The Tile Model*

Fabio Gadducci and Ugo Montanari

Università di Pisa, Dipartimento di Informatica, Corso Italia 40, 56125 Pisa, Italy
({gadducci,ugo}@di.unipi.it)

1 Introduction

It is not an overstatement to say that, in the latest years, there has been an unprecedented flow of proposals, aiming at methodologies that could offer a flexible framework (intended as a meta-formalism) for specifying the behaviour of rule-based computational systems. Widely spread in the field of concurrency theory, *transition systems* [24] offers a useful tool: They are roughly defined as a set of *states*, representing the possible *configurations* (e.g., memory contents, data structures, etc.) of an abstract machine; and a *transition relation* over states, where each element $\langle s, t \rangle$ denotes the evolution from state s to state t. Due to its simplicity, however, this view is clearly no more adequate when we consider formalisms with a compositional structure over states, and the transition relation needs to be inductively defined according to that structure. This is the case for example of *Petri nets* [42], where a state is a multi-set of atomic components, and disjoint subsets may evolve simultaneously (i.e., *in parallel*); of *term rewriting systems* [26], where states are terms of an algebra, and rewriting steps are obtained by closure under substitution and contextualisation from a set of rewrite rules; or, in general, of those formalisms that rely on the use of *synchronisation* and *side-effects* in determining the actual behaviour of a given system.

We consider as a major breakthrough the introduction of the *structural operational semantics* (SOS) approach [40]: States are terms of an algebra, whose operators express basic features of a system, and the transition relation is defined by means of inference rules, guided by the structure of the states. Some of the extensions to this approach that proved fruitful for our view are *context systems* [30], where the transition relation is defined on *open* terms (that is, terms that may contain free variables), describing partially unspecified component of a system; and *structured transition systems* [9, 15], where, in order to account for the spatial distribution of a system, transitions are equipped with an algebraic structure, too. The first approach generalises observational semantics to partially specified behaviours, while the latter characterises equationally those (sequences of) transitions that identify "computationally equivalent" behaviours.

To equip transitions (actually, rewriting steps) with an algebraic structure is also a key point of the *rewriting logic* approach [33]: A system is considered

* Research partially supported by MURST Project *Tecniche Formali per Sistemi Software*, by CNR Integrated Project *Metodi per Sistemi Connessi mediante Reti*, and by ESPRIT Working Group *CONFER2*.

as a logical theory, and any sequence of transitions as a *sequent* entailed by the theory. The entailment relation is defined by means of a suitable calculus, whose inference rules are driven by the structure of terms, and are intended to express some basic feature of the system. Computing is then identified with deduction, and equivalent computations correspond to proofs with the same structure.

Trying to summarise, we could say that our *tile model* combines the SOS idea of a structure-driven set of inference rules with the use of an incremental format (analogous to context systems, but more general) that allows us to build new rules from old ones. Furthermore, from structured transition systems and rewriting logic the framework retains the idea of an explicit representation of transitions as an algebra, where certain basic structural axioms describe the concurrency properties of the model.

In our logic, a sequent is then a tuple $\alpha : s \xrightarrow[b]{a} t$, where $s \to t$ is a rewriting step from term s to t, α is the *proof term* associated to the step, a is the *trigger* of the step, and b is its *effect*. Graphically, this is represented as a *tile*

$$\begin{tikzcd} n \arrow[r, "s"] \arrow[d, "a"'] \arrow[dr, phantom, "\alpha"] & m \arrow[d, "b"] \\ p \arrow[r, "t"'] & q \end{tikzcd}$$

stating that the *initial configuration* s of the system evolves to the *final configuration* t producing an effect b. However, s may be an open term, and the rewriting step is possible only if the sub-components of s may evolve producing a cumulative effect a, that serves as the trigger for α. Both trigger and effect are called *observations*, and model the interaction, during a computation, of the system with its environment. More precisely, both system configurations are equipped with an *input interface* and an *output interface*, and the trigger just describes the evolution of the input interface from its initial to its final configuration. Similarly for the effect. So, it is natural to visualise a tile as a two-dimensional structure, where the horizontal dimension corresponds to the extension of the system, while the vertical dimension corresponds to the extension of the computation. Actually, we should also imagine a third dimension (the *thickness* of the tile), which models parallelism: Configurations, observations, interfaces and tiles themselves are all supposed to consist of several components in parallel.

This spatial structure is reflected by the inference rules of the calculus, in particular, by the three composition rules. Two sequents α, β can be composed in parallel $(\alpha \otimes \beta)$, coordinated $(\alpha * \beta)$ or composed sequentially $(\alpha \cdot \beta)$, varying accordingly the corresponding source, target, trigger and effect. The first operator allows for different components of a system to act simultaneously, explicitly describing parallelism by a *monoidal* structure over transitions. Together with the second, they offer the possibility for different sub-components to synchronise, according to the information carried by their effects, and the presence of an eventual coordinator with an adequate trigger. The third expresses the execution of a sequence of transitions. Proof terms allow us to equip each rewriting step with a suitable encoding of its causes, and a suitable equivalence relation over proof terms will then provide a concurrent semantics for the systems that are under examination.

Space limitations prevent us from dealing with the full range of applications of the framework, or to give a full account of the expressive power that the use of a three-dimensional structure can give in capturing also higher-order formalisms, despite the simple term structure we chose for configurations. In fact, tiles have been used for coordination formalisms equipped with flexible synchronisation primitives [6, 37] and for calculi for mobile processes, like the asynchronous π-calculus [16]. Tiles are also convenient for handling concurrent process calculi [17]. These results are then strengthened by the tight relationship between the categorical models of sharing graphs structures (as introduced in [18]) and gs-monoidal theories (see Section 2), a framework used to accommodate the behaviour of e.g. Milner's *action calculus* [36].

As far as categorical models are concerned, *2-categories* [25] are the best-known cat-enriched structure in computer science: They are categories such that each hom-set (the set of arrows with same source and target) is the object-set of another category, whose arrows (called *cells*) are closed under certain composition operators, and are subject to suitable coherence axioms. A more refined enrichment is present in *double categories* [3], whose structure can be informally described as the superposition of a vertical and a horizontal category of cells. It is well-known that 2-categories represent a faithful model for term rewriting systems [8, 41, 43]: The arrows of the underlying category denote terms, and cells denote (equivalence classes of) rewrites. In the paper we show that a similar adequacy result holds for our tile model, if we consider double categories instead. Thus, the generality of our tile model is further confirmed by the richer structure of double categories with respect to 2-categories.

The paper has the following structure. In Section 2 we generalise the usual notion of term over a signature, providing three concrete descriptions which formalise the assumptions implicitly made in the ordinary term algebra construction. In Section 3 we introduce our rewriting systems, providing them with a suitable logic that describes the classes of derivations entailed by a system. In Section 4 we deal with our case study, showing how the standard observational semantics of many process algebras (and context systems) can be recovered in our framework. Finally, in Section 5 we provide a sound and complete categorical model for our logic by means of double categories.

2 Building States

We open this section recalling some basic definitions from graph theory, used to recast in a more general fashion the usual notion of term over a signature.

Definition 1 (graphs). *A* graph G *is a 4-tuple* $\langle O_G, A_G, \delta_0, \delta_1 \rangle$*:* O_G*,* A_G *are sets whose elements are called respectively* objects *and* arrows*, and* $\delta_0, \delta_1 : A_G \to O_G$ *are functions, called respectively* source *and* target*. A graph* G *is* reflexive *if there exists an* identity *function* $id : O_G \to A_G$ *such that* $\delta_0(id(a)) = \delta_1(id(a)) = a$ *for all* $a \in O_G$*; it is* with pairing *if its class of objects forms as monoid; it is* monoidal *if reflexive and both its classes of objects and arrows form a monoid, and the functions preserve the monoidal operator and the neutral element.* $\square$

We can think of an ordinary, one-sorted signature Σ of operators as a graph with pairing: Its nodes are elements of the monoid $\mathbb{N}_c$ of (underlined) natural numbers, where sum is defined as $\underline{n} \otimes \underline{m} = \underline{n+m}$, and $\underline{0}$ is the neutral element; its arcs are operators, such that $f : \underline{n} \to \underline{1}$ iff $f \in \Sigma_n$. This view allows for an inductive, step-by-step account of the usual algebraic notion of term, by means of a chain of structures of increasing complexity.

Definition 2 (graph theories). *Given a signature* Σ*, the associated* graph theory $G(\Sigma)$ *is the monoidal graph with objects the elements of the monoid* $\mathbb{N}_c$*, and arrows those generated by the following inference rules*

$$\text{(generators)}\ \frac{f \in \Sigma_n}{f : \underline{n} \to \underline{1} \in G(\Sigma)} \qquad \text{(pairing)}\ \frac{s : \underline{n} \to \underline{m},\, t : \underline{n'} \to \underline{m'}}{s \otimes t : \underline{n} \otimes \underline{n'} \to \underline{m} \otimes \underline{m'}}$$

$$\text{(identities)}\ \frac{\underline{n} \in \mathbb{N}_c}{id_{\underline{n}} : \underline{n} \to \underline{n}}$$

Monoidality implies that $id_{\underline{0}}$ *is the neutral object of the monoid of arrows, and that the* monoidality *axiom* $id_{\underline{n} \otimes \underline{m}} = id_{\underline{n}} \otimes id_{\underline{m}}$ *holds for all* $\underline{n}, \underline{m} \in \underline{\mathbb{N}}$. $\square$

Graph theories simply equip a signature with an explicit notion of pairing: Each arrow then denotes an array of operators and identities. This theory is rather expressiveless, *per se*: We will use it only in conjunction with the rewriting mechanism to be introduced later on.

Definition 3 (monoidal theories). *Given a signature* Σ*, the associated* monoidal theory $\mathbf{M}(\Sigma)$ *is the monoidal graph with objects the elements of the monoid* $\mathbb{N}_c$*, and arrows those generated by the following inference rules*

$$\text{(generators)}\ \frac{f \in \Sigma_n}{f : \underline{n} \to \underline{1} \in \mathbf{M}(\Sigma)} \qquad \text{(pairing)}\ \frac{s : \underline{n} \to \underline{m},\, t : \underline{n'} \to \underline{m'}}{s \otimes t : \underline{n} \otimes \underline{n'} \to \underline{m} \otimes \underline{m'}}$$

$$\text{(identities)}\ \frac{\underline{n} \in \mathbb{N}_c}{id_{\underline{n}} : \underline{n} \to \underline{n}} \qquad \text{(composition)}\ \frac{s : \underline{n} \to \underline{m},\, t : \underline{m} \to \underline{k}}{s;t : \underline{n} \to \underline{k}}$$

Moreover, the composition operator is associative, and the monoid of arrows satisfies the functoriality *axiom*

$$(s \otimes t);(s' \otimes t') = (s;s') \otimes (t;t')$$

(whenever both sides are defined) and the identity *axiom* $id_{\underline{n}}; s = s = s; id_{\underline{m}}$ *for all* $s : \underline{n} \to \underline{m}$. $\square$

Monoidal theories add composition: In fact, any arrow in $\mathbf{M}(\Sigma)$ can be written in a normal form as a sequence of concrete arrows already "appearing" in the underlying graph theory. Thanks to the functoriality axiom, it is easy to show that each term can be decomposed in a tree-like fashion: A monoidal theory is just an example of a so-called *strict monoidal category* [32], for which many representation results are well-known. Now we introduce the more expressive kind of theories we deal with in our paper, *gs-monoidal theories*.

Definition 4 (gs-monoidal theories). *Given a signature Σ, the associated* gs-monoidal theory $\mathbf{GS}(\Sigma)$ *is the monoidal graph with objects the elements of the monoid $\mathbb{N}_c$, and arrows those generated by the following inference rules*

$$\text{(generators)}\ \frac{f \in \Sigma_n}{f : \underline{n} \to \underline{1} \in \mathbf{GS}(\Sigma)} \qquad \text{(pairing)}\ \frac{s : \underline{n} \to \underline{m}, t : \underline{n'} \to \underline{m'}}{s \otimes t : \underline{n} \otimes \underline{n'} \to \underline{m} \otimes \underline{m'}}$$

$$\text{(identities)}\ \frac{\underline{n} \in \mathbb{N}_c}{id_{\underline{n}} : \underline{n} \to \underline{n}} \qquad \text{(composition)}\ \frac{s : \underline{n} \to \underline{m}, t : \underline{m} \to \underline{k}}{s; t : \underline{n} \to \underline{k}}$$

$$\text{(duplicators)}\ \frac{\underline{n} \in \mathbb{N}_c}{\nabla_{\underline{n}} : \underline{n} \to \underline{n} \otimes \underline{n}} \qquad \text{(dischargers)}\ \frac{\underline{n} \in \mathbb{N}_c}{!_{\underline{n}} : \underline{n} \to \underline{0}}$$

$$\text{(permutation)}\ \frac{\underline{n}, \underline{m} \in \mathbb{N}_c}{\rho_{\underline{n},\underline{m}} : \underline{n} \otimes \underline{m} \to \underline{m} \otimes \underline{n}}$$

Moreover, the composition operator is associative, and the monoid of arrows satisfies the functoriality axiom

$$(s \otimes t); (s' \otimes t') = (s; s') \otimes (t; t')$$

(whenever both sides are defined); the identity axiom $id_{\underline{n}}; s = s = s; id_{\underline{m}}$ for all $s : \underline{n} \to \underline{m}$; the monoidality *axioms*

$$\rho_{\underline{n} \otimes \underline{m}, \underline{p}} = (id_{\underline{n}} \otimes \rho_{\underline{m},\underline{p}}); (\rho_{\underline{n},\underline{p}} \otimes id_{\underline{m}})$$

$$!_{\underline{n} \otimes \underline{m}} = !_{\underline{n}} \otimes !_{\underline{m}} \qquad \nabla_{\underline{n} \otimes \underline{m}} = (\nabla_{\underline{n}} \otimes \nabla_{\underline{m}}); (id_{\underline{n}} \otimes \rho_{\underline{n},\underline{m}} \otimes id_{\underline{n}})$$

$$!_{\underline{0}} = \nabla_{\underline{0}} = \rho_{\underline{0},\underline{0}} = id_{\underline{0}}$$

for all $\underline{n}, \underline{m}, \underline{p} \in \mathbb{N}_c$; the coherence *axioms*

$$\nabla_{\underline{n}}; (id_{\underline{n}} \otimes \nabla_{\underline{n}}) = \nabla_{\underline{n}}; (\nabla_{\underline{n}} \otimes id_{\underline{n}}) \qquad \nabla_{\underline{n}}; \rho_{\underline{n},\underline{n}} = \nabla_{\underline{n}}$$

$$\nabla_{\underline{n}}; (id_{\underline{n}} \otimes !_{\underline{n}}) = id_{\underline{n}} \qquad \rho_{\underline{n},\underline{m}}; \rho_{\underline{m},\underline{n}} = id_{\underline{n}} \otimes id_{\underline{m}}$$

for all $\underline{n}, \underline{m} \in \mathbb{N}_c$; and the naturality *axiom*

$$(s \otimes t); \rho_{\underline{m},\underline{q}} = \rho_{\underline{n},\underline{p}}; (t \otimes s)$$

for all $s : \underline{n} \to \underline{m}, t : \underline{p} \to \underline{q} \in S$. $\square$

Intuitively, a gs-monoidal theory is a symmetric strict monoidal category enriched with additional structure, namely the operators ∇ and !, allowing for a (controlled) form of *duplication* and *discharge* of data. This structure falls short of the usual definition of cartesian category only for two axioms, imposing the satisfaction of naturality also for these operators.

Definition 5 (algebraic theories). *Given a signature Σ, the associated* algebraic theory $\mathbf{A}(\Sigma)$ *is the monoidal graph with objects the elements of the monoid $\mathbb{N}_c$, and arrows those generated by the same inference rules given for gs-monoidal theories. Moreover, besides the set of axioms already valid for these theories, also a naturality axiom for both dischargers and duplicators holds, namely*

$$s; !_{\underline{m}} = !_{\underline{n}} \qquad s; \nabla_{\underline{m}} = \nabla_{\underline{n}}; (s \otimes s)$$

for all $s : \underline{n} \to \underline{m} \in S$. □

It can be considered categorical folklore that a cartesian category can be decomposed into a symmetric monoidal category, together with a family of suitable natural transformations, usually denoted as *diagonals* and *projections* (related papers range from [20, 39] to the more recent [22, 28]). Then, our notion of algebraic theory can be proved equivalent to the classical definition, dating back to the early work of Lawvere [27, 31]: Hence, a classical result states the equivalence of these theories with the usual term algebras.

Proposition 1 (algebraic theories and term algebras). *Let Σ be a signature. Then for all $\underline{n}, \underline{m} \in \mathbb{N}_c$ there exists a one-to-one correspondence between the set of arrows from $\underline{n}$ to $\underline{m}$ of $\mathbf{A}(\Sigma)$ and the m-tuples of elements of the term algebra —over a set of n variables— associated to Σ.* □

In other words, each arrow $\underline{n} \to \underline{1}$ uniquely identifies an element t of the term algebra over the set $\{x_1, \ldots, x_n\}$: An arrow $\underline{n} \to \underline{m}$ is an m-tuple of such elements, and arrow composition is term substitution. Note that this correspondence *requires* that both ∇ and ! are natural: That is, gs-monoidal theories are too concrete, distinguishing elements that intuitively represent the same term. A fundamental property of correspondence can be shown between gs-monoidal theories and *term graphs* (roughly, graphs whose nodes are labeled by operators, as defined e.g. in the introductory chapter of [44]): Each arrow $\underline{n} \to \underline{m}$ identifies a term graph t over Σ with a specified m-tuple of roots and a specified n-tuple of variables nodes, and arrow composition is graph replacement [7]. This correspondence motivates the acronym, where *gs* stands for *graph substitution*.

Example 1 (terms and theories). Let us consider the signature $\Sigma_\epsilon = \bigcup_{i=0}^{2} \Sigma_i$, where $\Sigma_0 = \{a, b\}$, $\Sigma_1 = \{f, g\}$ and $\Sigma_2 = \{h\}$ (that same signature is often used in the next sections). Some of the elements in $\mathbf{GS}(\Sigma_\epsilon)$ are $a; f : \underline{0} \to \underline{1}$, $f; g : \underline{1} \to \underline{1}$ and $a; \nabla_{\underline{1}}; (f \otimes id_{\underline{1}}); h : \underline{0} \to \underline{1}$: They correspond to terms $f(a), g(f(x))$ and $h(f(a), a)$, respectively, for a given variable x. Let us consider instead the elements $a; \nabla_{\underline{1}}; h$ and $(a \otimes a); h$: They correspond to the same term $h(a, a)$, but they are different as elements of $\mathbf{GS}(\Sigma_\epsilon)$, while they are identified by the naturality axiom for ∇ in $\mathbf{A}(\Sigma_\epsilon)$. □

We believe that the incremental description of these theories, and in particular the relevance of the computational interpretation of (the different presentations of) gs-monoidal theories, has not received enough attention in the literature, despite a few notable exceptions (see e.g. [10, 23, 28]). In fact, the main point for our discussion is that, although their definition is more involved than the classical, set-theoretical ones, algebraic (and *a fortiori* gs-monoidal) theories allow a description of terms which is far more general, and at the same time more concrete, than the one allowed by the ordinary descriptions as elements of a term algebra, separating in a better way the "Σ-structure" from the additional algebraic structure that the *meta-operators* used in the set-theoretical presentation of term algebras (like *substitution*) implicitly enjoy.[1] In particular, they will allow for an easy graphical depiction of our *rewriting systems*.

3 Describing Systems

In this section we propose a general framework for describing the behaviour of rule-based systems, in the vein of both the *rewriting logic* formalism of Meseguer [33] and the SOS approach of Plotkin [40]. Intuitively, what we are seeking is a suitable definition of *rewriting system*: Each rule should be considered as a *module* (kind of basic component of a system) carrying information (equivalently, expressing a few restrictions) on the possible behaviour of its sub-components (that is, of the terms to which it can be applied).

Definition 6 (algebraic rewriting systems). *An* algebraic rewriting system *(*ARS*) $\mathcal{R}$ is a four-tuple $\langle \Sigma_\sigma, \Sigma_\tau, N, R\rangle$, where $\Sigma_\sigma, \Sigma_\tau$ are signatures, N is a set of (rule) names and R is a function $R : N \to \mathbf{A}(\Sigma_\sigma) \times G(\Sigma_\tau) \times G(\Sigma_\tau) \times \mathbf{A}(\Sigma_\sigma)$ such that for all $d \in N$, with $R(d) = \langle l, a, b, r\rangle$, we have $l : \underline{n} \to \underline{m}, r : \underline{p} \to \underline{q}$ iff $a : \underline{n} \to \underline{p}, b : \underline{m} \to \underline{q}$.* $\square$

With an abuse of notation, in the previous definition we denote the set of arrows of a theory by the theory itself. In the following, we usually write a *rule* as $d : l \xrightarrow[b]{a} r$ or, graphically, as a *tile*

$$\begin{tikzcd} n \arrow[r, "l"] \arrow[d, "a"'] \arrow[dr, phantom, "d"] & m \arrow[d, "b"] \\ p \arrow[r, "r"'] & q \end{tikzcd}$$

making explicit source and target of the operators.

A rewriting system is *gs-monoidal* (*monoidal*, *graph*) if, in defining R, l and r are elements of $\mathbf{GS}(\Sigma_\sigma)$ ($\mathbf{M}(\Sigma_\sigma)$ and $G(\Sigma_\sigma)$, respectively): The choice depends on *how expressive* we need our system to be. For example, a *context system*,

[1] In this view, ! and ∇ describe respectively *garbage collection* and *sharing* [7, 8].

introduced in [30] in order to generalise SOS rules to deal with process contexts, is just a graph system, where $R : N \to \Sigma_\sigma \times G(\Sigma_\tau) \times \Sigma_\tau \times \Sigma_\sigma$, with the further restriction that $a : \underline{1} \to \underline{1}$ for all $a \in \Sigma_\tau$ (hence, for all $d \in N$, if $R(d) = \langle l, a, b, r\rangle$ then l, r have the same source and target). A *term rewriting system* [26], instead, is given by a pair $\langle \Sigma, R\rangle$ where Σ is an ordinary signature, and R is a set of *rules*, i.e., of pairs $\langle l, r\rangle$ for l, r elements of the term algebra over Σ; thanks to Proposition 1, it is just an algebraic rewriting system, where Σ_τ is empty.

Example 2 (an algebraic rewriting system). Let us consider the signatures Σ_ϵ (already introduced) and $\Sigma_\eta = \Sigma_1 = \{u, v, w\}$. Our running example will be the algebraic rewriting system $\mathcal{R}_e = \langle \Sigma_\epsilon, \Sigma_\eta, N_e, R_e\rangle$, such that

$$R_e = \{d : a \xrightarrow[u]{id} b, d_1 : f \xrightarrow[v]{u} g, d_2 : f \xrightarrow[w]{v} f, d_3 : h \xrightarrow[w]{u\otimes v} !_{\underline{1}} \otimes g\},$$

(where id is shorthand for $id_{\underline{0}}$) described, in pictorial form, by the tiles

$$\begin{tikzcd} 0 \arrow[r, "a"] \arrow[d, "id"'] \arrow[dr, phantom, "d"] & 1 \arrow[d, "u"] \\ 0 \arrow[r, "b"'] & 1 \end{tikzcd} \qquad \begin{tikzcd} 1 \arrow[r, "f"] \arrow[d, "u"'] \arrow[dr, phantom, "d_1"] & 1 \arrow[d, "v"] \\ 1 \arrow[r, "g"'] & 1 \end{tikzcd} \qquad \begin{tikzcd} 1 \arrow[r, "f"] \arrow[d, "v"'] \arrow[dr, phantom, "d_2"] & 1 \arrow[d, "w"] \\ 1 \arrow[r, "f"'] & 1 \end{tikzcd} \qquad \begin{tikzcd} 2 \arrow[r, "h"] \arrow[d, "u\otimes v"'] \arrow[dr, phantom, "d_3"] & 1 \arrow[d, "w"] \\ 2 \arrow[r, "!_{\underline{1}}\otimes g"'] & 1 \end{tikzcd}$$

The intuitive meaning of rule d is that the element a can be rewritten to b, producing an effect u; of d_1, that f can be rewritten to g, producing an effect v, whenever there exists a suitable rewrite with an effect u. Or, in the ordinary term rewriting view: The term $f(x)$ is rewritten to $g(x)$, producing an effect v, but the rule can be applied only if the eventual sub-term associated to x is rewritten with an effect u; and so on for the other rules. $\square$

How to recover the actual behaviour of a large system, from the behaviours of its basic components? We consider a rewriting system $\mathcal{R}$ as a logical theory, and any rewrite —using rules in $\mathcal{R}$— as a *sequent* entailed by that theory. A sequent is therefore a five-tuple $\langle \alpha, s, a, b, t\rangle$, where $s \to t$ is a rewriting step, α is a *proof term* (an encoding of the causes of the step), a and b are respectively the input and output conditions, the *observations* associated to the rewrite. In the following, we say that *s rewrites to t via α* (using a *trigger a* and producing an *effect b*) if the sequent $\alpha : s \xrightarrow[b]{a} t$ can be obtained by finitely many applications of the set of *inference rules* described below.[2]

[2] Note that, from the point of view of the inference system, there was no need to restrict our attention to rules whose effect and trigger are just elements of a graph theory. The reason for this choice lies in the simpler characterisation of both the algebraic and the categorical semantics (respectively in Section 3.2 and Section 5) of the logic, yet obtaining at the same time a device that is powerful enough for our main case study in Section 4.

Definition 7 (the tile logic). *Let* $\mathcal{R} = \langle \Sigma_\sigma, \Sigma_\tau, N, R\rangle$ *be an* ARS. *We say that* $\mathcal{R}$ entails *the set* $T(\mathcal{R})$ *of* algebraic sequents $\alpha : s \xrightarrow[b]{a} t$ *obtained by a finite number of applications of the following set of* inference rules*: The* basic rules

$$\text{(gen)}\ \frac{d : s \xrightarrow[b]{a} t \in R}{d : s \xrightarrow[b]{a} t \in T(\mathcal{R})}$$

$$\text{(h-refl)}\ \frac{s : \underline{n} \to \underline{m} \in \mathbf{A}(\Sigma_\sigma)}{id^s : s \xrightarrow[id]{id} s \in T(\mathcal{R})} \qquad \text{(v-refl)}\ \frac{a : \underline{n} \to \underline{m} \in \mathbf{M}(\Sigma_\tau)}{id_a : id \xrightarrow[a]{a} id \in T(\mathcal{R})}$$

(where id *is shorthand for both* $id_{\underline{n}}$ *and* $id_{\underline{m}}$*); the* composition rules

$$\text{(par)}\ \frac{\alpha : s \xrightarrow[b]{a} t, \beta : u \xrightarrow[d]{c} v \in T(\mathcal{R})}{\alpha \otimes \beta : s \otimes u \xrightarrow[b\otimes d]{a\otimes c} t \otimes v \in T(\mathcal{R})}$$

$$\text{(hor)}\ \frac{\alpha : s \xrightarrow[c]{a} t, \beta : u \xrightarrow[b]{c} v \in T(\mathcal{R})}{\alpha * \beta : s; u \xrightarrow[b]{a} t; v \in T(\mathcal{R})}$$

$$\text{(vert)}\ \frac{\alpha : s \xrightarrow[b]{a} u, \beta : u \xrightarrow[d]{c} t \in T(\mathcal{R})}{\alpha \cdot \beta : s \xrightarrow[b;d]{a;c} t \in T(\mathcal{R})};$$

and finally, the auxiliary rules

$$\text{(perm)}\ \frac{a : \underline{n} \to \underline{m}, b : \underline{n'} \to \underline{m'} \in \mathbf{M}(\Sigma_\tau)}{\rho_{a,b} : \rho_{\underline{n},\underline{n'}} \xrightarrow[b\otimes a]{a\otimes b} \rho_{\underline{m},\underline{m'}} \in T(\mathcal{R})}$$

$$\text{(dupl)}\ \frac{a : \underline{n} \to \underline{m} \in \mathbf{M}(\Sigma_\tau)}{\nabla_a : \nabla_{\underline{n}} \xrightarrow[a\otimes a]{a} \nabla_{\underline{m}} \in T(\mathcal{R})} \qquad \text{(dis)}\ \frac{a : \underline{n} \to \underline{m} \in \mathbf{M}(\Sigma_\tau)}{!_a : !_{\underline{n}} \xrightarrow[id]{a} !_{\underline{m}} \in T(\mathcal{R})}$$

(where id *is shorthand for* $id_{\underline{0}}$*).* ∎

The basic rules provide the generators of the sequents, together with suitable identity arrows, whose intuitive meaning is that an element of $\mathbf{A}(\Sigma_\sigma)$ or $\mathbf{M}(\Sigma_\tau)$ stays idle during a rewrite (that is, it rewrites to itself) showing no effect and using no trigger. The composition rules express the way in which sequents can be combined, either sequentially (*vert*), or executing them in parallel (*par*) or nesting one inside the other (*hor*). The auxiliary rules are the counterpart of the auxiliary operators in Definition 4: They provide a way of permutating (*perm*) two sequents, duplicating (*dupl*) (or better, making a copy of the associated pointer, so to say) or discharging (*dis*) a sequent (making it inaccessible from the outside), as we discussed referring to the underlying structure of algebraic theories. In fact, we could denote a sequent as *gs-monoidal* if the underlying system is gs-monoidal; or *monoidal*, if the underlying rewriting system is monoidal, and the sequents are generated using basic and composition rules only.

Example 3 (rewrites as sequents). Let us consider now the ARS $\mathcal{R}_e$ we previously defined. As an example, it entails the sequent

$$\dfrac{\dfrac{\dfrac{}{d : a \xrightarrow[u]{id} b} \quad \dfrac{}{d_1 : f \xrightarrow[v]{u} g}}{d * d_1 : a; f \xrightarrow[v]{id} b; g}\,(hor) \qquad \dfrac{}{d_2 : f \xrightarrow[w]{v} f}}{(d * d_1) * d_2 : a; f; f \xrightarrow[w]{id} b; g; f}\,(hor)$$

(with id shorthand for $id_{\underline{0}}$), where the derivation is described in a natural deduction style. Both d and d_1 are axioms: They are combined through a *horizontal* composition, obtaining the sequent $d * d_1$, that expresses the simultaneous execution of two rewrites on nested sub-terms of $a; f$ (that is, of the term $f(a)$). Then, the resulting sequent is, again horizontally, composed with d_2.

Another example is represented by the sequent

$$\dfrac{\dfrac{\dfrac{}{d : a \xrightarrow[u]{id} b} \quad \dfrac{\dfrac{}{d : a \xrightarrow[u]{id} b} \quad \dfrac{}{d_1 : f \xrightarrow[v]{u} g}}{d * d_1 : a; f \xrightarrow[v]{id} b; g}\,(hor)}{d \otimes (d * d_1) : a \otimes (a; f) \xrightarrow[u\otimes v]{id} b \otimes (b; g)}\,(par) \qquad \dfrac{}{d_3 : h \xrightarrow[w]{u\otimes v} !_{\underline{1}} \otimes g}}{(d \otimes (d * d_1)) * d_3 : (a \otimes (a; f)); h \xrightarrow[w]{id} b; g; g}\,(hor)$$

At first, the sequent d is combined in parallel with $d * d_1$, so that they act simultaneously on disjoint parts of $a \otimes (a; f)$, corresponding to the ordered pair of terms $\langle a, f(a)\rangle$. Then, the resulting sequent $d \otimes (d * d_1)$ is nested inside d_3. A different sequent entailed by the system, yet with the same source and target of $d \otimes (d * d_1)$, is obtained through the following derivation

$$\dfrac{\dfrac{\dfrac{}{d : a \xrightarrow[u]{id} b} \quad \dfrac{}{d : a \xrightarrow[u]{id} b}}{d \otimes d : a \otimes a \xrightarrow[u\otimes u]{id} b \otimes b}\,(par) \qquad \dfrac{\dfrac{}{id_u : id_{\underline{1}} \xrightarrow[u]{u} id_{\underline{1}}} \quad \dfrac{}{d_1 : f \xrightarrow[v]{u} g}}{id_u \otimes d_1 : id_{\underline{1}} \otimes f \xrightarrow[u\otimes v]{u\otimes u} id_{\underline{1}} \otimes g}\,(par)}{(d \otimes d) * (id_u \otimes d_1) : (a \otimes a); (id_{\underline{1}} \otimes f) \xrightarrow[u\otimes v]{id} (b \otimes b); (id_{\underline{1}} \otimes g)}\,(hor)$$

since $(a \otimes a); (id_{\underline{1}} \otimes f) = a \otimes (a; f)$ and $(b \otimes b); (id_{\underline{1}} \otimes g) = b \otimes (b; g)$ by the functoriality and identity axioms. ∎

3.1 An operational semantics

The set $T(\mathcal{R})$ we just defined can be regarded as too concrete, in the sense (as we argued on a different level about gs-monoidal and algebraic theories) that sequents that intuitively represent the same rewrite may have different representations. An equivalence over sequents can then be considered as an abstraction from implementation details, identifying *computationally* equivalent derivations.

Definition 8 (abstract algebraic sequents). *Let $\mathcal{R} = \langle \Sigma_\sigma, \Sigma_\tau, N, R\rangle$ be an* ARS. *We say that it entails the set $T(\mathcal{R})_E$ of* abstract algebraic sequents, *whose elements are the equivalence classes of the algebraic sequents entailed by $\mathcal{R}$ modulo a set E of axioms, which are intended to apply to the corresponding proof terms. The set E contains three* associativity *axioms, stating that all the composition operators are associative; the* functoriality *axioms*

$$(\alpha \otimes \beta) \cdot (\gamma \otimes \delta) = (\alpha \cdot \gamma) \otimes (\beta \cdot \delta) \qquad (\alpha \otimes \beta) * (\gamma \otimes \delta) = (\alpha * \gamma) \otimes (\beta * \delta)$$

$$(\alpha \cdot \beta) * (\gamma \cdot \delta) = (\alpha * \gamma) \cdot (\beta * \delta)$$

(satisfied whenever both sides are defined); the identity *axioms $id_{id_{\underline{n}}} = id^{id_{\underline{n}}}$, $id^s \cdot \alpha = \alpha = \alpha \cdot id^t$ and $id_a * \alpha = \alpha = \alpha * id_b$ for all $\alpha : s \xrightarrow[b]{a} t$; the* monoidality *axioms*

$$id^{s\otimes t} = id^s \otimes id^t \qquad id_{a\otimes b} = id_a \otimes id_b$$

$$id^{s;t} = id^s * id^t \qquad id_{a;b} = id_a \cdot id_b$$

$$\alpha \otimes id_{id_{\underline{0}}} = \alpha = id_{id_{\underline{0}}} \otimes \alpha \qquad \rho_{a\otimes b,c} = (id_a \otimes \rho_{b,c}) * (\rho_{a,c} \otimes id_b)$$

$$!_{a\otimes b} = !_a \otimes !_b \qquad \nabla_{a\otimes b} = (\nabla_a \otimes \nabla_b) * (id_a \otimes \rho_{a,b} \otimes id_b)$$

$$!_{id_{\underline{0}}} = \nabla_{id_{\underline{0}}} = \rho_{id_{\underline{0}},id_{\underline{0}}} = id_{id_{\underline{0}}}$$

for all $\alpha \in T(\mathcal{R})$, $s,t \in \mathbf{A}(\Sigma_\sigma)$ and $a,b,c \in \mathbf{M}(\Sigma_\tau)$; the coherence *axioms*

$$\nabla_a * (id_a \otimes \nabla_a) = \nabla_a * (\nabla_a \otimes id_a) \qquad \nabla_a * \rho_{a,a} = \nabla_a$$

$$\nabla_a * (id_a \otimes !_a) = id_a \qquad \rho_{a,b} * \rho_{b,a} = id_a \otimes id_b$$

for all $a,b \in \mathbf{M}(\Sigma_\tau)$; the composition *axioms*

$$\nabla_{a;b} = \nabla_a \cdot \nabla_b \qquad !_{a;b} = !_a \cdot !_b \qquad \rho_{a;b,c;d} = \rho_{a,c} \cdot \rho_{b,d}$$

for all $a;b,c;d \in \mathbf{M}(\Sigma_\tau)$; and the naturality *axioms*

$$(\alpha \otimes \beta) * \rho_{b,d} = \rho_{a,c} * (\beta \otimes \alpha)$$

$$\alpha * !_b = !_a \qquad \alpha * \nabla_b = \nabla_a * (\alpha \otimes \alpha)$$

for all $\alpha : s \xrightarrow[b]{a} t, \beta : u \xrightarrow[d]{c} v \in T(\mathcal{R})$. □

It can be easily checked that the previous axioms preserve the "borderline" of sequents: For any rewriting system $\mathcal{R}$ entailing the sequents $\alpha : s \xrightarrow[b]{a} t$ and $\beta : u \xrightarrow[d]{c} v$, whenever the proof terms α, β are equivalent, also their source and target (and trigger and effect) are equivalent as elements of the associated algebraic (monoidal, respectively) theory.

The axiomatisation we propose properly extends the one given for rewriting logic [33]. It could then be argued that this axiomatisation is able to capture the *concurrent behaviour* of a system: Each equivalence class of sequents should intuitively describe the same set of *causally unrelated computations*. This is not so different in spirit from the well-known *permutation equivalence* [5, 21], and there exists in fact a tight correspondence between the two notions [29]. For a few initial considerations about the actual degree of concurrency expressed by the axioms, we refer to [8].

Example 4 (equating sequents). Let us consider again the ARS $\mathcal{V}$. As shown in Example 3, the system entails the sequents $d \otimes (d * d_1)$ and $(d \otimes d) * (id_u \otimes d_1)$. Their respective source and target are equivalent as elements of the algebraic theory over Σ_ϵ; and also the two sequents coincide as abstract sequents, since their proof terms are identified by functoriality and identity axioms, even though they have different derivations. ∎

3.2 An observational semantics

We already noted that the abstract semantics we defined in the previous section preserves the "along the border" structure of sequents: That is, to equivalent proof terms correspond sequents whose source and target (and trigger and effect) coincide as elements of the algebraic (monoidal, respectively) theory. Such a semantics can then be considered as operational in flavour, since it does correspond to a certain extent to proof normalisation, reflecting more what we could call the degree of abstraction of a system as a computational device, that is, the way in which the deduction process is actually implemented.

Nevertheless, we should remark that the two spatial dimensions of a sequent —horizontal for source and target, vertical for effect and trigger— hardly play the same role. In fact, when we introduced pictorially the system $\mathcal{R}_e$ in Example 2, we explicitly referred to source and target as *states* of our system, and to trigger and effect as *conditions* to be verified, before applying a given rule. It seems then rather perspicuous to introduce a semantics over states, which is only observational: That is, identifying states that show the same behaviour on the input (trigger) and output (effect) components. To this end, we simplify the structure of sequents, dropping the proof term, thus recovering a generalised notion of *transition system* [24].

Definition 9 (tile transition system). *Let $\mathcal{R} = \langle \Sigma_\sigma, \Sigma_\tau, N, R \rangle$ be an* ARS. *The associated* tile transition system *is the relation $Tr(\mathcal{R}) \subseteq \mathbf{A}(\Sigma_\sigma) \times \mathbf{M}(\Sigma_\tau) \times \mathbf{M}(\Sigma_\tau) \times \mathbf{A}(\Sigma_\sigma)$ obtained dropping the first component from relation $T(\mathcal{R})$.* ∎

In the following, with an abuse of notation, we will refer to a four-tuple $\langle s, a, b, t\rangle$ as a sequent entailed by an ARS. This is justified by the fact that an equivalent relation is obtained adding to the set E of Definition 8 the axiom

$$\frac{\alpha : s \xrightarrow[b]{a} t, \beta : s \xrightarrow[b]{a} t}{\alpha = \beta}.$$

intuitively equating those five-tuples with the same border. Restricting our attention to a transition system allows us to define a suitable notion of behavioural equivalence by the well-known technique of *bisimulation* [38].

Definition 10 (tile bisimulations). *Let $\mathcal{R} = \langle \Sigma_\sigma, \Sigma_\tau, N, R\rangle$ be an* ARS. *An equivalence relation $\equiv_{\mathtt{b}} \subseteq \mathbf{A}(\Sigma_\sigma) \times \mathbf{A}(\Sigma_\sigma)$ is a* tile bisimulation *for $\mathcal{R}$ if, whenever $s \equiv_{\mathtt{b}} t$ for s, t elements of $\mathbf{A}(\Sigma_\sigma)$, then for any sequent $s \xrightarrow[b]{a} s'$ entailed by $\mathcal{R}$ a corresponding sequent $t \xrightarrow[b]{a} t'$ is entailed by $\mathcal{R}$ with $s' \equiv_{\mathtt{b}} t'$.* ▯

Proposition 2 (strong tile bisimulation). *Tile bisimulations are closed under union. Hence there exists a maximal tile bisimulation: It is called* strong tile bisimulation, *and denoted by $\equiv$.* ▯

A fundamental requirement for any behavioural equivalence is *congruence*: This allows for an inductive account of a system, where equivalent sub-terms can be interchanged, without modifying the overall behaviour. In the framework of universal algebra, an equivalence is a congruence whenever it preserves the operators. In our case, this "operator preserving" property can be restated in terms of parallel and horizontal composition.

Definition 11 (tile functoriality and algebraicity). *Let $\mathcal{R} = \langle \Sigma_\sigma, \Sigma_\tau, N, R\rangle$ be an* ARS. *An equivalence relation $\cong_{\mathtt{f}} \subseteq \mathbf{A}(\Sigma_\sigma) \times \mathbf{A}(\Sigma_\sigma)$ is* functorial *for $\mathcal{R}$ if for s, s', t, t' elements of $\mathbf{A}(\Sigma_\sigma)$, whenever $s \cong_{\mathtt{f}} t$ and $s' \cong_{\mathtt{f}} t'$, then $s; s' \cong_{\mathtt{f}} t; t'$ (whenever defined) and $s \otimes s' \cong_{\mathtt{f}} t \otimes t'$. It is* algebraic *for $\mathcal{R}$ if for $s_i, t_i : \underline{0} \to \underline{1}$, $i = 1 \ldots n$, elements of $\mathbf{A}(\Sigma_\sigma)$, whenever $s_i \cong_{\mathtt{f}} t_i$, then $(s_1 \otimes \ldots \otimes s_n); f \cong_{\mathtt{f}} (t_1 \otimes \ldots \otimes t_n); f$ for each n-ary operator $f \in \Sigma_\sigma$.* ▯

In other words, functoriality requires that the *quotient category* $\mathbf{A}(\Sigma_\sigma)/\cong_{\mathtt{f}}$ is well-defined, and it is cartesian; hence, it can be considered as some kind of congruence with respect to parallel and sequential composition. Instead, algebraicity is a weaker property, and it simply requires that the equivalence on (tuples of) *closed* terms (that is, terms with source $\underline{0}$ and target $\underline{1}$) is preserved under composition with an operator; it is indeed reminiscent of the usual notion of algebraic congruence, as it will be clarified by the results of next section.

In general, a tile bisimulation is neither functorial nor algebraic: A characterisation of the former property can be given in terms of *tile decomposition*.

Definition 12 (tile decomposition). *Let $\mathcal{R}$ be an* ARS. *We say that it is* decomposable *if for s, t elements of $\mathbf{A}(\Sigma_\sigma)$*

1. *whenever it entails a sequent $s; t \xrightarrow[b]{a} u$, then it entails also $s \xrightarrow[c]{a} s'$ and $t \xrightarrow[b]{c} t'$ with $u = s'; t'$;*
2. *whenever it entails a sequent $s \otimes t \xrightarrow[b]{a} u$, then it entails also $s \xrightarrow[b_1]{a_1} s'$ and $t \xrightarrow[b_2]{a_2} t'$ with $u = s' \otimes t'$, $a = a_1 \otimes a_2$ and $b = b_1 \otimes b_2$.* $\square$

A very simple system that is not decomposable is given by $\mathcal{R}_\alpha = \langle \Sigma_\alpha, \Sigma_a, N_a, R_a \rangle$, where $\Sigma_\alpha = \{nil : 0 \to 1, a : 1 \to 1\}$, $\Sigma_a = \{a_1 : 1 \to 1, a_2 : 1 \to 1\}$ and

$$R_a = \{act : nil; a \xrightarrow[a_1]{id} nil, cons : a \xrightarrow[a_1]{a_1} id_{\underline{1}}\}.$$

The basic sequent *act* cannot be decomposed, while its source obviously can.

Proposition 3 (decomposition and functoriality). *Let $\mathcal{R}$ be an* ARS. *If it is decomposable, then the associated strong tile bisimulation is functorial.* $\square$

The converse is not true. In fact, the strong tile bisimulation associated to $\mathcal{R}_a$ is functorial, and it is generated from the basic classes $\{nil\}, \{id_{\underline{0}}\}, \{id_{\underline{1}}\}$ and $\{a, a; a, \ldots\} = \{a^n | n \geq 1\}$, but the system does not verify the decomposition property. Note also the importance of $a_2 \in \Sigma_a$, which is responsible for the non-equivalence of $id_{\underline{1}}$ and a: Without, functoriality would not hold.

Also algebraicity may be characterized in a syntactical way, by requiring that the rules verify a very simple format.

Theorem 1 (basic components and algebraicity). *Let $\mathcal{R} = \langle \Sigma_\sigma, \Sigma_\tau, N, R \rangle$ be an* ARS. *If Σ_τ contains only unary operators, and for all rules $\alpha : s \xrightarrow[b]{a} t \in R$ we have that $s \in \Sigma_\sigma$, then the strong tile bisimulation associated to $\mathcal{R}$ is algebraic.*

Proof sketch. Let us make explicit some of the properties of the theories. General results ensure us that each arrow $s : \underline{m} \to \underline{n} \in \mathbf{A}(\Sigma_\sigma)$ is factored as $d_s; (s_1 \otimes \ldots \otimes s_n)$, for $s_i : \underline{m}_i \to \underline{1} \in \mathbf{M}(\Sigma_\sigma)$ and $d_s : \underline{m} \to \underline{m}_1 \otimes \ldots \otimes \underline{m}_n$ a *functional* (namely, an arrow obtained composing only identities, duplicators, dischargers and permutations). In addition, since all operators of Σ_τ are unary, the source and target of any arrow $b \in \mathbf{M}(\Sigma_\tau)$ coincide, and the arrow can then be factored as $b_1 \otimes \ldots \otimes b_p$ for $b_i : \underline{1} \to \underline{1}$ and $p \in \mathbb{N}$.

Algebraicity says that, given closed terms $s_i \cong_{\mathtt{f}} t_i$ and operator $f \in \Sigma_\sigma$, then $(s_1 \otimes \ldots \otimes s_n); f \cong_{\mathtt{f}} (t_1 \otimes t_n); f$. In order to prove that, for any sequent β entailed by $\mathcal{R}$ with source $(s_1 \otimes \ldots \otimes s_n); f$, a corresponding one with source

$(t_1 \otimes \ldots \otimes t_n); f$ must also be entailed. Our proof will proceed in three steps, presenting a factorisation property for sequents verifying a suitable format.

At first, we need to prove that any sequent $\gamma : u \xrightarrow[c]{id} v$, with $u, v : \underline{0} \to \underline{n} \in \mathbf{M}(\Sigma_\sigma)$ and $c : \underline{n} \to \underline{n}$, can be factored as $\gamma_1 \otimes \ldots \otimes \gamma_n$ for $\gamma_i : u_i \xrightarrow[c_i]{id} v_i$ with $u_i, v_i : \underline{0} \to \underline{1} \in \mathbf{M}(\Sigma_\sigma)$ and $c_i : \underline{1} \to \underline{1}$; basically, the proof uses the fact that the effect of any rule is an operator of Σ_τ, and it is unary.

The second step is to prove that each sequent $\delta : w \xrightarrow[d]{id} z$, with $w, z : \underline{0} \to \underline{1} \in \mathbf{M}(\Sigma_\sigma)$ and $d : \underline{1} \to \underline{1}$, can be factored as $\delta_1 \cdot \ldots \cdot \delta_k$, for $\delta_i : w_{i-1} \xrightarrow[d_i]{id} w_i$ with $w_0 = w$, $w_k = z$ and any δ_i can be entailed without using the (*vert*) rule; basically, the proof uses the previous factorisation result, and the fact that the source of any rule is just an operator.

The third step proves that for any sequent ξ entailed by $\mathcal{R}$ without using the (*vert*) rule and with initial configuration $x : \underline{0} \to \underline{1}$, and for any factorisation $(x_1 \otimes \ldots \otimes x_l); g$ of x, then ξ can be decomposed as $(\xi_1 \otimes \ldots \otimes \xi_l) * \xi'$ for $\xi_i : x_i \xrightarrow[e_i]{id} x'_i$ and $\xi' : g \xrightarrow[e]{e_1 \otimes \ldots \otimes e_l} x'$ rule.

Let us then consider again our closed terms $s_i \cong_{\mathtt{f}} t_i$ and operator $f \in \Sigma_\sigma$, and let $\beta : (s_1 \otimes \ldots \otimes s_n); f \xrightarrow[b]{id} t$ be a sequent entailed by $\mathcal{R}$ without using the (*vert*) rule. Then β can be decomposed as $(\beta_1 \otimes \ldots \otimes \beta_n) * \alpha$ for $\beta_i : s_i \xrightarrow[b_i]{id} s'_i$ and $\alpha : f \xrightarrow[b]{b_1 \otimes \ldots \otimes b_n} t'$ rule. By induction hypothesis, $\mathcal{R}$ must entails also $\beta'_1, \ldots, \beta'_n$ with $\beta'_i : t_i \xrightarrow[b_i]{id} t'_i$ and $s'_i \cong_{\mathtt{f}} t'_i$. Now, note that $t' = \delta_{t'}; t''$ for functional $\delta_{t'}$ and that the application of the functional simply results in some possible duplication and/or deletion (plus a reshuffling) of terms s'_i's and t'_i's. Then $(s'_1 \otimes \ldots \otimes s'_n); \delta_{t'}; t''$ can be factored as $(s'_{i_1} \otimes \ldots \otimes s'_{i_m}); t''$ for $i_j \in \{1 \ldots n\}$, and similarly $(t'_1 \otimes \ldots \otimes t'_n); \delta_{t'}; t''$ can be factored as $(t'_{i_1} \otimes \ldots \otimes t'_{i_m}); t''$. Since $t'' \in \mathbf{M}(\Sigma_\sigma)$ can be factored as $(u_1 \otimes \ldots u_p); h$ for $u_i \in \mathbf{M}(\Sigma_\sigma)$, the theorem holds by coinductive hypothesis. ∎

In fact, $\mathcal{R}_a$ verifies the conditions of Theorem 1, hence the associated (and rather dull) strong tile bisimulation is algebraic.

4 A Case Study on Process Algebras

In this section we show how to recast *process algebras* and their well-known operational semantics by suitable rewriting systems. In particular, we first introduce CCS [35], maybe the best known example of these formalisms; and then we define an algebraic rewriting system $\mathcal{R}_{ccs}$ which faithfully corresponds to the CCS transition system. Furthermore, we show that, when applied to the sequents entailed by $\mathcal{R}_{ccs}$, tile bisimulation provides a recasting of bisimilarity for CCS processes.

4.1 Operational semantics of CCS

In *concurrency theory* it is quite common to deal with formalisms relying on the notions of *side-effect* and *synchronisation* in determining the actual behaviour of a system. *Process (Description) Algebras* [4, 19, 35] offer a constructive way to describe *concurrent systems*, considered as structured entities (the *agents*) interacting by means of some synchronisation mechanism. A system is then a term of an algebra over a set of process constructors, on the assumption that algebraic operators represent basic concurrency features. We present here one of the best known examples of process algebra, the *Calculus of Communicating Systems* (CCS), introduced by Milner in the early Eighties (see [35] for an up-to-date presentation), restricting our attention to *finite* CCS.

Definition 13 (the Calculus of Communicating Systems). *Let* Act *be a set of atomic* actions*, ranged over by μ, with a distinguished symbol τ and equipped with an involutive function $\bar{\cdot} : Act \to Act$ preserving τ. Moreover, let $\alpha, \overline{\alpha}, \ldots$ range over* $\mathrm{Act}\backslash\{\tau\}$. *A* CCS process *(also* agent*) is a term generated by the following syntax*

$$P ::= nil, \mu.P, P\backslash_{\alpha}, P[\Phi], P_1 + P_2, P_1||P_2$$

where $\Phi : Act \to Act$ is a relabeling *(that is, a bijective function preserving involution and τ). We let $P, Q, R, \ldots$ range over the set* Proc *of processes.* ☐

In the following, we indicate as Σ_{ccs} the signature associated with CCS processes (for example, *nil* is a constant, μ stands for a unary operator, one for each element in *Act*, and so on). Given a process P, its dynamic behaviour can be described by a suitable transition system, along the lines of the SOS approach, where the transition relation is freely generated by a set of inference rules.

Definition 14 (operational semantics of CCS). *The* CCS transition system *is the relation $T_{ccs} \subseteq Proc \times Act \times Proc$ inductively generated by the following set of axioms and inference rules*

$$\frac{}{\mu.P \xrightarrow{\mu} P} \text{ for } \mu \in \mathrm{Act} \qquad \frac{P \xrightarrow{\mu} Q}{P[\Phi] \xrightarrow{\Phi(\mu)} Q[\Phi]} \text{ for } \Phi \text{ a relabeling}$$

$$\frac{P \xrightarrow{\mu} Q}{P\backslash_{\alpha} \xrightarrow{\mu} Q\backslash_{\alpha}} \text{ for } \mu \notin \{\alpha, \overline{\alpha}\}$$

$$\frac{P \xrightarrow{\mu} Q}{P + R \xrightarrow{\mu} Q} \qquad \frac{P \xrightarrow{\mu} Q}{R + P \xrightarrow{\mu} Q}$$

$$\frac{P \xrightarrow{\mu} Q}{P||R \xrightarrow{\mu} Q||R} \qquad \frac{P \xrightarrow{\alpha} Q, P' \xrightarrow{\overline{\alpha}} Q'}{P||P' \xrightarrow{\tau} Q||Q'} \qquad \frac{P \xrightarrow{\mu} Q}{R||P \xrightarrow{\mu} R||Q}$$

where $P \xrightarrow{\mu} Q$ means that $\langle P, \mu, Q\rangle \in T_{ccs}$. ☐

A process P may execute an action μ and become Q if (we can *inductively* construct a sequence of rule applications, such that) the *transition* $\langle P, \mu, Q\rangle \in T_{ccs}$. As an example, to infer that $P = (\alpha.nil + \beta.nil)||\overline{\alpha}.nil$ may evolve into $P \xrightarrow{\alpha} Q = nil||\overline{\alpha}.nil$, three different rules must be applied. A *computation* from P to Q is a chain $P = P_0 \xrightarrow{\mu_1} P_1 \ldots P_{n-1} \xrightarrow{\mu_n} P_n = Q$ of one-step transitions.

The operational semantics we just defined is however too intensional, and more abstract semantics have been introduced by defining suitable *behavioural equivalences*, which identify processes exhibiting the same *observational behaviour*. Most of them are defined on the basic notion of *bisimulation*: Two processes P, Q are *bisimilar* if, whenever P performs an action μ evolving to P', then also Q may execute that same action, evolving to Q' bisimilar to P'.

Definition 15 (bisimulations). *An equivalence relation* $\sim_{\mathsf{b}} \subseteq Proc \times Proc$ *is a* bisimulation *if, whenever* $P \sim_{\mathsf{b}} Q$ *for* P, Q *processes, then for any transition* $P \xrightarrow{\mu} P'$ *there exists a corresponding transition* $Q \xrightarrow{\mu} Q'$ *with* $Q \sim_{\mathsf{b}} Q'$. ☐

Proposition 4 (strong bisimulation). *Bisimulations are closed under union. Hence there exists a maximal bisimulation: It is called* strong bisimulation, *and denoted by* $\sim$. ☐

4.2 Using tiles for CCS

From an operational point of view a process algebra can be faithfully described by a triple $\langle \Sigma, A, R\rangle$, where Σ is the signature of the algebra of agents, A is the set of actions, and R is the set of deduction rules. Note that these rules are *conditional*: You need information on the *actions* performed by the transitions in the premise before applying a rule. Moreover, the rewriting steps are always performed *on top*: The order in which the rewrites are actually executed is important since, as an example, the correct operational behaviour of the agent $P = \alpha.\beta.nil$ is expressed saying that it executes first α and then β. If we let A_{ccs} be the signature containing all the atomic actions of Act (i.e., $A_{ccs} = \{\mu : \underline{1} \to \underline{1} \mid \mu \in Act\}$), then both those features are easily described using tile logic.

Definition 16 (the CCS rewriting system). *The* ARS $\mathcal{R}_{ccs}$ *associated with* CCS *is the tuple* $\langle \Sigma_{ccs}, A_{ccs}, N, R\rangle$, *with the following set of rules*

$$act_\mu : \mu \xrightarrow[\mu]{id} id_{\underline{1}} \qquad rel_\Phi : \Phi \xrightarrow[\Phi(\mu)]{\mu} \Phi$$

$$res_\alpha : \backslash_\alpha \xrightarrow[\mu]{\mu} \backslash_\alpha \qquad for \quad \mu \notin \{\alpha, \overline{\alpha}\}$$

$$\langle + : + \xrightarrow[\mu]{\mu \otimes id} id_{\underline{1}} \otimes !_{\underline{1}} \qquad +\rangle : + \xrightarrow[\mu]{id \otimes \mu} !_{\underline{1}} \otimes id_{\underline{1}}$$

$$\xi_l : || \xrightarrow[\mu]{\mu \otimes id} || \qquad \xi_r : || \xrightarrow[\mu]{id \otimes \mu} || \qquad \xi_s : || \xrightarrow[\tau]{\alpha \otimes \overline{\alpha}} ||$$

(where id *is shorthand for* $id_{\underline{1}}$*).* ☐

Note that there is exactly one basic rule for each operational rule of CCS; some of them (such as act_μ and rel_Φ) are parametric with respect to the set of actions or the set of relabelings, since the corresponding rules are so. The effect μ indicates that the process is actually "running", outputting the action μ. For example, the rule act_μ prefixes an idle process with the action μ, and then starts the execution, consuming that same action. There are also three rules dealing with the parallel operator: ξ_s synchronises two running processes, while ξ_l and ξ_r perform an asynchronous move, taking a running and an idle process.

Example 5 (three processes). Let us consider again the process $P = \alpha.\beta.nil$, executing sequentially first the action α, then the action β. It is not easy to model even such a simple agent in term rewriting, since the execution ordering, that is fundamental for expressing its behaviour correctly, is difficult to model in that setting. The operational behaviour is described by the sequent

$$(id^{nil;\beta} * act_\alpha) \cdot (id^{nil} * act_\beta) : nil;\beta;\alpha \xrightarrow[\alpha;\beta]{id} nil$$

whose two-steps entailment is the following

$$\dfrac{\dfrac{}{id^{nil;\beta} : nil;\beta \xrightarrow[id]{id} nil;\beta} \quad \dfrac{}{act_\alpha : \alpha \xrightarrow[\alpha]{id} id_{\underline{1}}}}{id^{nil;\beta} * act_\alpha : nil;\beta;\alpha \xrightarrow[\alpha]{id} nil;\beta}\,(hor)$$

$$\dfrac{\dfrac{}{id^{nil} : nil \xrightarrow[id]{id} nil} \quad \dfrac{}{act_\beta : \beta \xrightarrow[\beta]{id} id_{\underline{1}}}}{id^{nil} * act_\beta : nil;\beta \xrightarrow[\beta]{id} nil}\,(hor)$$

(where id is shorthand for both $id_{\underline{0}}$ and $id_{\underline{1}}$), showing the importance of effects in expressing the ordering constraints: P can execute α only if the underlying process $P' = \beta.nil$ is actually idle.

For the agent $P = (\alpha.nil)\backslash_\beta$, instead, the execution of the action α is represented by the sequent $(id^{nil} * act_\alpha) * res_\beta$

$$\dfrac{\dfrac{\dfrac{}{id^{nil} : nil \xrightarrow[id]{id} nil} \quad \dfrac{}{act_\alpha : \alpha \xrightarrow[\alpha]{id} id_{\underline{1}}}}{id^{nil} * act_\alpha : nil;\alpha \xrightarrow[\alpha]{id} nil}\,(hor) \quad \dfrac{}{res_\beta : \backslash_\beta \xrightarrow[\alpha]{\alpha} \backslash_\beta}}{(id^{nil} * act_\alpha) * res_\beta : nil;\alpha;\backslash_\beta \xrightarrow[\alpha]{id} nil;\backslash_\beta}\,(hor)$$

Nesting act_α into res_β is possible since res_β can be applied to any sequent whose effect is different from either β or $\overline{\beta}$.

Finally, the agent $P = \alpha.nil + \alpha.nil$ can execute α in two different ways, namely with the sequent $(id^{nil;\alpha} \otimes (id^{nil} * act_\alpha)) * +\rangle$

$$\cfrac{\cfrac{\cfrac{}{id^{nil;\alpha} : nil;\alpha \xrightarrow[id]{id} nil;\alpha} \quad \cfrac{}{id^{nil} * act_\alpha : nil;\alpha \xrightarrow[\alpha]{id} nil}}{id^{nil;\alpha} \otimes (id^{nil} * act_\alpha) : (nil;\alpha) \otimes (nil;\alpha) \xrightarrow[id\otimes\alpha]{id} (nil;\alpha) \otimes nil}\,(par) \qquad \cfrac{}{+\rangle : + \xrightarrow[\alpha]{id\otimes\alpha} !_{\underline{1}} \otimes id_{\underline{1}}}}{(id^{nil;\alpha} \otimes (id^{nil} * act_\alpha)) * +\rangle : ((nil;\alpha) \otimes (nil;\alpha)); + \xrightarrow[\alpha]{id} nil}\,(hor)$$

and the symmetric, yet computationally unrelated, $((id^{nil} * act_\alpha) \otimes id^{nil;\alpha}) * \langle +$ (belonging to the same equivalence class of $(id^{nil;\alpha} \otimes (id^{nil} * act_\alpha)) * \rho_{id,\alpha} * \langle +)$.
□

The abstract sequents entailed by $\mathcal{R}_{ccs}$ offer a description where many derivations are identified, corresponding to "essentially" equivalent CCS computations. This description is still more concrete than the one given by the set–theoretic relation entailed by the CCS transition system: It suffices to consider the process $P = \alpha.nil + \alpha.nil$ in Example 5. However, if we restrict ourselves to the tile transition system $Tr(\mathcal{R}_{ccs})$, an obvious adequacy result can be proved.

Proposition 5 (interleaving correspondence). *Let P, Q be* CCS *agents, and P_a, Q_a the associated elements of $\mathbf{A}(\Sigma_{ccs})$. Then the transition $P \xrightarrow{\mu} Q$ is entailed by the* CCS *transition system T_{ccs} iff the sequent $P_a \xrightarrow[\mu]{id} Q_a$ is entailed by the tile transition system $Tr(\mathcal{R}_{ccs})$.* □

There are however many more "transitions" in $Tr(\mathcal{R}_{ccs})$ than in T_{ccs}: In fact, the last result simply states that the two transition systems coincide when we restrict our attention to so-called *closed* processes, terms with source $\underline{0}$ and target $\underline{1}$. Then there is a complete coincidence between bisimilarity over CCS processes and tile bisimilarity over the corresponding elements of $\mathbf{A}(\Sigma_{ccs})$.

Proposition 6 (bisimulation correspondence). *Let P, Q be* CCS *agents, and P_a, Q_a the associated elements of $\mathbf{A}(\Sigma_{ccs})$. Then $P \sim Q$ iff $P_a \equiv Q_a$.* □

Moreover, $\mathcal{R}_{ccs}$ verifies the conditions of Theorem 1, hence the following corollary holds.

Corollary 1 (strong bisimulation is algebraic). *The strong tile bisimulation $\equiv$ associated to $\mathcal{R}_{ccs}$ is algebraic.* □

Thanks to Proposition 6, this result implies that strong bisimilarity for CCS processes is also a congruence. As an example, let P, Q be CCS agents, P_a, Q_a the associated elements of $\mathbf{A}(\Sigma_{ccs})$, and let us assume that $P \sim Q$. Hence $P_a \equiv Q_a$ and, by algebraicity, $P_a;\alpha \equiv Q_a;\alpha$, so that $\alpha.P \sim \alpha.Q$. And since also $Q_a \equiv P_a$ by symmetry, so then $(P_a \otimes Q_a);|| \equiv (Q_a \otimes P_a);||$, and $P||Q \sim Q||P$ holds.

4.3 On formats and expressiveness

In the previous section we have taken into account the classical operational semantics of CCS, and proved a correspondence theorem with our model. However, one of the claims of the paper is that tile logic represents a generalisation of SOS specifications. This is obviously true to a certain extent, since we are able to take into account contexts (that is, partially unspecified processes): A useful device both in verification [30] and truly concurrent analysis of systems [1]. Nevertheless, a natural question to be asked is if, for a given SOS specification, there exists a suitable rewriting system preserving its operational behaviour. In a first approximation, this obviously depends on which kind of system we consider. As an example, we already noted that graph systems generalise context systems [30]; moreover, they are fully adequate for those algebras in *basic deSimone format* [1], i.e., where all the rules are of the form

$$\frac{P_i \xrightarrow{a_i} Q_i \text{ for } i \in I}{f(P_1, \dots, P_n) \xrightarrow{a} g(Q_1, \dots, Q_n)}$$

where $f, g \in \Sigma_{\mathcal{P}}$, $a_i, a \in A_{\mathcal{P}}$ and $I \subseteq \{1 \dots n\}$. Moreover, all the P_i's, Q_j's are different, except for $P_k = Q_k$ with $k \notin I$.

Actually, algebraic rewriting systems allow for dealing also with process algebras in *deSimone format* [12], i.e, such that all their rules have the form

$$\frac{P_i \xrightarrow{a_i} Q_i \text{ for } i \in I}{f(P_1, \dots, P_n) \xrightarrow{a} D[Q_1 \dots Q_n]} \text{ for } Pr(a_1, \dots, a_n, a)$$

where $f \in \Sigma_{\mathcal{P}}$, $I \subseteq \{1 \dots n\}$ and all the P_i's, Q_j's are different, except for $P_k = Q_k$ with $k \notin I$, as before. Moreover, each process variable appears *at most once* in the process context D (i.e., a term with undefined sub-terms, indicated by the occurrence of process variables), a_i, a *range over* $A_{\mathcal{P}}$ and Pr is an $n+1$-ary relation over $A_{\mathcal{P}}$. Equivalently, this means that each rule actually is just a schema, corresponding to a possibly infinite set of rules: A typical example is the rule for the restriction operator $\backslash_\alpha$ of CCS we gave in the previous section.

Note that rewriting systems satisfying the "basic components" requirement of Theorem 1 are more general than SOS specifications in the deSimone format. We are then able to recast the most important result for this format, namely, that for all SOS specifications satisfying it, strong bisimulation is actually a congruence.

Thanks to their expressiveness, ARS's are able to characterise those specifications whose rules are in what we called the *algebraic format*.

Definition 17 (rules in algebraic format). *A process algebra $\mathcal{P}$ is in the* algebraic format *iff all its deduction rules have the form*

$$\frac{P_i \xrightarrow{a_i} Q_i \textit{ for } i \in I}{C[P_1 \dots P_n] \xrightarrow{a} D[Q_1 \dots Q_n]} \textit{ for } Pr(a_1, \dots, a_n, a)$$

where $I \subseteq \{1 \dots n\}$, a_i, a range over $A_{\mathcal{P}}$ and Pr *is an $n+1$-ary relation over $A_{\mathcal{P}}$. C, D are process contexts, containing* any number of times *each process variable, and all the P_i's, Q_j's are different, except for $P_k = Q_k$ with $k \notin I$.* ◻

For instance, an axiom such as

$$\frac{}{\delta P \xrightarrow{\tau} \delta P || P} \qquad \dagger$$

(which represents the spawning with replication of a process) is in the algebraic format, but not in the deSimone one. Actually, this format is maybe too expressive: A meaningful restriction would be to assume that, whenever a variable P_k, $k \notin I$, appears in $D[Q_1 \ldots Q_n]$, then it must appear also in $C[P_1 \ldots P_n]$. For term rewriting, this restriction is analogous to assuming that for each rule $l \to r$ the set of free variables of r is contained in that of l.

Definition 18 (from process algebras to rewriting systems). *Let $\mathcal{P}$ be a process algebra $\langle \Sigma, A, R \rangle$ in the algebraic format. Then the associated* ARS *is the tuple $\langle \Sigma_{\mathcal{P}}, A_{\mathcal{P}}, N, R_{\mathcal{P}} \rangle$, where N is an arbitrary set of names, and $R_{\mathcal{P}}$ is the set of rules such that*

$$C \xrightarrow[a]{b} D \in R_{\mathcal{P}} \qquad \textit{iff} \qquad \frac{P_i \xrightarrow{a_i} Q_i \textit{ for } i \in I}{C[P_1 \ldots P_n] \xrightarrow{a} D[Q_1 \ldots Q_n]} \in R$$

where $b = a_1 \otimes \ldots \otimes a_n$ and $a_k = id_{\underline{1}}$ for $k \notin I$. ∎

In the previous definition we assumed that the conditions on actions are explicitly provided, dealing in this way with a possibly infinite set of rules. This is meaningful thanks to Proposition 1, which implies that, for a given process algebra $\mathcal{P}$ in the algebraic format, any process context appearing in a rule can be uniquely described by an element of the algebraic theory associated with $\Sigma_{\mathcal{P}}$. For example, the axiom † is described by

$$d_{\dagger} : \delta \xrightarrow[\tau]{id} \nabla_{\underline{1}}; (\delta \otimes id); ||$$

Proposition 7 (process algebras and sequents). *Let P, Q be terms of a process algebra $\mathcal{P} = \langle \Sigma, A, R \rangle$ in the algebraic format, and $P_{\mathcal{P}}, Q_{\mathcal{P}}$ the associated elements of $\mathbf{A}(\Sigma_{\mathcal{P}})$. Then, the transition $P \xrightarrow{a} Q$ belongs to the transition system associated to $\mathcal{P}$ iff the sequent $P_{\mathcal{P}} \xrightarrow[a]{id} Q_{\mathcal{P}}$ is entailed by $\mathcal{R}_{\mathcal{P}}$.* ∎

5 A Categorical Semantics for Rewriting Systems

The aim of this section is to provide an alternative characterisation for the notion of abstract sequent given in Definition 8. First, we sketch a categorical description for the theories defined in Section 2. Then, we introduce *double categories* and, starting from there, we characterise a rewriting system as a suitable finitary structure, a *computad*, able to generate by a free construction the different classes of abstract sequents. For the sake of space, a few categorical definitions are assumed to be known by the reader: We refer for details to [32].

5.1 Theories as free categories

The relevance of *monoidal categories* in computer science as a suitable framework for expressing basic properties of computing devices has been shown by a large amount of work in recent years. In this vein, we could consider the categorical description of Petri nets [42] proposed in [34] as one of the starting point for our work, and *gs-monoidal* categories [7] as a further enrichment, in order to deal with specific problems involving duplication and erasing of data.

Definition 19 (gs-monoidal categories). *A* gs-monoidal category $\mathbf{C}$ *is a six-tuple* $\langle \mathbf{C}_0, \otimes, e, \rho, \nabla, ! \rangle$*, where* $\langle \mathbf{C}_0, \otimes, e, \rho \rangle$ *is a symmetric strict monoidal category and* $! : Id_{\mathbf{C}_0} \Rightarrow e : \mathbf{C}_0 \to \mathbf{C}_0$, $\nabla : Id_{\underline{\mathbf{C}}_0} \Rightarrow \otimes \circ \Delta : \mathbf{C}_0 \to \mathbf{C}_0$ *are two transformations* (Δ *is the diagonal functor), such that* $!_e = \nabla_e = id_e$ *and satisfying the* coherence *axioms*

$$\begin{tikzcd} a \arrow[r, "\nabla_a"] \arrow[d, "\nabla_a"'] & a \otimes a \arrow[d, "id_a \otimes \nabla_a"] \\ a \otimes a \arrow[r, "\nabla_a \otimes id_a"'] & a \otimes a \otimes a \end{tikzcd} \qquad \begin{tikzcd} a \arrow[r, "\nabla_a"] \arrow[dr, "id_a"'] & a \otimes a \arrow[d, "id_a \otimes !_a"] \\ & a = a \otimes e \end{tikzcd} \qquad \begin{tikzcd} a \arrow[r, "\nabla_a"] \arrow[dr, "\nabla_a"'] & a \otimes a \arrow[d, "\rho_{a,a}"] \\ & a \otimes a \end{tikzcd}$$

and the monoidality *axioms*

$$\begin{tikzcd} a \otimes b \arrow[r, "\nabla_{a \otimes b}"] \arrow[dr, "\nabla_a \otimes \nabla_b"'] & a \otimes b \otimes a \otimes b \arrow[d, "id_a \otimes \rho_{b,a} \otimes id_b"] \\ & a \otimes a \otimes b \otimes b \end{tikzcd} \qquad \begin{tikzcd} e \arrow[dr, "id_e"'] & a \otimes b \arrow[l, "!_{a \otimes b}"'] \arrow[d, "!_a \otimes !_b"] \\ & e = e \otimes e \end{tikzcd}$$

A gs-monoidal functor $F : \mathbf{C} \to \mathbf{C}'$ *is a symmetric monoidal functor such that* $F(!_a) = !'_{F(a)}$ *and* $F(\nabla_a) = \nabla'_{F(a)}$*. The category of small gs-monoidal categories and gs-monoidal functors is denoted by* **GSM-Cat**. ∎

The monoidality axioms could be rephrased stating that the two transformations are actually *symmetric monoidal.* The relevant point however is that they are *not* natural: Such a requirement would change dramatically their properties, as stated by the following result, that seems to have become part of the categorical folklore in recent years (see e.g. [7, 22, 28]).

Proposition 8 (cartesianness as enriched monoidality). *Let* $\mathbf{C}$ *be a gs-monoidal category* $\langle \mathbf{C}_0, \otimes, e, \rho, \nabla, ! \rangle$ *such that* ∇ *and* $!$ *are* natural *transformations. Then* $\mathbf{C}$ *is cartesian.* ∎

Hence, a gs-monoidal category such that both ∇ and ! are natural is a category with *finite products*, and a gs-monoidal functor between such cartesian categories also preserve products. Then, we have a chain of adjunctions

$$\mathbf{GR}_p \overset{F_1}{\rightleftarrows} \mathbf{GR}_m \overset{F_2}{\rightleftarrows} \textbf{M-Cat} \overset{F_3}{\rightleftarrows} \textbf{GSM-Cat} \overset{F_4}{\rightleftarrows} \textbf{FC-Cat}$$

relating the category $\mathbf{GR}_p$ of graphs with pairing (that is, whose set of objects forms a monoid) and graph morphisms preserving the monoidal structure; the category $\mathbf{GR}_m$ of monoidal graphs and monoidal graph morphisms; and the category **M-Cat** (**GSM-Cat**, **FC-Cat**) of monoidal (gs-monoidal, with finite products) categories and monoidal (gs-monoidal) functors.

Although space limitations prevent us to provide further details, the previous chain basically means that the theories we introduced in Section 2 can actually be characterised as suitable free constructions, starting from the graph with pairing G_Σ associated to a signature Σ.

Proposition 9 (theories as free categories). *Let Σ be a signature. Then the following categorical isomorphisms hold*

$$F_1(G_\Sigma) = G(\Sigma) \qquad F_2(G(\Sigma)) = \mathbf{M}(\Sigma)$$

$$F_3(\mathbf{M}(\Sigma)) = \mathbf{GS}(\Sigma) \qquad F_4(\mathbf{GS}(\Sigma)) = \mathbf{A}(\Sigma)$$

where G_Σ is a graph with pairing, with objects the elements of $\mathbb{N}_c$, and arrows $f : \underline{n} \to \underline{1}$ for each $f \in \Sigma_n$. ∎

5.2 Some categorical notions

Introduced in [13, 14], *double categories* roughly are *category objects* in **Cat**, that is, categories whose classes of objects and arrows actually form categories, and the source, target and identity functions are actually functors (see [3] for a survey, or [11] for some recent results on *pasting*). They represent an intuitive generalisation of the better known *2-categories*, and admit the following *naïve* presentation, adapted from [25].

Definition 20 (double categories). *A double category $\underline{\mathbf{C}}$ consists of a collection $\{a, b, c, \ldots\}$ of* objects*, a collection $\{f, g, h, \ldots\}$ of* horizontal arrows*, a collection $\{x, y, z, \ldots\}$ of* vertical arrows*, and a collection $\{\alpha, \beta, \gamma, \ldots\}$ of* cells*. Objects and horizontal arrows form a category, the* horizontal 1-category $\mathbf{C}_h$*, with identity id_a for each object a; objects and vertical arrows also form a category, the* vertical 1-category $\mathbf{C}_v$*, with identity id^a for each object a. Cells are assigned* horizontal source *and* target *(which are vertical arrows), written as $\alpha : x \Rightarrow_h y$, and* vertical source *and* target *(which are horizontal arrows), written as $\alpha : f \Rightarrow_v g$. Furthermore, these arrows must be* compatible*, i.e., they satisfy particular requirements on their source and target. In graphical terms,*

$$\begin{tikzcd} a \arrow[r, "f"] \arrow[d, "x"'] \arrow[dr, phantom, "\alpha"] & b \arrow[d, "y"] \\ c \arrow[r, "g"'] & d \end{tikzcd}$$

In addition, cells can be composed both horizontally ($\alpha *_h \beta$) and vertically ($\gamma *_v \delta$)

$$
\begin{tikzcd}
a \arrow[r, "f;h"] \arrow[d, "x"'] \arrow[dr, phantom, "\alpha *_h \beta"] & e \arrow[d, "z"] \\
c \arrow[r, "g;i"'] & m
\end{tikzcd}
\quad = \quad
\begin{tikzcd}
a \arrow[r, "f"] \arrow[d, "x"'] \arrow[dr, phantom, "\alpha"] & b \arrow[r, "h"] \arrow[d, "y"] \arrow[dr, phantom, "\beta"] & e \arrow[d, "z"] \\
c \arrow[r, "g"'] & d \arrow[r, "i"'] & m
\end{tikzcd}
$$

$$
\begin{tikzcd}
a \arrow[r, "f"] \arrow[d, "x;w"'] \arrow[dr, phantom, "\gamma *_v \delta"] & b \arrow[d, "y;z"] \\
e \arrow[r, "h"'] & m
\end{tikzcd}
\quad = \quad
\begin{tikzcd}
a \arrow[r, "f"] \arrow[d, "x"'] \arrow[dr, phantom, "\gamma"] & b \arrow[d, "y"] \\
c \arrow[r, "g"] \arrow[d, "w"'] \arrow[dr, phantom, "\delta"] & d \arrow[d, "z"] \\
e \arrow[r, "h"'] & m
\end{tikzcd}
$$

Under each of these laws cells form a category, the horizontal category $\mathbf{C}_H$ *and the* vertical category $\mathbf{C}_V$ *respectively, with identities*

$$
\begin{tikzcd}
a \arrow[r, "id_a"] \arrow[d, "x"'] \arrow[dr, phantom, "1_x"] & a \arrow[d, "x"] \\
b \arrow[r, "id_b"'] & b
\end{tikzcd}
\qquad
\begin{tikzcd}
a \arrow[r, "f"] \arrow[d, "id^a"'] \arrow[dr, phantom, "1^f"] & b \arrow[d, "id^b"] \\
a \arrow[r, "f"'] & b
\end{tikzcd}
$$

Moreover, the following equations must hold

1. *whenever the composite*

$$
\begin{tikzcd}
\bullet \arrow[r] \arrow[d] \arrow[dr, phantom, "\alpha"] & \bullet \arrow[r] \arrow[d] \arrow[dr, phantom, "\beta"] & \bullet \arrow[d] \\
\bullet \arrow[r] \arrow[d] \arrow[dr, phantom, "\gamma"] & \bullet \arrow[r] \arrow[d] \arrow[dr, phantom, "\delta"] & \bullet \arrow[d] \\
\bullet \arrow[r] & \bullet \arrow[r] & \bullet
\end{tikzcd}
$$

   is well-defined, it is unique, namely $(\alpha *_v \gamma) *_h (\beta *_v \delta) = (\alpha *_h \beta) *_v (\gamma *_h \delta)$*;*

2. *the composite*

$$
\begin{tikzcd}
a \arrow[r, "f"] \arrow[d, "id^a"'] \arrow[dr, phantom, "1^f"] & b \arrow[r, "g"] \arrow[d, "id^b"] \arrow[dr, phantom, "1^g"] & c \arrow[d, "id^c"] \\
a \arrow[r, "f"'] & b \arrow[r, "g"'] & c
\end{tikzcd}
$$

   has to be $1^{f;g}$*, and similarly for vertical composition of horizontal identities;*

3. *finally, the horizontal and vertical identities*

$$
\begin{tikzcd}
a \arrow[r, "id_a"] \arrow[d, "id^a"'] \arrow[dr, phantom, "1_{id^a}"] & a \arrow[d, "id\text{`}a"] \\
a \arrow[r, "id_a"'] & a
\end{tikzcd}
\qquad
\begin{tikzcd}
a \arrow[r, "id_a"] \arrow[d, "id^a"'] \arrow[dr, phantom, "1^{id_a}"] & a \arrow[d, "id^a"] \\
a \arrow[r, "id_a"'] & a
\end{tikzcd}
$$

must coincide.

A double functor $F : \underline{\mathbf{C}} \to \underline{\mathbf{D}}$ *is a four-tuple of functions mapping objects to objects, horizontal (vertical) arrows to horizontal (vertical) arrows and cells to cells, preserving identities and compositions of all kinds. We denote by* **D-Cat** *the category of double categories and double functors.* $\square$

In the "internal category" view, we may simply consider $\mathbf{C}_h$ and $\mathbf{C}_H$ as the categories of objects and arrows of the (double) category $\underline{\mathbf{C}}$. Vertical source and target functions are then functors between these categories, and the components are the structures forming the vertical categories: For example, the set of objects of $\mathbf{C}_h$ and $\mathbf{C}_H$ forms respectively the set of objects and arrows of $\mathbf{C}_v$, and so on; while, at the same time, horizontal source and target become functors between these categories. Since we need double categories with additional structure, this is intuitively obtained enriching the underlying horizontal categories, corresponding to internalising over richer categorical universes. Notationally, we denote this structure by prefixing an adjective which indicates which category we are internalising into.

Definition 21 (monoidal double categories). *Let* $\underline{\mathbf{C}}$ *be a double category. We say that it is* monoidal *if both* $\mathbf{C}_h$ *and* $\mathbf{C}_H$ *are monoidal, vertical source and target are monoidal functors, and the monoidal structure preserves composition in the vertical categories. A double functor is* monoidal *if it preserves the additional structure.* $\square$

The preservation requirement is equivalent to imposing the functoriality of horizontal product with respect to vertical composition: That is, whenever the composite is well-defined, then $(\alpha *_v \beta) \otimes (\gamma *_v \delta) = (\alpha \otimes \gamma) *_v (\beta \otimes \delta)$. This implies then that both $\mathbf{C}_v$ and $\mathbf{C}_V$ are monoidal: In fact, given any two cells

$$\begin{tikzcd} a \arrow[r, "f"] \arrow[d, "x"'] \arrow[dr, phantom, "\alpha"] & b \arrow[d, "y"] \\ c \arrow[r, "g"'] & d \end{tikzcd} \qquad \begin{tikzcd} m \arrow[r, "h"] \arrow[d, "w"'] \arrow[dr, phantom, "\beta"] & n \arrow[d, "z"] \\ p \arrow[r, "i"'] & q \end{tikzcd}$$

then their composition is given by the cell $\gamma = \alpha \otimes \beta$, such that

$$\begin{tikzcd} a \otimes m \arrow[r, "f \otimes h"] \arrow[d, "x \otimes w"'] \arrow[dr, phantom, "\gamma"] & b \otimes n \arrow[d, "y \otimes z"] \\ c \otimes p \arrow[r, "g \otimes i"'] & d \otimes q \end{tikzcd}$$

Equivalently, we could say that a monoidal double category is a category object in **M-Cat**. And, analogously, a *gs-monoidal* double category is a category object in **GSM-Cat**.

Definition 22 (gs-monoidal double categories). *Let* $\underline{\mathbf{C}}$ *be a double category. We say that it is* gs-monoidal *if both* $\mathbf{C}_h$ *and* $\mathbf{C}_H$ *are gs-monoidal, vertical source and target are gs-monoidal functors, and the auxiliary operators preserves composition in the vertical categories. A double functor is* gs-monoidal *if it preserves the additional structure.* ◻

Hence, gs-monoidality implies the existence of three kind of cells

$$\begin{tikzcd} a \otimes b \arrow[r, "\rho_{a,b}"] \arrow[d, "x\otimes y"'] \arrow[dr, phantom, "\rho_{x,y}"] & b \otimes a \arrow[d, "y\otimes x"] \\ c \otimes d \arrow[r, "\rho_{c,d}"'] & d \otimes c \end{tikzcd} \qquad \begin{tikzcd} a \arrow[r, "\nabla_a"] \arrow[d, "x"'] \arrow[dr, phantom, "\nabla_x"] & a \otimes a \arrow[d, "x\otimes x"] \\ b \arrow[r, "\nabla_b"'] & b \otimes b \end{tikzcd} \qquad \begin{tikzcd} a \arrow[r, "!_a"] \arrow[d, "x"'] \arrow[dr, phantom, "!_x"] & e \arrow[d, "id_e"] \\ b \arrow[r, "!_b"'] & e \end{tikzcd}$$

which satisfies a few additional axioms on $\mathbf{C}_H$, making it gs-monoidal, like the naturality axioms for $\rho_{x,y}$: Given the cell $\alpha \otimes \beta$, then $(\alpha \otimes \beta) *_h \rho_{y,z} = \rho_{x,w} *_h (\beta \otimes \alpha)$. Graphically, the two cells

$$\begin{tikzcd} a \otimes m \arrow[r, "f\otimes h"] \arrow[d, "x\otimes w"'] \arrow[dr, phantom, "\alpha\otimes\beta"] & b \otimes n \arrow[r, "\rho_{b,n}"] \arrow[d, "y\otimes z"] \arrow[dr, phantom, "\rho_{y,z}"] & n \otimes b \arrow[d, "z\otimes y"] \\ c \otimes p \arrow[r, "g\otimes i"'] & d \otimes q \arrow[r, "\rho_{d,q}"'] & q \otimes d \end{tikzcd} \qquad \begin{tikzcd} a \otimes m \arrow[r, "\rho_{a,m}"] \arrow[d, "x\otimes w"'] \arrow[dr, phantom, "\rho_{x,w}"] & m \otimes a \arrow[r, "h\otimes f"] \arrow[d, "w\otimes x"] \arrow[dr, phantom, "\beta\otimes\alpha"] & n \otimes b \arrow[d, "z\otimes y"] \\ c \otimes p \arrow[r, "\rho_{c,p}"'] & p \otimes c \arrow[r, "i\otimes g"'] & q \otimes d \end{tikzcd}$$

must coincide. We say that a double category has *finite products* if it is a cat-object in **FC-Cat**: That is, if it is gs-monoidal, and also ∇ and ! satisfy the naturality axiom on $\mathbf{C}_H$. In the following, we will denote with **DFC-Cat** the category of double categories with finite products, and gs-monoidal double functors.

We want now to introduce a finitary structure, which is able to generate a double category with a free construction, starting from a given set of cells. We then present the notion of *computad* [45, 46].

Definition 23 (computads). *A* computad *is a triple* $\langle \mathbf{C}, D, S\rangle$*, where* $\mathbf{C}$ *is a category,* D *is a graph (whose set of objects coincides with the set of objects of* $\mathbf{C}$*) and* S *is a set of cells, each of which has assigned two pairs of compatible arrows in* $\mathbf{C}$ *and* D *as* horizontal *and* vertical source *and* target*, respectively. A* computad morphism $\langle F, G, h\rangle : \langle \mathbf{C}, D, S\rangle \to \langle \mathbf{C}', D', S'\rangle$ *is a triple such that* $F : \mathbf{C} \to \mathbf{C}'$ *is a functor,* $G : D \to D'$ *is a graph morphism, and* $h : S \to S'$ *is a function preserving identities and compositions of all kinds.* ◻

If we then define **Comp** as the category of computads, general results (see e.g. [2], and [9] for an applications of these results in the structured transition system approach) state the existence of an adjoint pair $\langle U_d, F_d\rangle$ between **Comp** and **D-Cat**, where U_d is the forgetful component and F_d the left adjoint. This adjoint composes the cells of a computad in all the possible ways, both horizontally and vertically, imposing further equalities in order to satisfy the axioms of a double category. Analogous adjoints exist also when computads have a more enriched structure. As an example, we indicate with **DFC-Comp** the category of computads $\langle \mathbf{C}, D, S\rangle$ such that $\mathbf{C}$ has finite products and D is monoidal: There exists a forgetful functor $U_{df} : \mathbf{DFC\text{-}Cat} \to \mathbf{DFC\text{-}Comp}$, with left-adjoint F_{df}.

5.3 Sequents as cells

From the point of view of entailment, the key component in the definition of a rewriting system $\langle \Sigma_\sigma, \Sigma_\tau, N, R\rangle$ is the relation R, which basically describes the building blocks to be used in the inductive construction of the sequents. Such a relation has its counterpart in the basic cells of a computad, which can then be used to freely generate the double category associated to the system.

Definition 24 (from rewriting systems to computads). *Let $\mathcal{R} = \langle \Sigma_\sigma, \Sigma_\tau, N, R\rangle$ be an* ARS. *The associated computad $\mathbf{C}(\mathcal{R})$ is the tuple $\langle \mathbf{A}(\Sigma_\sigma), G(\Sigma_\tau), S_R\rangle$, where S_R is the set of cells such that*

$$\begin{tikzcd} n \arrow[r, "s"] \arrow[d, "a"'] \arrow[dr, phantom, "d"] & m \arrow[d, "b"] \\ p \arrow[r, "t"'] & q \end{tikzcd} \in S_R \qquad \mathit{iff} \qquad d : s \xrightarrow[b]{a} t \in R.$$

∎

Of course, different kinds of computads (monoidal, algebraic, etc.) could be used, according to which underlying theories we consider. Nevertheless, whatever computad we have, the adjunction with the corresponding double category is pivotal in defining a model for rewriting theories: A double category is freely generated from a computad, such that its cells represent (abstract) sequents.

Definition 25 (spaces of computations). *Let $\mathcal{R}$ be an* ARS, *and let $\mathbf{C}(\mathcal{R})$ be its associated computad. Then the associated* algebraic double theory $\underline{\mathbf{A}(\mathcal{R})}$ *is the double category $F_{df}(\mathbf{C}(\mathcal{R}))$ with finite products.* ∎

The algebraic double theory truly represents a "space of computations", intended as a set of rewrites, equipped with an equivalence relation. If we consider our running example $\mathcal{R}_e$, the computad $\mathbf{C}(\mathcal{R}_e)$ has the following set of cells

$$\begin{tikzcd} 0 \arrow[r, "a"] \arrow[d, "id"'] \arrow[dr, phantom, "d"] & 1 \arrow[d, "u"] \\ 0 \arrow[r, "b"'] & 1 \end{tikzcd} \quad \begin{tikzcd} 1 \arrow[r, "f"] \arrow[d, "u"'] \arrow[dr, phantom, "d_1"] & 1 \arrow[d, "v"] \\ 1 \arrow[r, "g"'] & 1 \end{tikzcd} \quad \begin{tikzcd} 1 \arrow[r, "f"] \arrow[d, "v"'] \arrow[dr, phantom, "d_2"] & 1 \arrow[d, "w"] \\ 1 \arrow[r, "f"'] & 1 \end{tikzcd} \quad \begin{tikzcd} 2 \arrow[r, "h"] \arrow[d, "u\otimes v"'] \arrow[dr, phantom, "d_3"] & 1 \arrow[d, "w"] \\ 2 \arrow[r, "!_{\underline{1}}\otimes g"'] & 1 \end{tikzcd}$$

which are by no chance coincidental to the pictorial form for those same rules we used in Example 2. Some of the cells in $\underline{\mathbf{A}(\mathcal{R}_e)}$ are

$$
\begin{tikzcd}
0 \arrow[r, "a"] \arrow[d, "id"'] \arrow[dr, phantom, "d"] & 1 \arrow[r, "f"] \arrow[d, "u"] \arrow[dr, phantom, "d_1"] & 1 \arrow[d, "v"] \\
0 \arrow[r, "b"'] & 1 \arrow[r, "g"'] & 1
\end{tikzcd}
\qquad
\begin{tikzcd}
0 \arrow[r, "a"] \arrow[d, "id"'] \arrow[dr, phantom, "d"] & 1 \arrow[r, "f"] \arrow[d, "u"] \arrow[dr, phantom, "d_1"] & 1 \arrow[r, "f"] \arrow[d, "v"] \arrow[dr, phantom, "d_2"] & 1 \arrow[d, "w"] \\
0 \arrow[r, "b"'] & 1 \arrow[r, "g"'] & 1 \arrow[r, "f"'] & 1
\end{tikzcd}
$$

$$
\begin{tikzcd}
0 \arrow[r, "a\otimes(a;f)"] \arrow[d, "id"'] \arrow[dr, phantom, "d\otimes(d*d_1)"] & 2 \arrow[r, "h"] \arrow[d, "u\otimes v"] \arrow[dr, phantom, "d_3"] & 1 \arrow[d, "w"] \\
0 \arrow[r, "b\otimes(b;g)"'] & 2 \arrow[r, "!_{\underline{1}}\otimes g"'] & 1
\end{tikzcd}
$$

Next theorem states the correspondence, for a given ARS $\mathcal{R}$, between the cells of the associated algebraic double theory, and the families of abstract sequents.

Theorem 2 (correspondence between models). *Let $\mathcal{R}$ be an* ARS. *Then there exists a bijective, operator-preserving function $\phi : T(\mathcal{R})_E \to \underline{\mathbf{A}(\mathcal{R})}$ between the class of abstract algebraic sequents entailed by $\mathcal{R}$ and the cells of the associated algebraic double theory, such that*

$$
\begin{tikzcd}
n \arrow[r, "s"] \arrow[d, "a"'] \arrow[dr, phantom, "\phi(\alpha)"] & m \arrow[d, "b"] \\
p \arrow[r, "t"'] & q
\end{tikzcd}
\quad \in \underline{\mathbf{A}(\mathcal{R})} \qquad \textit{iff} \qquad \alpha : s \xrightarrow[b]{a} t \in T(\mathcal{R})_E
$$

Proof sketch. We inductively define ϕ over the structure of proof terms. First, note that cells are obtained simply closing with respect to the monoidal operation and the vertical and horizontal composition; moreover, the axioms of double categories with finite products are in a one-to-one correspondence with those of the abstract algebraic sequents. Hence, the intuitive function mapping each generator sequent to the corresponding generator cell, each horizontal composition of sequents to the horizontal composition of the associated cells, and so on, preserves the axioms and it is also a bijection. ∎

We close the section by explicitly providing the computad associated to CCS, since we feel that the geometrical intuition may ease the grasp of the rewriting system construction.

Definition 26 (the CCS computad). *The computad $\mathbf{C}(\mathcal{R}_{ccs})$ associated to* CCS *is the tuple $\langle \mathbf{A}(\Sigma_{ccs}), G(A_{ccs}), S_{ccs}\rangle$, where S_{ccs} is the set of cells*

$$
\begin{tikzcd}
1 \arrow[r, "\mu"] \arrow[d, "id"'] \arrow[dr, phantom, "\mathrm{act}_\mu"] & 1 \arrow[d, "\mu"] \\
1 \arrow[r, "id"'] & 1
\end{tikzcd}
\qquad
\begin{tikzcd}
1 \arrow[r, "\Phi"] \arrow[d, "\mu"'] \arrow[dr, phantom, "\mathrm{rel}_\Phi"] & 1 \arrow[d, "\Phi(\mu)"] \\
1 \arrow[r, "\Phi"'] & 1
\end{tikzcd}
$$

$$\begin{tikzcd} 1 \arrow[r, "\backslash\alpha"] \arrow[d, "\mu"'] \arrow[dr, phantom, "\mathrm{res}_\alpha"] & 1 \arrow[d, "\mu"] \\ 1 \arrow[r, "\backslash\alpha"'] & 1 \end{tikzcd} \qquad \mathit{for} \quad \mu \notin \{\alpha, \overline{\alpha}\}$$

$$\begin{tikzcd} 2 \arrow[r, "+"] \arrow[d, "\mu\otimes id"'] \arrow[dr, phantom, "\langle +"] & 1 \arrow[d, "\mu"] \\ 2 \arrow[r, "id\otimes !"'] & 1 \end{tikzcd} \qquad \begin{tikzcd} 2 \arrow[r, "+"] \arrow[d, "id\otimes\mu"'] \arrow[dr, phantom, "+\rangle"] & 1 \arrow[d, "\mu"] \\ 2 \arrow[r, "!\otimes id"'] & 1 \end{tikzcd}$$

$$\begin{tikzcd} 2 \arrow[r, "||"] \arrow[d, "\mu\otimes id"'] \arrow[dr, phantom, "\xi_l"] & 1 \arrow[d, "\mu"] \\ 2 \arrow[r, "||"'] & 1 \end{tikzcd} \qquad \begin{tikzcd} 2 \arrow[r, "||"] \arrow[d, "id\otimes\mu"'] \arrow[dr, phantom, "\xi_r"] & 1 \arrow[d, "\mu"] \\ 2 \arrow[r, "||"'] & 1 \end{tikzcd} \qquad \begin{tikzcd} 2 \arrow[r, "||"] \arrow[d, "\alpha\otimes\overline{\alpha}"'] \arrow[dr, phantom, "\xi_s"] & 1 \arrow[d, "\tau"] \\ 2 \arrow[r, "||"'] & 1 \end{tikzcd}$$

(where id is a shorthand for $id_{\underline{1}}$). $\square$

There is one cell for each rule, horizontal arrows are used to describe processes, vertical ones actions. As an example, a cell associated to the process $P = \alpha.\beta.nil$ is

$$\begin{tikzcd} 0 \arrow[r, "nil"] \arrow[d, "id"'] & 1 \arrow[r, "\beta"] \arrow[d, "id"] & 1 \arrow[r, "\alpha"] \arrow[d, "id"] & 1 \arrow[d, "\alpha"] \\ 0 \arrow[r, "nil"] \arrow[d, "id"'] & 1 \arrow[r, "\beta"] \arrow[d, "id"] & 1 \arrow[r, "id"] \arrow[d, "\beta"] & 1 \arrow[d, "\beta"] \\ 0 \arrow[r, "nil"'] & 1 \arrow[r, "id"'] & 1 \arrow[r, "id"'] & 1 \end{tikzcd}$$

showing the importance of the vertical arrows in expressing the ordering constraints: The process can execute α only if the underlying process is actually idle (i.e., the corresponding vertical arrow is $id_{\underline{1}}$).

Proposition 10 (rephrasing interleaving correspondence). *Let P, Q be* CCS *agents, and P_a, Q_a the associated elements of $\mathbf{A}(\Sigma_{ccs})$. Then the transition $P \xrightarrow{\mu} Q$ is entailed by the* CCS *transition system T_{ccs} iff there exists a cell d in $F_{df}(\mathbf{C}(\mathcal{R}_{ccs}))$ such that*

$$\begin{tikzcd} 0 \arrow[r, "P_a"] \arrow[d, "id"'] \arrow[dr, phantom, "d"] & 1 \arrow[d, "\mu"] \\ 0 \arrow[r, "Q_a"'] & 1 \end{tikzcd}$$

$\square$

Acknowledgements. We would like to thank Roberto Bruni, Andrea Corradini, Gianluigi Ferrari, Reiko Heckel, Wolfram Kahl, Narciso Martí-Oliet and José Meseguer for the interesting discussions and the careful reading of the paper.

References

1. E. Badouel and P. Darondeau. Trace nets and process automata. *Acta Informatica*, 32:647–679, 1995.
2. M. Barr and C. Wells. *Category Theory for Computing Science*. Prentice Hall, 1990.
3. A. Bastiani and C. Ehresmann. Multiple functors I: Limits relative to double categories. *Cahiers de Topologie et Géométrie Différentielle*, 15:545–621, 1974.
4. J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Computation*, 60:109–137, 1984.
5. G. Boudol. Computational semantics of term rewriting systems. In M. Nivat and J. Reynolds, editors, *Algebraic Methods in Semantics*, pages 170–235. Cambridge University Press, 1985.
6. R. Bruni and U. Montanari. Zero-safe nets, or transition synchronization made simple. In C. Palamidessi and J. Parrow, editors, *Expressiveness in Concurrency*, volume 7 of *Electronic Notes in Computer Science*. Elsevier, 1997.
7. A. Corradini and F. Gadducci. A 2-categorical presentation of term graph rewriting. In E. Moggi and G. Rosolini, editors, *Category Theory and Computer Science*, volume 1290 of *LNCS*, pages 87–105. Springer Verlag, 1997.
8. A. Corradini, F. Gadducci, and U. Montanari. Relating two categorical models of term rewriting. In J. Hsiang, editor, *Rewriting Techniques and Applications*, volume 914 of *LNCS*, pages 225–240. Springer Verlag, 1995.
9. 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.
10. V.-E. Căzănescu and Gh. Ştefănescu. Towards a new algebraic foundation of flowchart scheme theory. *Fundamenta Informaticae*, 13:171–210, 1990.
11. R. Dawson and R. Paré. General associativity and general composition for double categories. *Cahiers de Topologie ed Géométrie Différentielle Catégoriques*, 34:57–79, 1993.
12. R. De Simone. Higher level synchronizing devices in MEIJE–SCCS. *Theoret. Comput. Sci.*, 37:245–267, 1985.
13. E. Ehresmann. Catégories structurèes: I–II. *Annales École Normal Superieur*, 80:349–426, 1963.
14. E. Ehresmann. Catégories structurèes: III. *Cahiers de Topologie et Géométrie Différentielle*, 5, 1963.
15. G. Ferrari and U. Montanari. Towards the unification of models for concurrency. In A. Arnold, editor, *Trees in Algebra and Programming*, volume 431 of *LNCS*, pages 162–176. Springer Verlag, 1990.
16. G. Ferrari and U. Montanari. A tile-based coordination view of the asynchronous π-calculus. In I. Prívara and P. Ružička, editors, *Mathematical Foundations of Computer Science*, volume 1295 of *LNCS*, pages 52–70. Springer Verlag, 1997.
17. G. Ferrari and U. Montanari. Tiles for concurrent and located calculi. In C. Palamidessi and J. Parrow, editors, *Expressiveness in Concurrency*, volume 7 of *Electronic Notes in Computer Science*. Elsevier, 1997.

18. M. Hasegawa. *Models of Sharing Graphs.* PhD thesis, University of Edinburgh, Department of Computer Science, 1997.
19. C.A.R. Hoare. *Communicating Sequential Processes.* Prentice Hall, 1985.
20. H.-J. Hoenke. On partial recursive definitions and programs. In M. Karpiński, editor, *Fundamentals of Computation Theory*, volume 56 of *LNCS*, pages 260–274. Springer Verlag, 1977.
21. G. Huet and J.-J. Lévy. Computations in orthogonal rewriting systems, I. In J.-L. Lassez and G. Plotkin, editors, *Computational Logic: Essays in honour of Alan Robinson*, pages 395–414. MIT Press, 1991.
22. B. Jacobs. Semantics of weakening and contraction. *Annals of Pure and Applied Logic*, 69:73–106, 1994.
23. P. Katis, N. Sabadini, and R.F.C. Walters. Bicategories of processes. *Journal of Pure and Applied Algebra*, 115:141–178, 1997.
24. R. Keller. Formal verifications of parallel programs. *Communications of the ACM*, 7:371–384, 1976.
25. G.M. Kelly and R.H. Street. Review of the elements of 2-categories. In G.M. Kelly, editor, *Sydney Category Seminar*, volume 420 of *Lecture Notes in Mathematics*, pages 75–103. Springer Verlag, 1974.
26. J.W. Klop. Term rewriting systems. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 1, pages 1–116. Oxford University Press, 1992.
27. A. Kock and G.E. Reyes. Doctrines in categorical logic. In J. Bairwise, editor, *Handbook of Mathematical Logic*, pages 283–313. North Holland, 1977.
28. Y. Lafont. Equational reasoning with 2–dimensional diagrams. In H. Comon and J.-P. Jouannaud, editors, *Term Rewriting*, volume 909 of *LNCS*, pages 170–195. Springer Verlag, 1995.
29. C. Laneve and U. Montanari. Axiomatizing permutation equivalence in the λ-calculus. In H. Kirchner and G. Levi, editors, *Algebraic and Logic Programming*, volume 632 of *LNCS*, pages 350–363. Springer Verlag, 1992.
30. K.G. Larsen and L. Xinxin. Compositionality through an operational semantics of contexts. In M.S. Paterson, editor, *Automata, Languages and Programming*, volume 443 of *LNCS*, pages 526–539. Springer Verlag, 1990.
31. F.W. Lawvere. Functorial semantics of algebraic theories. *Proc. National Academy of Science*, 50:869–872, 1963.
32. S. Mac Lane. *Categories for the working mathematician.* Springer Verlag, 1971.
33. J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoret. Comput. Sci.*, 96:73–155, 1992.
34. J. Meseguer and U. Montanari. Petri nets are monoids. *Information and Computation*, 88:105–155, 1990.
35. R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.
36. R. Milner. Calculi for interaction. *Acta Informatica*, 33:707–737, 1996.
37. U. Montanari and F. Rossi. Graph rewriting and constraint solving for modelling distributed systems with synchronization. In P. Ciancarini and C. Hankin, editors, *Coordination Languages and Models*, volume 1061 of *LNCS*, pages 12–27. Springer Verlag, 1996.
38. D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Theoretical Computer Science*, volume 104 of *LNCS*, pages 167–183. Springer Verlag, 1981.
39. M. Pfender. Universal algebra in s-monoidal categories. Technical Report 95–22, University of Munich, Department of Mathematics, 1974.

40. G. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Aarhus University, Computer Science Department, 1981.
41. A.J. Power. An abstract formulation for rewrite systems. In D.H. Pitt, D.E. Rydehard, P. Dybjer, A.M. Pitts, and A. Poigné, editors, *Category Theory and Computer Science*, volume 389 of *LNCS*, pages 300–312. Springer Verlag, 1989.
42. W. Reisig. *Petri Nets: An Introduction.* EACTS Monographs on Theoretical Computer Science. Springer Verlag, 1985.
43. D.E. Rydehard and E.G. Stell. Foundations of equational deductions: A categorical treatment of equational proofs and unification algorithms. In D.H. Pitt, A. Poigné, and D.E. Rydehard, editors, *Category Theory and Computer Science*, volume 283 of *LNCS*, pages 114–139. Springer Verlag, 1987.
44. M.R. Sleep, M.J. Plasmeijer, and M.C. van Eekelen, editors. *Term Graph Rewriting: Theory and Practice.* Wiley, 1993.
45. R.H. Street. Higher categories, strings, cubes and simplex equations. *Applied Categorical Structures*, 3:29–77, 1995.
46. R.H. Street. Categorical structures. In M. Hazewinkel, editor, *Handbook of Algebra*, pages 529–577. Elsevier, 1996.