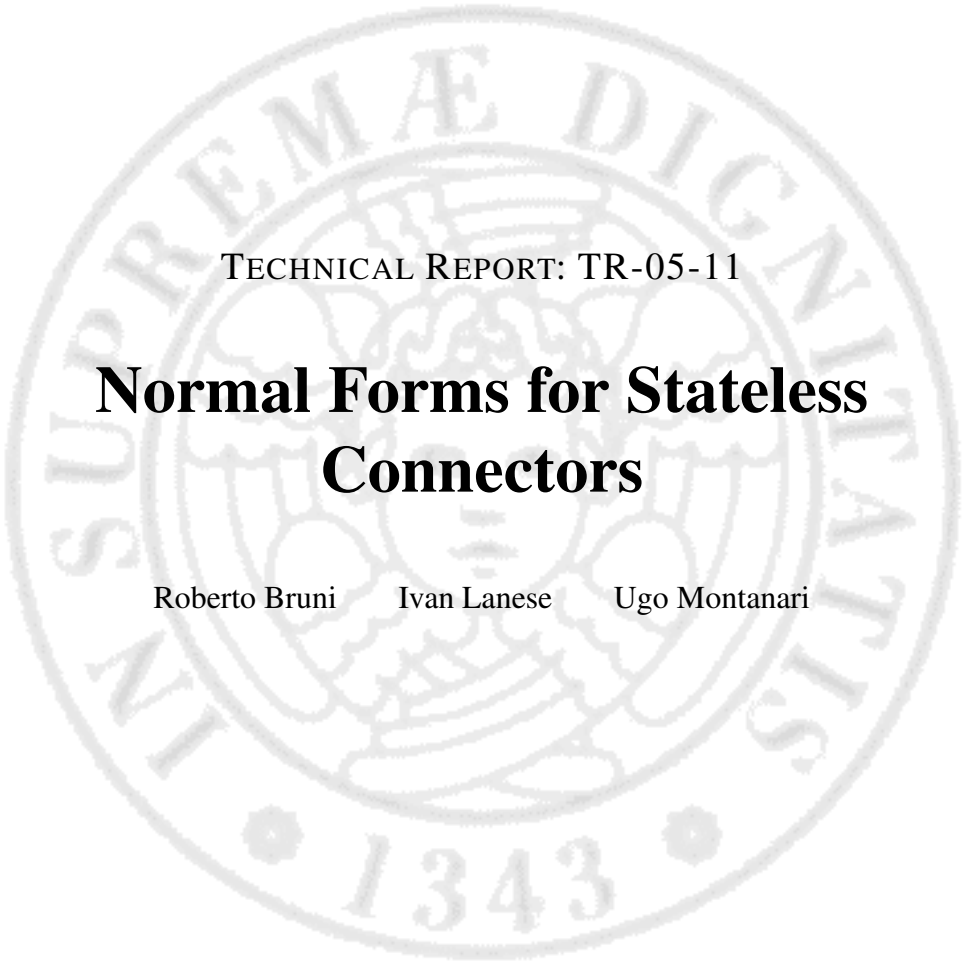
UNIVERSITÀ DI PISA
DIPARTIMENTO DI INFORMATICA



TECHNICAL REPORT: TR-05-11

Normal Forms for Stateless Connectors

Roberto Bruni Ivan Lanese Ugo Montanari

May 3, 2005

ADDRESS: via F. Buonarroti 2, 56127 Pisa, Italy. TEL: +39 050 2212700 FAX: +39 050 2212726

Normal Forms for Stateless Connectors*

Roberto Bruni, Ivan Lanese, and Ugo Montanari

Dipartimento di Informatica, Università di Pisa, Italia.
{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 [22], "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* has emerged in different contexts, with slightly different meanings, expressiveness and functionalities. The common trait is the role of a connector: a (software, architectural, process) component that mediates the interaction of other computational components and connectors. In particular, connectors have been studied within both *algebraic* and *categorical approaches*, two important frameworks for system modeling.

The algebraic approach [15,21] models systems as terms in a suitable term algebra, with constants modeling basic components that can be composed through the other operators, e.g., parallel composition and name restriction. Operational and abstract semantics are then usually based on a labelled transition system defined by structural induction.

The categorical approach [14] 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

* Research supported by the FET-GC Project IST-2001-32747 AGILE.

colimit, building an object in the same category that behaves as the whole system and that is uniquely determined up to isomorphisms. 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, relying e.g. on the study of observational semantics of process contexts [18,24,12,13,17], or on the analysis of suitable equational theories and reduction to normal forms [25,6,4,16].

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 operational semantics is expressed using the Tile Model [12]. The observational semantics we select is tile bisimilarity (see Definition 1.4), 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 we have defined an encoding from CommUnity into the Tile Model. One of the main results of [2] is that the translation of a diagram is tile bisimilar to the translation of its colimit. Due to space limitation, we refer the interested reader to [2] for full details about the encoding.

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 [5,25,6,16], our main contribution is the introduction of the mutual exclusion connector, which allows to specify a wider range of possible synchronization policies. In this sense, we extend specifically a previous work presented at WADT'98 [3] (full version [4]). Furthermore, the semantics based on matrices is new and it provides a clean and concrete mathematical definition of connectors. Finally, we also provide a characterization of the classes of matrices that can be specified, both with and without mutual exclusion.

Structure of the paper. § 1 contains some background on the symmetric monoidal structure of connectors and on the Tile Model, although we assume the reader has some familiarity with the basics of category theory. § 2 presents syntax and semantics of our 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. Proofs are in Appendix A.

1 Background

Symmetric monoidal categories for connectors. It has been highlighted in the literature that distributed systems can be conveniently modeled as graphs [8,23,9] that straightforwardly account for the network distribution of processes, mobile agents, etc. The advantage of using (freely generated) symmetric monoidal categories for representing configuration graphs is three-fold. First, it introduces a suitable notion of (observable) interfaces for configurations. Second, it introduces 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 isomorphisms.

We recall that a *(strict) monoidal category* [19] $(\mathcal{C},\otimes,e)$ 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 $f\colon n\to m$. 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.

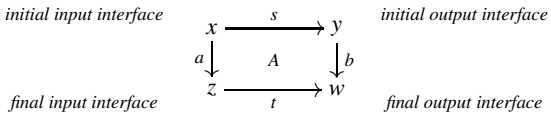


Fig. 1. Graphical representation of a tile A.

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

Essentially, 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, *context systems* [18], *structured transition systems* [7] and *rewriting logic* [20].

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 contribute by producing a, which acts as the *trigger* of A (see Figure 1). Triggers and effects are called *observations* and tile vertices are called *interfaces*.

Tiles can be composed horizontally, in parallel, and vertically to generate larger steps (see Figure 2). 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.

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 underlying 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.*

Like rewrite rules in rewriting logic, 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 horizontally, in parallel and vertically 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 3.

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 $\langle$trigger, effect$\rangle$ pairs

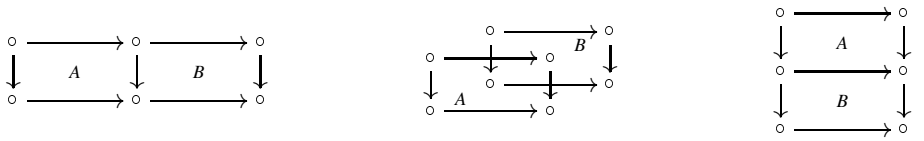


Fig. 2. Horizontal, parallel and vertical tile compositions.

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

Fig. 3. Inference rules for tile logic.

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 $\sim_t$ on configurations is called* tile bisimulation *if whenever $s \sim_t t$ and $\mathcal{R} \vdash s \overset{a}{\underset{b}{\to}} s'$, then t' exists such that $\mathcal{R} \vdash t \overset{a}{\underset{b}{\to}} t'$ and $s' \sim_t t'$.*

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

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 $\simeq_t$ 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 aiming to model systems where multiple actions can be executed at each time, either independently or synchronized. Connectors are used to ensure that the distributed components behave properly, i.e. to guarantee the global consistency of local evolutions. For instance, in the translation of CommUnity [2], the basic 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 $n-1$ unticks (forced inactivity) on their interfaces.

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, for which the ordinary coherence, naturality and functoriality axioms hold.

The complete list of connectors is in Figure 4. The ordinary basic connectors are in the leftmost part of the table, while their duals are on the right (symmetry is selfdual).

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	$\forall\colon 1 \to 2$		**comex**	$\Delta\colon 2 \to 1$	
zero	$\mathbf{0}\colon 1 \to 0$		**cozero**	$\overline{\mathbf{0}}\colon 0 \to 1$	

Fig. 4. Syntax of basic connectors.

The term mex stands for "mutual exclusion". We also speak about synch connectors (∇ and Δ), choice connectors ($\forall$ and Δ), 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 as distinctive feature the extreme separation between computation and coordination.

We now define the tile semantics for our connectors. As usual for the Tile Model, we must first define the categories of configurations and of observations (sharing the same set of objects) and then 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 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 5. 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 must coincide. Last rule in the first row defines the only allowed behaviour for zero, which allows 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:

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

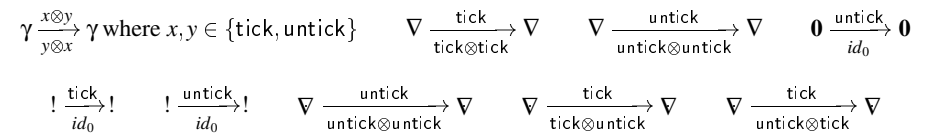


Fig. 5. Basic tiles for ordinary connectors.

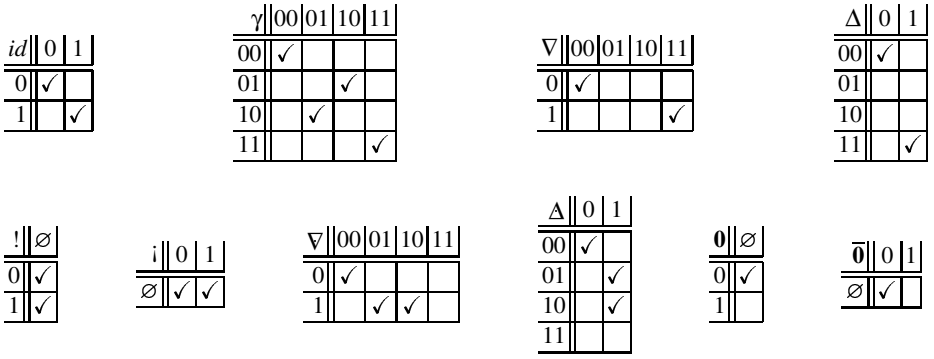


Fig. 6. Denotational semantics.

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).*

It is worth noting that the coherence and naturality axioms for symmetry "bisimulate", in the sense that the left hand side and the right hand side of each axiom are tile bisimilar and thus equated by the observational semantics.

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 represents a combination of tick/untick values (denoted as 1 or 0 in the tick-tables) for the n inputs, while each column represents a combination of tick/untick values for the 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 6.

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.

The solution of the network of constraints S associated with a connector $\alpha\colon n \to m$ 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 (i.e., neither inputs nor outputs) have been existentially quantified, that is the projection of the concrete semantics on the interfaces. Thus tick-tables can be seen as the *denotational semantics* of connectors.

Next lemma shows the operations on tick-tables that correspond to sequential and parallel composition of 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* $T(\alpha) = T(\beta)$.

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 an architectural diagram in CommUnity).

Several axioms for connectors have been proposed, studied and applied in the literature, see e.g. [1,5,25,6,4]. Axioms over connectors are usually aimed at characterizing a suitable category of *links between objects* as the equational term algebra freely generated from a restricted set of basic connectors, like those in Figure 4. 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 therefore 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 $S = \varnothing$.

A connector can be seen as an hypergraph where basic connectors are edges and elements of interfaces are nodes. Two edges are *adjacent* if they share a node. 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:

$$\begin{array}{ll} \nabla_0 = id_0 & \nabla_{n+1} = \nabla \otimes \nabla_n; id \otimes \gamma_{1,n} \otimes id_n \\ !_0 = id_0 & !_{n+1} = ! \otimes !_n \end{array}$$

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* $d_{T(\alpha)}([i,j]) = d_{T(\alpha)}([i_1, j_1]) \setminus d_{T(\alpha)}([i_2, j_2])$ *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. For example, it is an easy consequence that, for any $\alpha \in \mathbf{CC}(\nabla, \Delta, !, \mathsf{i})$, $T(\alpha)[\vec{1}, \vec{1}] = \checkmark$. We call *synch-tables* the tables that satisfy these properties. As we will show later (Theorem 3.10) any synch-table can be implemented by a connector in $\mathbf{CC}(\nabla, \Delta, !, \mathsf{i})$.

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* $b(T_1) = b(T_2)$.

Intuitively the previous lemma says that connectors built of synch and hiding connectors individuate equivalence classes on the elements of the interfaces, and that different equivalence classes act independently.

Analogous structures have been already studied in the literature [5,25,6,4]. If we inspect which equalities are satisfied among those in [4], then according to the terminology therein, we have a gs-monoidal structure $(\nabla, !)$, a cogs-monoidal structure (Δ, i), a match-share structure (∇, Δ) and a new-bang structure $(!, \mathsf{i})$. 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.

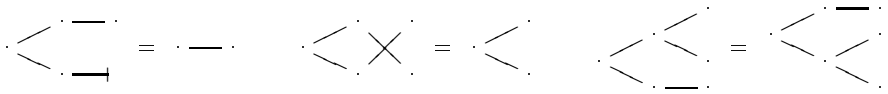


Fig. 7. Gs-monoidal axioms for ∇ and !.

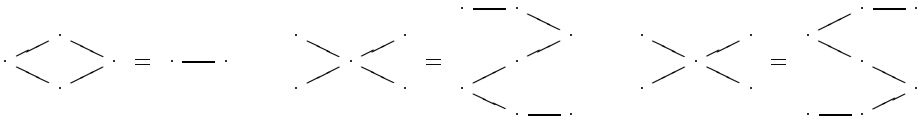


Fig. 8. Match-share axioms.

As far as the gs-structure is concerned there are three axioms expressing the "associativity", "commutativity" and "unit" for the ∇ (with ! as "unit"). The quoted terminology can be easily understood by looking at their graphical representation in Figure 7.

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

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:

$$(id\otimes ¡);\Delta = id \qquad \gamma;\Delta = \Delta \qquad (\Delta\otimes id);\Delta = (id\otimes\Delta);\Delta$$

The axioms of match-share categories have been proposed in [4], 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 (see Figure 8):

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

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 $¡;! = id_0$ 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* $\alpha \in \mathbf{CC}(\nabla,\Delta,!,¡)$ *is in sorted form iff*

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

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

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

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.*

The theorems below establish the correspondence 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.*

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

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 (in particular any synch-table is uniquely determined by its base). For instance, it is not expressive enough to model all CommUnity connectors. In order to solve that problem we will add the mutual exclusion connector $\nabla : 2 \to 1$.

Following the analogy with Section 3.1, one may think that also the dual connector Δ must be explicitly introduced, but this is not strictly required since the complex term $id \otimes (¡; \nabla) \otimes id; id \otimes \nabla \otimes id_2; id_2 \otimes \gamma \otimes id; \Delta \otimes id \otimes \Delta; ! \otimes id \otimes !$ (see Figure 9) exhibits the same behaviour as Δ. Similarly both inaction connectors $\mathbf{0}$ and $\overline{\mathbf{0}}$ can be derived as auxiliary connectors. In fact we have for instance $T(\mathbf{0}) = T(\nabla; \Delta; !)$ (see Figure 9).

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 $(\nabla, \mathbf{0})$, a cogs-monoidal structure $(\Delta, \overline{\mathbf{0}})$ and a new-bang structure $(\mathbf{0}, \overline{\mathbf{0}})$. Unluckily no simple axiomatization can be found for (∇, Δ), since they form neither a match-share category since $T(\Delta; \nabla) \neq T(\nabla \otimes id; id \otimes \Delta)$ nor an r-monoidal [4] category since $T(\Delta; \nabla) \neq T(\nabla \otimes \nabla; id \otimes \gamma \otimes id; \Delta \otimes \Delta)$.

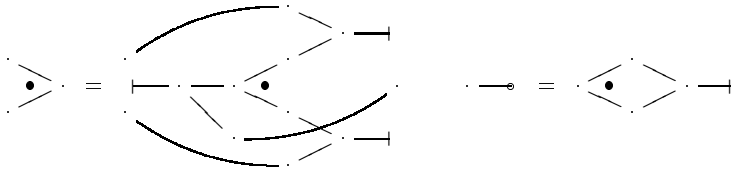


Fig. 9. Comex and zero as derived operators.

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

Fig. 10. Axioms for mutual exclusion, textually.

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 10. For simplicity, dual axioms are omitted. Axioms 1–9 are quite simple and their graphical representations are in Appendix (Figure 12). The other ones, which are more complex, are in Figure 11 and commented below. The last one, which is actually an axiom scheme, is drawn only for $n = 3$. Axiom 10 deals with commutation of $\underline{\Delta}$ and ∇, 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 $\underline{\Delta};!$ (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. $\nabla_n;\Delta^n \otimes \Delta^n = \Delta^n;\nabla$.

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

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

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

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

where α_σ *and* β_σ *are layers of* σ*s 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* $\alpha \in \mathbf{CC}(\nabla,\Delta,!,\mathsf{i},\nabla)$ *can be transformed in sorted form using the axioms.*

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

1. α *has the form* $\alpha_{\mathsf{i}};\alpha_{\mathbf{0}};\alpha_{\nabla};\alpha_{\gamma};\alpha_{\Delta};\beta_{\nabla};\beta_{\gamma};\beta_{\Delta};\beta_{\overline{\mathbf{0}}};\beta_{!}$ *(note that* $\mathbf{0}$ *and* $\overline{\mathbf{0}}$ *are swapped w.r.t. sorted form);*
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;*
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* $\alpha \in \mathbf{CC}(\nabla,\Delta,!,\mathsf{i},\nabla)$ *can be transformed in normal form 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.*

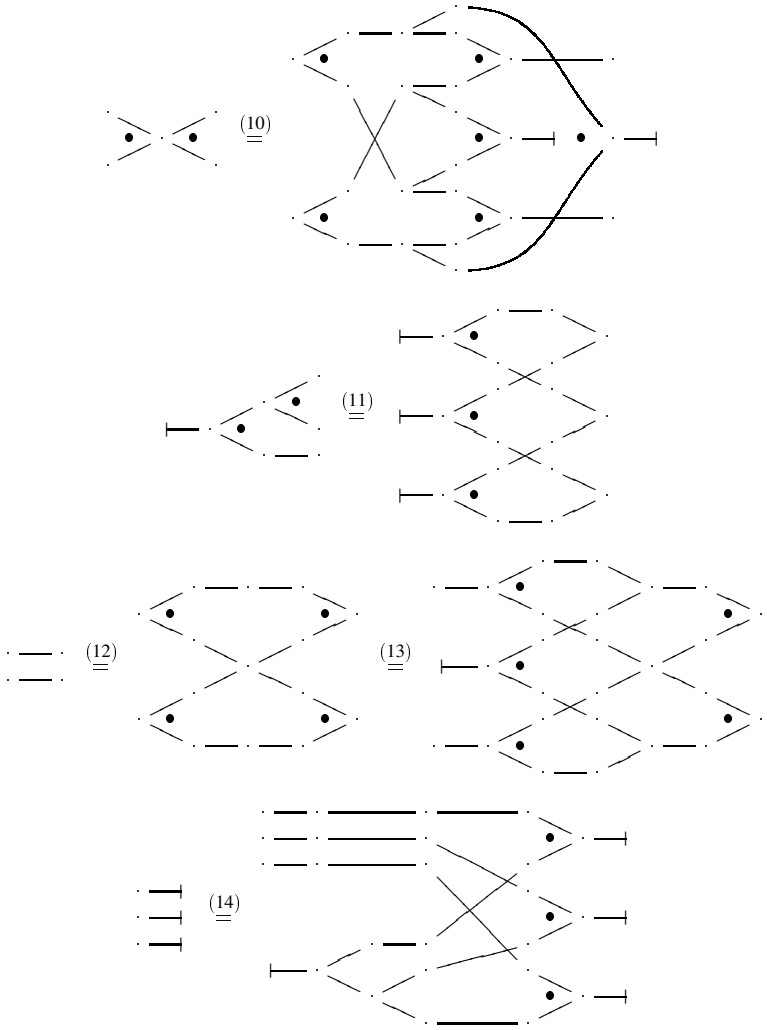


Fig. 11. Complex axioms for mutual exclusion, graphically.

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* $\mathbf{CC}(\nabla,\Delta,!,\mathsf{i},\nabla)$ *up to the axioms.*

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 suitable axioms, classes of bisimilar connectors and denotations, thus proving the coherence of all views. 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 11) is needed for the completeness of the axiomatization, but we do not know whether a different finite axiomatization of $\mathbf{CC}(\nabla, \Delta, !, \mathsf{i}, \nabla)$ 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 would be given by studying 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 Partitions and Relations. In *Proc. WADT'98*, *LNCS* 1589, pp. 31–47. Springer Verlag, 1999.
4. R. Bruni, F. Gadducci, and U. Montanari. Normal forms for algebras of connections. *Theoret. Comput. Sci.*, 286(2):247–292, 2002. Full version of [3].
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 Verlag, 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 Verlag, 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. F. Gadducci and U. Montanari. Comparing logics for rewriting: Rewriting logic, action calculi and tile logic. *Theoret. Comput. Sci.*, 285(2):319–358, 2002.
14. J. A. Goguen. Categorical foundations for general systems theory. In *Advances in Cybernetics and Systems Research*, pp. 121–130. Transcripta Books, 1973.
15. C.A.R. Hoare. *CSP – Communicating Sequential Processes*. International Series in Computer Science. Prentice-Hall, 1985.
16. P. Katis, N. Sabadini, and R.F.C. Walters. Bicategories of Processes. *Journal of Pure and Applied Algebra*, 115:141–178, 1997.
17. Y. Lafont. Interaction combinators. *Inform. and Comput.*, 137(1):69–101, 1997.
18. K.G. Larsen and L. Xinxin. Compositionality through an operational semantics of contexts. In *Proc. ICALP'90*, *LNCS* 443, pp. 526–539. Springer Verlag, 1990.
19. S. MacLane. *Categories for the Working Mathematician*. Springer Verlag, 1971.
20. J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoret. Comput. Sci.*, 96:73–155, 1992.
21. R. Milner. *A Calculus of Communicating Systems*, *LNCS* 92. Springer, 1989.
22. R. Milner. Turing, computation and communication. Turing anniversary lecture, October 1997. `http://www.fairdene.com/picalculus/milner-infomatics.pdf`.
23. R. Milner. Bigraphical reactive systems. In *Proc. CONCUR 2001*, *LNCS* 2154, pp. 16–35. Springer Verlag, 2001.
24. A. Rensink. Bisimilarity of open terms. *Inform. and Comput.*, 156(1/2):345–385, 2000.
25. G. Stefanescu. *Network Algebra*. Discrete Mathematics and Theoretical Computer Science. Springer Verlag, 2000.

A Proofs

Proof (of Theorem 2.1). To prove this theorem we need some basic results about tile system from the literature, that we summarize here briefly.

The *decomposition property* below ensures that $\simeq_t$ is a congruence.

Definition A.1. *A tile system $\mathcal{R} = (\mathcal{H}, \mathcal{V}, N, R)$ enjoys the* tile decomposition property *if for all $s \in \mathcal{H}$ and for all $\mathcal{R} \vdash s \xrightarrow[b]{a} t$: (i) if $s = s_1; s_2$ then there exist $c \in \mathcal{V}$ and $t_1, t_2 \in \mathcal{H}$ such that $\mathcal{R} \vdash s_1 \xrightarrow[c]{a} t_1$, $\mathcal{R} \vdash s_2 \xrightarrow[b]{c} t_2$ and $t = t_1; t_2$; and (ii) if $s = s_1 \otimes s_2$ then there exist $a_1, a_2, b_1, b_2 \in \mathcal{V}$ and $t_1, t_2 \in \mathcal{H}$ such that $\mathcal{R} \vdash s_1 \xrightarrow[b_1]{a_1} t_1$, $\mathcal{R} \vdash s_2 \xrightarrow[b_2]{a_2} t_2$ and $a = a_1 \otimes a_2$, $b = b_1 \otimes b_2$ and $t = t_1 \otimes t_2$.*

Proposition A.2 (cfr. [12]). *If the tile system $\mathcal{R}$ enjoys the tile decomposition property, then tile bisimilarity is a congruence.*

A syntactic property on tiles guaranteeing the decomposition property (and hence that $\simeq_t$ is a congruence) is the so-called *basic source format*, which amounts to 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.

By inspection one can check that all the basic tiles satisfy the basic source property and thus the tile decomposition property holds. Thus thanks to Proposition A.2 tile bisimilarity is a congruence. □

Proof (of Lemma 2.2). Let us consider two connectors $\alpha\colon n \to h$ and $\beta\colon h \to m$. A cell $T(\alpha;\beta)[i,j]$ is marked iff there exists a concrete semantics with a tick on each element of interface in $d_{T(\alpha;\beta)}([i,j])$ and an untick on all the other elements of interface. In this case we can choose one of these concrete semantics (we may have many of them), which induces two denotational semantics for α and β which are compatible, i.e. the effect of α coincides with the trigger of β. Thus we have a k such that $T(\alpha)[i,k] = \checkmark$ and $T(\beta)[k,j] = \checkmark$. Thus $\bigvee_k (T(\alpha)[i,k] \wedge T(\beta)[k,j]) = \checkmark$. Thus for each $\checkmark$ in $T(\alpha;\beta)$ we have a $\checkmark$ in the product matrix. Viceversa if the cell $T(\alpha;\beta)[i,j]$ is not marked, there is no concrete semantics that agrees with that denotation. For each i,j,k we must have either $T(\alpha)[i,k] \neq \checkmark$ or $T(\beta)[k,j] \neq \checkmark$ and so we have $\bigvee_k (T(\alpha)[i,k] \wedge T(\beta)[k,j]) \neq \checkmark$ as desired.

As far as parallel composition is concerned, let us consider two connectors $\alpha\colon n \to h$ and $\beta\colon l \to m$. Refining the marked cells in $T(\beta)$ by $T(\alpha)$ and the unmarked cells by the empty table (with the same dimension of $T(\alpha)$) means that for each pair of cells in $T(\beta)$ and $T(\alpha)$ respectively, the cell in $T(\alpha \otimes \beta)$ with as domain the disjoint union of the domains is marked iff both the starting cells were marked. Let us prove that this is the correct denotational semantics for the composed connector. Given two marked cells in $T(\alpha)$ and $T(\beta)$ we can choose for the connectors two corresponding concrete semantics. The disjoint union of these two semantics is a correct concrete semantics for $\alpha \otimes \beta$, and thus the corresponding cell must be marked in $T(\alpha \otimes \beta)$. This cell has exactly the disjoint union of the two domains as domain. Thus the marked cells in the refined table are marked in $T(\alpha \otimes \beta)$. We have now to prove that if a cell is not marked in the refined table, then it is not marked in $T(\alpha \otimes \beta)$. Suppose by contradiction that the cell is marked in $T(\alpha \otimes \beta)$. Then we can choose a concrete semantics for the connector that corresponds to this denotational semantics, and its two projections on the subgraphs α and β are correct concrete semantics for them. Thus, since concrete semantics exist, in the denotational semantics $T(\alpha)$ and $T(\beta)$ the corresponding two cells are marked and so is the refined cell associated with them, which is absurd.

Let us finally consider dual connectors. The thesis can be verified by inspection for basic connectors. For the parallel composition of connectors note that the operations of refinement and transposition (of both the matrices) commute. Finally for sequential composition we have $T((\alpha;\beta)^c)[i,j] = T(\beta^c;\alpha^c)[i,j] = \bigvee_k (T(\beta^c)[i,k] \wedge T(\alpha^c)[k,j]) = \bigvee_k (T(\beta)[k,i] \wedge T(\alpha)[j,k]) = \bigvee_k (T(\alpha)[j,k] \wedge T(\beta)[k,i]) = T(\alpha;\beta)[j,i]$ as required. □

Proof (of Theorem 2.3). Since all connectors are stateless and each tile can be decomposed as vertical composition of horizontal composition of basic tiles, then two connectors are tile bisimilar iff their allowed combinations of tick and untick on the interfaces are equal.

We will prove the theorem by induction on the structure of the connector, showing that the cell (i,j) is marked in a table $T(\alpha)$ iff we have a tile $\alpha \xrightarrow[\mathsf{obs}(j)]{\mathsf{obs}(i)} \alpha$ where $\mathsf{obs}(i)$ (resp. $\mathsf{obs}(j)$) is a monoidal composition of tick and untick arrows, with a tick for each 1 and an untick for each 0 in i (resp. j), in the corresponding position.

For basic connectors the proof is trivially by inspection. For the inductive cases, let us consider for instance sequential composition (parallel composition is analogous). We have a marked cell in position (i,j) of the tick-table $T(\alpha;\beta)$ iff $\bigvee_k (T(\alpha)[i,k] \wedge T(\beta)[k,j]) = \checkmark$. Thus we have a k such that $T(\alpha)[i,k] = \checkmark$ and $T(\beta)[k,j] = \checkmark$. By inductive hypotheses we have two tiles $\alpha \xrightarrow[\mathsf{obs}(k)]{\mathsf{obs}(i)} \alpha$ and $\beta \xrightarrow[\mathsf{obs}(j)]{\mathsf{obs}(k)} \beta$ whose sequential composition yields the tile $\alpha;\beta \xrightarrow[\mathsf{obs}(j)]{\mathsf{obs}(i)} \alpha;\beta$ as required. Viceversa if the cell is unmarked then no such two tiles exist and we cannot derive the composed tile. Note in fact that, since our tile system enjoys the basic source property and thus the decomposition property, if a tile for $\alpha;\beta$ exists, then it can be derived by composing a tile for α and a tile for β. □

Proof (of Proposition 3.1). The first property is easily proved by noticing that for each basic connector α, $T(\alpha)[\vec{0},\vec{0}] = \checkmark$ and that the property is preserved by sequential and parallel composition.

As far as the other properties are concerned, notice that in a network of synch and hiding connectors, the only constraint is that connected nodes must have the same value. Thus we can divide the nodes in equivalence classes, with two nodes being in the same class iff they are connected. All the nodes in the same equivalence class must have the same value, while different equivalence classes are independent one from the other.

Thus the domain of each marked cell must correspond to the union of a set of equivalence classes. In particular each equivalence class is either completely contained in a domain, or it is disjoint from it.

Let us consider for instance the intersection of two domains (the proof is similar for other operations): each equivalence class is either contained in both the domains and thus in the intersection, or it is disjoint from at least one domain and thus from the intersection. Thus the intersection is the union of some set of equivalence classes. Since different equivalence classes are independent this is an allowed configuration and it corresponds to the domain of a marked cell. □

Proof (of Lemma 3.3). We proceed by contradiction. Suppose we have two different synch-tables of the same dimension and with the same base. Take a domain which corresponds to a marked cell in just one of the two tables. Each element of the base must either be a subset of it or must be disjoint from it. Furthermore the domain of the marked cell must be the union of the domains of the elements of the base that it contains, otherwise one can consider the difference (see Proposition 3.1) and get another minimal, which must be an element of the base, but which is not, raising an absurd. Since the base is the same for both the tables by hypothesis, then the domain corresponds to a marked cell in both the tables for the property of closure by domain union (see Proposition 3.1), that is absurd. □

Proof (of Proposition 3.5). The proof is by induction on the construction of the connector (see Lemma 1.2). The basic case is trivial. For the inductive case, we have to prove that given a connector $\alpha\colon n \to m$ in sorted form, we can transform in sorted form any connector of the form $id_{n_1} \otimes \sigma \otimes id_{n_2};\alpha$. There are a few cases to consider, according to the kind of σ.

Case ¡) We are already in sorted form.

Case γ) The connector can be moved to the layer α_γ by functoriality.

Case Δ) Either we have already a sorted form or Δ has some permutations of layer α_γ on the left and we can apply naturality and functoriality to reach a sorted form.

Case ∇) We have to consider different cases according to what is attached to the two elements of the right interface of ∇.

If there are just ! and ∇ connectors then the connector is already in sorted form.

If there is a Δ then we can apply the axiom $\nabla;\Delta = id$ if it is attached to both the elements of the interface and $\nabla \otimes id; id \otimes \Delta = \Delta;\nabla$ otherwise.

The case of permutations is the most difficult. In that case the connector is already in sorted form if only !, identities and other permutations follow. If the permutation is followed by ∇ or Δ then we have to apply some axioms. If it is followed by ∇ then we can use naturality to move the permutations from layer α_γ to β_γ and get back to the previous cases. Otherwise we can suppose that the permutation is attached to just one interface of each connector (if not, it could be simplified using commutativity). Thus we have a connector of the form $id \otimes \nabla; \gamma \otimes id$ and after some more permutations there is a Δ connector. Using naturality of permutations and functoriality we can always resort either to a sorted form (if there was no linear connection between ∇ and Δ) or to a redex for the axiom $\nabla \otimes id; id \otimes \Delta = \Delta;\nabla$. After having applied this axiom (one or more times) we can apply again naturality of permutations and functoriality to reach the sorted form.

Case !) The connector can be moved to the rightmost layer by functoriality since it has an interface of arity 0 on the right side. □

Proof (of Theorem 3.8). Thanks to Proposition 3.5 any connector $\alpha \in \mathbf{CC}(\nabla, \Delta, !, ¡)$ can be transformed in sorted form. Thus the first condition can be satisfied.

Suppose now that the second condition does not hold. This may happen only if we have a ¡ adjacent to a Δ, or a ! adjacent to a ∇ or either a ¡ or a ! adjacent to a permutation. In the first case, we just need to apply $¡ \otimes id; \Delta = id$, in the second case we can apply the dual axiom and in the last case we can apply the naturality of permutations. Note that these transformations only delete connectors, thus the resulting connector is still in sorted form.

As far as the third condition is concerned, note that the only possibility is to have subconnectors of the form ¡;!, which can be deleted by applying $¡;! = id_0$. □

Proof (of Theorem 3.9). Given a synch-table T let us consider its base $b(T)$ (which completely characterizes the table thanks to Lemma 3.3). We build a connector 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 of connectors Δ on the left of each central point P_b, where n is the number of elements in b that are in the left interface;
- we do a similar thing on the right, using ∇^m connectors on the right of P_b, where m is the number of elements in b that are in the right interface;
- we add permutations to connect the trees Δ^n and ∇^m of each central point P_b to the elements of the interfaces that correspond to elements of b.

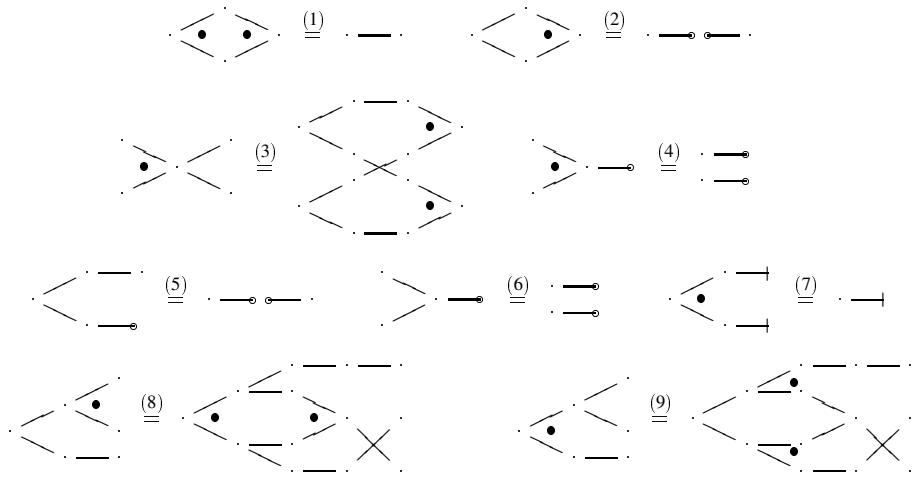


Fig. 12. Simple axioms for mutual exclusion, graphically.

We have to prove that $T(\alpha) = T$. Note that α has one connected component for each element in the base. Thus for each of them we have a concrete semantics where we have a tick on all the elements of the interfaces in its domain and untick on the other ones. This is a correct assignment, thus the corresponding cell is marked as required.

We now need to prove that these are the only elements in the base. Suppose by contradiction that we have another marked cell whose domain d is minimal. Let us choose a variable in d. Since each variable is linearly connected to at least a central point P, let us consider the domain m that corresponds to that central point. Since all the variables in m are connected, then they must all be in d. Thus $m \subseteq d$, that is absurd.

As far as uniqueness is concerned, note that the above construction is deterministic up to the axioms of symmetric monoidal categories and associativity and commutativity of synch connectors (in order to permute two central points we can use axiom $id_2 = \gamma;\gamma$, and then move the two permutations to layers α_γ and β_γ using naturality). □

Proof (of Theorem 3.10). In the proof of Theorem 3.9 we have shown how to construct the connector associated to a given synch-table and that the construction is deterministic up to the axioms of symmetric monoidal categories and associativity and commutativity of synch connectors, thus this is deterministic up to the axioms.

The construction defines a function from synch-tables to equivalence classes of connectors. Note that this function is injective since axioms are correct w.r.t. the denotational semantics. We have to prove that it is also surjective. Suppose by contradiction that the equivalence class of some connector α is not in the image of the function. Without loss of generality, we can assume that α is in normal form (thanks to Theorem 3.8). This connector has a denotational semantics $T(\alpha)$ that satisfies the properties of Proposition 3.1. The construction maps $T(\alpha)$ to α'. By construction (since the base is determined by the connections between central points and interfaces and viceversa) α

and α' are equal up to the axioms of symmetric monoidal categories and of associativity of synch connectors and thus are equivalent. This gives the absurd. □

Proof (of Lemma 3.11). The proof is by induction on n.

Case 1) Trivial, since $\Delta^1 = id$.
Case $n+1$)

$$\begin{aligned}
\nabla_{n+1};\Delta^{n+1}\otimes\Delta^{n+1} &= \nabla\otimes\nabla_n;id\otimes\gamma_{1,n}\otimes id_n;id\otimes\Delta^n\otimes id\otimes\Delta^n;\Delta\otimes\Delta\\
&= \nabla\otimes\nabla_n;id\otimes id\otimes\Delta^n\otimes id_n;id\otimes\gamma_{1,1}\otimes\Delta^n;\Delta\otimes\Delta\\
&= \nabla\otimes id_n;id_2\otimes\nabla_n;id_2\otimes\Delta^n\otimes\Delta^n;id\otimes\gamma_{1,1}\otimes id;\Delta\otimes\Delta\\
\{\text{ind. hyp.}\} &= \nabla\otimes id_n;id_2\otimes\Delta^n;id_2\otimes\nabla;\Delta_2\\
&= id\otimes\Delta^n;\nabla\otimes\nabla;\Delta_2\\
&= id\otimes\Delta^n;\Delta;\nabla\\
&= \Delta^{n+1};\nabla
\end{aligned}$$

where we have applied the inductive hypothesis as shown. □

Proof (of Lemma 3.12). The proof can be done by induction on the number of non identity basic connectors in (some representation of) α.

Basic step) We have to prove the thesis for each basic connector. Note that since $(\alpha^c)^c = \alpha$ then proofs that deal with dual connectors are essentially identical. Proofs for ! and ∇ (using the definition of Δ) are quite easy. We will present in detail the proof for γ, the one for ∇ being similar. In the proof we exploit the fact that $!_2 = \gamma;!_2$ by naturality.

$$\begin{aligned}
\gamma\otimes id_2;\Delta_2;!_2 &= \gamma\otimes id_2;id\otimes\gamma\otimes id;\Delta\otimes\Delta;\gamma;!_2\\
&= \gamma\otimes id_2;id\otimes\gamma\otimes id;\gamma_{2,2};\Delta\otimes\Delta;!_2\\
&= \gamma\otimes id_2;id\otimes\gamma\otimes id;id\otimes\gamma\otimes id;\gamma\otimes\gamma;id\otimes\gamma\otimes id;\Delta\otimes\Delta;!_2\\
&= \gamma\otimes id_2;\gamma\otimes\gamma;\Delta_2;!_2\\
&= id_2\otimes\gamma;\Delta_2;!_2
\end{aligned}$$

Inductive step) Any connector $\alpha\colon m\to n$ built using k non identity basic connectors can be written using functoriality as the composition $\alpha_1;\alpha_2$ of two connectors $\alpha_1\colon m\to p$ and $\alpha_2\colon p\to n$, each of which contains strictly less than k non identity basic connectors. Thus we have:

$$\begin{aligned}
\alpha\otimes id_n;\Delta_n;!_n &= \alpha_1\otimes id_n;\alpha_2\otimes id_n;\Delta_n;!_n\\
\{\text{ind. hyp.}\} &= \alpha_1\otimes id_n;id_p\otimes\alpha_2^c;\Delta_p;!_p\\
&= id_m\otimes\alpha_2^c;\alpha_1\otimes id_n;\Delta_p;!_p\\
\{\text{ind. hyp.}\} &= id_m\otimes\alpha_2^c;id_m\otimes\alpha_1^c;\Delta_m;!_m\\
&= id_m\otimes\alpha^c;\Delta_m;!_m
\end{aligned}$$

□

Proof (of Proposition 3.14). This is the most complex proof in the paper, and it extends the proof of Proposition 3.5 to deal with all the connectors in $\mathbf{CC}(\nabla, \Delta, !, \mathsf{i}, \overline{\nabla})$. We have to show that basic connectors that are in the wrong layer, can be made to commute with adjacent layers until they reach the correct layer. In general, during the operation, other basic connectors may be created, and in this case we have to check that the procedure indeed terminates.

We start by presenting some commutation properties that we will use later:

1. γ can be commuted thanks to naturality when traversing a layer from the side where connectors have arity 1 to the side where connectors have arity 2. For instance, to traverse layer $\alpha_{\overline{\nabla}}$ from left to right we can use $\gamma; \nabla \otimes id = id \otimes \nabla; \gamma_{1,2}$. The procedure can be iterated until the whole layer is traversed. Note that this is enough to move γ connectors from any place to either layer α_γ or β_γ, exactly as required.
2. $\underline{\Delta}$ connectors can be commuted with ∇ connectors using axiom $\underline{\Delta}; \nabla = \nabla \otimes \nabla; id \otimes \gamma \otimes id; \underline{\Delta} \otimes \underline{\Delta}$. This operation creates some γ connectors which can be moved to the correct layer for what already proved.
3. Dually, Δ can be commuted with $\overline{\nabla}$.
4. $\underline{\Delta}$ connectors can be commuted with ∇ connectors using axiom 10 in Figure 10. This creates γ, ∇ and ! connectors. When applying this property we must check that we are able to move the created connectors to their correct layers without cycling.
5. ∇ can be commuted with $\overline{\nabla}$ using axiom $\nabla; \overline{\nabla} \otimes id = \nabla; \nabla \otimes \nabla; id \otimes \underline{\Delta}\ id; id \otimes \gamma$. This will create γ connectors and $\underline{\Delta}$ connectors. In this case too we have to check that there is no risk to cycle.
6. Dually, $\underline{\Delta}$ can be commuted with Δ. This will create γ connectors and $\overline{\nabla}$ connectors. Here a check for avoiding cycles is required too.
7. We can commute any combination of ∇, γ and Δ thanks to Theorem 3.8.

We will now prove the thesis by induction on the construction of the connector (see Lemma 1.2). The basic case is trivial. For the inductive case we have to prove that given any basic connector σ and any connector $\alpha\colon m \to n$ in sorted form, we can transform $id_{n_1} \otimes \sigma \otimes id_{n_2}; \alpha$ in sorted form (whenever the sequential composition is defined). In other words, we must show that we can commute σ with the layers of α until we reach the sorted form. We consider different cases, according to which basic connector σ is.

Note also that we can always move !, i, $\mathbf{0}$ and $\overline{\mathbf{0}}$ connectors to layers $\beta_!$, α_{i}, $\beta_{\mathbf{0}}$ and $\alpha_{\overline{\mathbf{0}}}$ as required using the properties of identity and functoriality, thanks to the fact that one of their interfaces has arity 0. Thus we can just focus on the restructuring of connectors composed by γ, ∇, Δ, $\overline{\nabla}$ and $\underline{\Delta}$.

We can now consider the different cases:

Case $\overline{\nabla}$) Trivial.

Case γ) Property 1 is enough to move γ through the layer $\alpha_{\overline{\nabla}}$ if needed.

Case Δ) The Δ connectors need just to be commuted with the $\overline{\nabla}$ and the γ connectors. The first part can be done thanks to property 3 and the second one thanks to property 1 (moving γ towards the left).

Case ∇ and $\underline{\Delta}$) We prove these two cases in one single step since commuting ∇ connectors produces also $\underline{\Delta}$ connectors and viceversa. This is the most complex part of the proof. The ∇ connectors need to be commuted with: (ia) $\overline{\nabla}$, (iia) γ, and (iiia) Δ. The $\underline{\Delta}$ connectors need to be commuted with: (ib) $\overline{\nabla}$, (iib) γ, (iiib) Δ, and (ivb) ∇.

- Part (ia) and (ib) are by induction on the width of layer α_{∇}, that is on the maximum number of ∇ connectors in all the paths of the layer. For ∇ connectors, the commutation can be done thanks to property 5, but each time that we apply the axiom involved, we create one Δ and one γ connector. The γ connectors can be moved to layer α_{γ} using property 1. As far as Δ connectors are concerned, the step (ib) can be done using property 4, which creates additional γ, ∇ and ! connectors. As usual ! connectors can always be moved to the layer $\beta_{!}$ and symmetries can be moved to either layer α_{γ} or β_{γ} using property 1. As far as the ∇ connector is concerned, note that all the ∇ connectors created have to commute with a layer whose width is one less than the width of the starting layer (note in fact that none of the axioms used increases the width of the layer, and the first axiom applied decreases it by one), thus this can be done thanks to the inductive hypothesis.
- Steps (iia) and (iib) can both be done thanks to property 1.
- The ∇ connectors can commute with layer α_{Δ} using property 7 and thus reaching the layer β_{∇} as required, what proves part (iiia).
- As far as Δ connectors are concerned, we can use property 6 to commute them with layer α_{Δ}. This will produce ∇ connectors that need to be moved back to layer α_{∇}. This can be done easily using properties 1 and 3. The problem is that these new ∇ connectors may have to be traversed again by the ∇ connectors that created them, and in this case we have to follow a different strategy. Note that when applying axiom 8 ($\nabla;\nabla \otimes id = \nabla;\nabla \otimes \nabla; id \otimes \Delta \otimes id; id \otimes \gamma$) the Δ connector is created attached to an element of interface which is not the same to which the ∇ connector in the left hand side was attached to, what means that we risk to cycle 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. If they join on a Δ connector then we can use associativity, commutativity and naturality to individuate a connector of the form $\nabla;\nabla \otimes \nabla; id \otimes \Delta \otimes id$ which is equal up to the axioms to $\nabla;\nabla \otimes id; id \otimes \gamma$. Thus we can use this equality to commute ∇ and ∇ without cycling. If they join on a ∇ connector instead we can individuate a connector $\nabla;\Delta$ which is equal up to the axioms to $\mathbf{0};\overline{\mathbf{0}}$. Thus both the paths are removed and we do not cycle. This concludes part (iiib).
- To finish we have to commute the Δ connectors with the layer β_{∇} (ivb). This can be done using property 2. □

Proof (of Theorem 3.16). Thanks to Proposition 3.14 we can suppose that the connector is already in sorted form. We have to satisfy all the conditions for a connector to be in normal form.

For condition 1 note that $\mathbf{0}$ (resp. $\overline{\mathbf{0}}$) connectors absorb each connector they are attached to, unless this is a ∇ (resp. Δ) connector (from the side with the interface of arity 2) and in that case they are absorbed by it. Thus one can use axioms for $\mathbf{0}$ and $\overline{\mathbf{0}}$ to make them adjacent to free variables, as required by the new order of connectors.

Suppose condition 2 is not satisfied, that is we have an hiding connector whose interface if neither the root of a mutual exclusion tree nor a central point. The only remaining possibility is that it is adjacent to a leaf of a synch connector or to a permu-

tation, but in the first case we can use axiom $\nabla;! \otimes id = id$ (or the dual one) to delete it, while in the second case we can use naturality to delete the γ itself ($\gamma;! \otimes id = id \otimes !$).

As far as condition 3 is concerned, note that if a central point is linearly connected using two different paths to the same variable, then we can use associativity and commutativity to individuate a $\nabla;\Delta$ or a $\nabla;\Delta$ connector. Thus we can apply the axiom $\nabla;\Delta = \mathbf{0};\overline{\mathbf{0}}$ (or the dual one) to remove the paths and then propagate the inaction connectors. This will satisfy the condition.

In order to satisfy condition 4 we must prove that we can merge any two central points linearly connected to the same set of variables. We can use Lemma 3.11 on the left of the central point and the dual property (which holds, as can be proved using the dual axioms) on the right. In order to do that we must individuate two redexes in the graph where the properties can be applied. This can be done using associativity, commutativity and naturality. We can then apply the lemma (and its dual) and then delete the $\nabla;\Delta$ in the centre using $\nabla;\Delta = id$. Note that the result is equal to the starting connector, but with one central point less as required.

Suppose condition 5 is not satisfied. If the central point has a $¡$ connector as left (trivial) tree then we can apply directly the axiom schema and delete the central point. Otherwise we can use Lemma 3.12 which will create exactly the previous situation, where we can apply the axiom schema as before. Note that the created ! connectors can be simplified with the ∇ connectors they are attached to.

Suppose condition 6 is not satisfied. Then we can apply axiom 11 in Figure 10, which allows to transform a mutual exclusion on a set of connectors into a single mutual exclusion between each pair of points, and solve the problem.

Suppose condition 7 is not satisfied. Then we can use associativity and commutativity to individuate a connector of the form $¡_2;\nabla \otimes \nabla;\Delta_2$ which can be transformed using the axioms into $¡_2;\Delta;\nabla = ¡;\nabla$.

Suppose that condition 8 is not satisfied, that is we have two central points linearly connected to disjoint sets of free variables, and no hidden variable linearly connected to both of them. Then we can use axiom 13 in Figure 10 to add the hidden variable, together with a new central point for the synchronized execution of the two actions. If this one existed already then we can first delete it using axiom 12 in Figure 10.

Finally suppose that condition 9 is not satisfied, that is we have an hidden variable on the right of a central point, which is not adjacent to it. For what already proved it must be adjacent to a mutual exclusion tree, which is linearly connected to exactly two central points. Thus using associativity and commutativity we can individuate a connector of the form $\nabla \otimes \nabla;id \otimes (\Delta;!) \otimes id$ which can be transformed using the axioms into $id \otimes (¡;\nabla) \otimes id;\Delta \otimes \Delta$ using the definition of Δ and then simplifying the resulting connector. □

Proof (of Theorem 3.17). Given a tick-table T, the connector α is realized in the following way:

- if we have an input (resp. output) variable that has value 0 in all the concrete semantics, then we connect that variable to a $\mathbf{0}$ (resp. $\overline{\mathbf{0}}$) connector;
- for other input (resp. output) variables, we count the number of checked cells that have that variable in the domain and we build a tree of ∇ (resp. Δ) connectors with that number of leaves;

- for each pair of central points with disjoint sets of free variables, we create an hidden variable for them on the left (this variable will be associated to both the central points);
- for each checked cell in the table, we create a central point in the connector, with two outgoing trees (possibly trivial) of synch connectors, a tree Δ^n to the left and a tree ∇^m to the right. Here n is the number of input variables in the domain of the cell, plus the number of hidden variables that are associated to the central point while m is the number of output variables in the domain;
- we connect leaves of synchronization trees with leaves of mutual exclusion trees using permutations, connecting each central point to the associated variables.

It remains to be proved that $T(\alpha) = T$. Note that if a variable has value 0 in all the concrete semantics, then the variable is adjacent to an inaction connector and its value is forced to zero. Note also that these are the only inaction connectors in α. From now on we will not consider these variables. We define a node active in a concrete semantics iff a tick action is performed on it. We will prove that: (i) in each concrete semantics at most a central point can be active, (ii) that the choice of the central point determines the variables in the domain, and (iii) that the corresponding cell in the tick-table must be marked.

For part (i), note that each pair of central points has a common variable (either a free one or an hidden one). Thus if two central points are active we have two active leaves in the corresponding mutual exclusion tree, which is absurd.

As far as part (ii) is concerned, note that the variables in the domain are exactly the ones which are linearly connected to the central point. These variables must be in the domain since one leaf of their mutual exclusion tree is connected to the active central point using only synch connectors and symmetries. Suppose now by contradiction that another variable is in the domain. This variable is linearly connected to at least a central point. Since the variable is in the domain a leaf of its mutual exclusion tree is active, and thus the central point attached to it. This raises the absurd.

Part (iii) follows since by construction we have a central point for each marked cell, and this central point is linearly connected exactly to the variables in the domain.

As far as uniqueness is concerned, note that the above construction is deterministic up to the axioms of symmetric monoidal categories and associativity and commutativity of mutual exclusion and synch connectors. □

Proof (of Theorem 3.18). In the proof of Theorem 3.17 we have shown how to construct the connector associated to a given tick-table that satisfies our assumptions and that the construction is deterministic up to the axioms.

The construction defines a function from tick-tables to equivalence classes of connectors. Note that this function is injective since axioms are correct w.r.t. the denotational semantics. We have to prove that it is also surjective. Suppose by contradiction that the equivalence class of some connector α is not in the image of the function. Without loss of generality, we can assume that α is in normal form (thanks to Theorem 3.16). This connector has a denotational semantics $T(\alpha)$ with the cell with empty domain marked. By construction (since the marked cells of the tick-table are determined by the connections between central points and interfaces and viceversa) α and α' are

equal up to the axioms of symmetric monoidal categories and of associativity of synch and choice connectors and thus are equivalent. This gives the absurd. □