xinxin. Compositionality through an operational semantics of contexts. In *Proc. ICALP'90*, *LNCS* 443, pp. 526–539. Springer, 1990.
18. S. MacLane. *Categories for the Working Mathematician*. Springer, 1971.
19. J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoret. Comput. Sci.*, 96:73–155, 1992.
20. R. Milner. *A Calculus of Communicating Systems*, *LNCS* 92. Springer, 1989.
21. R. Milner. Turing, computation and communication. Turing anniversary lecture, 1997.
22. R. Milner. Bigraphical reactive systems. In *Proc. CONCUR 2001*, *LNCS* 2154, pp. 16–35. Springer, 2001.
23. G.D. Plotkin. A structural approach to operational semantics. Tech. Rep. DAIMI FN-19, Aarhus University, 1981.
24. A. Rensink. Bisimilarity of open terms. *Inform. and Comput.*, 156(1/2):345–385, 2000.
25. G. Stefanescu. *Network Algebra*. Discrete Math. and Theoret. Comp. Sci., Springer, 2000.