Università di Pisa

Dipartimento di Informatica

Technical Report : TR-96-27

The Tile Model

F. Gadducci, U. Montanari

June 30, 1996

ADDR: Corso Italia 40,56125 Pisa,Italy. TEL: +39-50-887111. FAX: +39-50-887226

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)

Abstract. In this paper we introduce a model for a wide class of computational systems, whose behaviour can be described by certain rewriting rules. We gathered our inspiration both from the world of term rewriting, in particular from the *rewriting logic* framework [Mes92], and of concurrency theory: among the others, the *structured operational semantics* [Plo81], the *context systems* [LX90] and the *structured transition systems* [CM92] approaches. Our model recollects many properties of these sources: first, it provides a compositional way to describe both the states and the sequences of transitions performed by a given system, stressing their distributed nature. Second, a suitable notion of typed proof allows to take into account also those formalisms relying on the notions of *synchronization* and *side-effects* to determine the actual behaviour of a system. Finally, an equivalence relation over sequences of transitions is defined, equipping the system under analysis with a concurrent semantics, where each equivalence class denotes a family of "computationally equivalent" behaviours, intended to correspond to the execution of the same set of (causally unrelated) events. As a further abstraction step, our model is conveniently represented using double-categories: its operational semantics is recovered with a free construction, by means of a suitable adjunction.

Keywords: algebraic specifications, categorical models, concurrency, context systems, SOS semantics.

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 to describe the semantics of rule-based computational systems. Widely spread in the field of concurrency theory, *transition systems* [Kel76] offered a useful tool for recovering suitable descriptions. They are roughly defined as a set of *states*, representing e.g. the possible memory contents, and a *transition relation* over states, where each element $\langle s, t\rangle$ denotes the evolution from the state s to the state t. Due to its simplicity, however, this view is clearly no more adequate when we need to take into account a compositional structure over states, and the transition relation

* Research partially supported by Progetto Speciale del CNR "Strumenti per la Specifica e la Verifica di Proprietà Critiche di Sistemi Concorrenti e Distribuiti".

needs to be inductively defined according to that structure. This is the case of formalisms like *Petri Nets* [Rei85], where a state is a multiset of basic components, and each of them may evolve simultaneously (i.e., *in parallel*); or *term rewriting systems* [Klo92], where states are terms of a free algebra, and rewrites are freely obtained from a basic set of deduction rules. Furthermore, we may need to consider formalisms relying on the use of *synchronization* and *side-effects* in determing the actual behaviour of a given system. Maybe, the most important breakthrough is represented by the so-called SOS approach [Plo81]: states are compositionally described as terms of a suitable 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. Along this line, further extensions, which proved fruitful for our view, are *context systems* [LX90], where the transition relation is defined not on states but on contexts, each of them describing a partially unspecified component of a system; and *structured transition systems* [FM90, CM92], where, in order to give a faithful account of the spatial distribution of a system, also transitions are equipped with an algebraic structure. The latter approach allows for us dealing with concurrency, by means of a suitable axiomatization over (sequences of) transitions: each equivalence class denotes a family of "computationally equivalent" behaviours, i.e., those *computations* corresponding to the execution of the same (causally unrelated) events.

As it happened for *rewriting logic* [Mes92], the underlying idea of our *tile model* is to take a logical viewpoint, regarding a rule-based system $\mathcal{R}$ as a logical theory, and any transition step — making use of rules in $\mathcal{R}$ — as a *sequent* entailed by the theory. The entailment relation is defined inductively by a set of *inference rules*, expressing some basic feature of the model, like its compositional and spatial properties. In particular, there are three composition rules. First, they allows different components of a system to act simultaneously, explicitly describing parallelism by a *monoidal* structure over transitions. Moreover, the compositional structure of states is reflected on computations: sub-components may synchronize and, according to their action, be contextualized. Finally, they can be sequentially composed, expressing this way the execution of a sequence of transitions.

A sequent $\alpha : s \xrightarrow[b]{a} t$ is a tuple where $s \to t$ is a rewrite step, α is a *proof term* (representing the structure of the step), a is the *trigger* of the step, and b is its *effect*. Its intuitive meaning is: the context s is rewritten to the context t, producing an effect b, but the rule can be applied only if the variables of s (representing still unspecified sub-components) are rewritten with a cumulative effect a. Moreover, two sequents α, β can be composed in parallel ($\alpha \otimes \beta$), composed sequentially ($\alpha \cdot \beta$) or contextualized ($\alpha * \beta$), varying accordingly the corresponding source, target, trigger and effect. Proof terms allow us to equip each rewriting step with a suitable encoding of its causes, while the fact that sequents carry information also about the effect of the associated computation expresses certain *restrictions* about the class of sequents a given rule can be applied to. Alternatively, a sequent can be considered as *synchronized* to its context

via its trigger and effect components. The possibility of expressing restrictions and synchronization will be fundamental when applying our paradigm to the operational description of *distributed systems*, and a suitable equivalence relation over proof terms will then provide a concurrent semantics for the systems that are under examination.

It may be that *2-categories* [KS74] are the better-known cat-enriched structure in computer science: they are categories such that each hom-set (the class 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* [BE74], 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 [RS87, Pow89, CGM95]: the arrows of the underlying category denote terms, and cells denote (equivalence classes of) rewrites (accordingly to the so-called *permutation equivalence* [Bou85, HL91]). In the paper we show that a similar adequacy result holds for our tile model, if we consider double-categories instead. Thus, the generality we claimed for out tile model is further confirmed by the richer structure of double-categories with respect to 2-categories.

Summarizing, the 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 model retains the idea of an explicit representation of transitions as an algebra, where certain basic structural axioms describe the concurrency properties of the model, together with the idea of providing a further abstraction by means of a categorical representation, that underlines the generality and "naturality" of the model.

The paper has the following structure. In Section 2 we introduce hypersignatures, an extension of the ordinary signatures in the vein of the context system approach. Also the usual notion of term is generalized, providing three concrete descriptions which formalize the assumptions implicitly made in the ordinary notions. In Section 3 we introduce our rewriting systems, providing them with suitable logics that describe the classes of derivations entailed by a system using (possibly abstract) sequents. In Section 4 we provide sound and complete categorical models for our logics by means of double-categories. In Section 5 and 6 we deal with two case studies. In the first one, we show that the standard (interleaving) operational semantics of many process algebras and context systems can be easily recovered in our framework. In the second we show that the very basic axioms of our model allow us to recover a faithful description of a truly concurrent semantics for these calculi. Finally, in the Appendix we recall the relevant categorical definitions used in the paper, which are not in the main body of the literature but are scattered among many different works.

2 Building States

We open this section recalling some basic definitions about graph theory: they will be used to introduce *hyper-signatures*[2], a generalization of the ordinary notion of 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* (ranged over by $a, b, \ldots$ and $f, g, \ldots$), 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 O_G of objects forms a monoid; it is *monoidal* if it is reflexive and also its class of arrows forms a monoid, such that, if 0 is the neutral element of O_G, then $id(0)$ is the neutral element of A_G. ▯

We can think of an ordinary, first-order signature Σ as a very simple graph, whose nodes are (underlined) natural numbers, and its arcs are univocally labeled by an operator, such that $f : \underline{n} \to \underline{1}$ iff $f \in \Sigma_n$. So, an intuitive generalization is given by a graph where each operator has associated a target that is not necessarily, as in the ordinary case, the element $\underline{1}$.

Definition 2 (Hyper-Signatures). A *hyper-signature* $\Sigma_S = \langle S, \delta \rangle$ is a set of operators S, and a function $\delta = \langle \delta_0, \delta_1 \rangle : S \to \underline{\mathbb{N}} \times \underline{\mathbb{N}}$. ▯

In more general words, a hyper-signature Σ_S is a graph with pairing such that its set of arcs is univocally labeled by elements in S. Also the usual notion of "term" can be generalized. In fact, we will present here an alternative, more concrete characterization of term algebras: it is well-suited to deal with hyper-signatures, as well as to recover structures more expressive than the ordinary term algebras, which will be used to describe actions and states of our systems.

Definition 3 (Graph Theories). Given a hyper-signature Σ_S, the associated *graph theory* $G(\Sigma_S)$ is the monoidal graph with objects the elements of the commutative monoid $(\underline{\mathbb{N}}, \otimes, \underline{0})$ of underlined natural numbers (where $\underline{0}$ is the neutral object and the sum is defined as $\underline{n} \otimes \underline{m} = \underline{n+m}$); and arrows those generated by the following inference rules:

$$(generators)\ \frac{f : \underline{n} \to \underline{m} \in S}{f : \underline{n} \to \underline{m} \in G(\Sigma_S)} \qquad (sum)\ \frac{s : \underline{n} \to \underline{m}, t : \underline{n'} \to \underline{m'}}{s \otimes t : \underline{n} \otimes \underline{n'} \to \underline{m} \otimes \underline{m'}}$$

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

satisfying the *monoidality* axiom $id_{\underline{n} \otimes \underline{m}} = id_{\underline{n}} \otimes id_{\underline{m}}$ for all $\underline{n}, \underline{m} \in \underline{\mathbb{N}}$. ▯

[2] By analogy with *hypergraphs*, where arcs may have more than two adjacent nodes.

Identities could be defined just for $\underline{0}, \underline{1}$, using the monoidality axiom to define inductively the operator for all the objects, so obtaining a finitary presentation of these theories. The solution we chose is equivalent, yet easier to get, and it is used also for the auxiliary operators we introduce in the following definitions.

Definition 4 (Monoidal Theories). Given a hyper-signature Σ_S, the associated *monoidal theory* $\mathbf{M}(\Sigma_S)$ is the monoidal graph with objects the elements of the commutative monoid $(\underline{\mathbb{N}}, \otimes, \underline{0})$ of underlined natural numbers and arrows those generated by the following inference rules:

$$(\textit{generators})\ \frac{f : \underline{n} \to \underline{m} \in S}{f : \underline{n} \to \underline{m} \in \mathbf{M}(\Sigma_S)} \qquad (\textit{sum})\ \frac{s : \underline{n} \to \underline{m}, t : \underline{n'} \to \underline{m'}}{s \otimes t : \underline{n} \otimes \underline{n'} \to \underline{m} \otimes \underline{m'}}$$

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

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); the *identity* axiom $id_{\underline{n}}; s = s = s; id_{\underline{m}}$ for all $s : \underline{n} \to \underline{m}$; and the monoidality axiom $id_{\underline{n} \otimes \underline{m}} = id_{\underline{n}} \otimes id_{\underline{m}}$ for all $\underline{n}, \underline{m} \in \underline{\mathbb{N}}$. $\square$

Finally, we present the more expressive kind of theories we deal with in our paper, *s-monoidal theories*. Introduced in [Gad96], their use as a suitable specification tool represents an original contribution of the paper.

Definition 5 (S-Monoidal Theories). Given a hyper-signature Σ_S, the associated *s-monoidal theory* $\mathbf{S}(\Sigma_S)$ is the monoidal graph with objects the elements of the commutative monoid $(\underline{\mathbb{N}}, \otimes, \underline{0})$ of underlined natural numbers and arrows those generated by the following inference rules:

$$(\textit{generators})\ \frac{f : \underline{n} \to \underline{m} \in S}{f : \underline{n} \to \underline{m} \in \mathbf{S}(\Sigma_S)} \qquad (\textit{sum})\ \frac{s : \underline{n} \to \underline{m}, t : \underline{n'} \to \underline{m'}}{s \otimes t : \underline{n} \otimes \underline{n'} \to \underline{m} \otimes \underline{m'}}$$

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

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

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

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

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

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

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

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

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

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

As for identities, also permutation and the other auxiliary operators could be inductively extended to all $\underline{n} \in \underline{\mathbb{N}}$ starting from the basic cases, interpreting in a constructive way the monoidality axioms (see the Definition 47 of s-monoidal categories for more details).

Let us consider for example the ordinary signature $\Sigma_{\sigma\epsilon} = \bigcup_{i=0}^{2} \Sigma_i$, where $\Sigma_0 = \{a, b\}$, $\Sigma_1 = \{f, g\}$ and $\Sigma_2 = \{h\}$ (that same signature is also used in the following sections). Some of the elements in $\mathbf{S}(\Sigma_{\sigma\epsilon})$ are $a_\Sigma; f_\Sigma : \underline{0} \to \underline{1}$, $f_\Sigma; g_\Sigma : \underline{1} \to \underline{1}$, $a_\Sigma; \nabla_{\underline{1}}; (f_\Sigma \otimes id_{\underline{1}}); h_\Sigma : \underline{0} \to \underline{1}$, intuitively corresponding to the terms $f(a), g(f(x))$ and $h(f(a), a)$, respectively, for a given variable x. In fact, in a different work by the first author [Gad96] it is shown that for ordinary signatures a fundamental property of correspondence holds between s-monoidal theories and *term graphs* (as defined in the introductory chapter of [EPS93]): each arrow $t_\Sigma : \underline{n} \to \underline{m}$ 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.

Definition 6 (Algebraic Theories). Given a hyper-signature Σ_S, the associated *algebraic theory* $\mathbf{A}(\Sigma_S)$ is the monoidal graph with objects the elements of the commutative monoid $(\underline{\mathbb{N}}, \otimes, \underline{0})$ of underlined natural numbers and arrows those generated by the same inference rules given for s-monoidal theories, such that also for dischargers and duplicators a naturality axiom holds, namely

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

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

For ordinary signatures, a classical result proves that algebraic theories (although usually defined in a different, yet equivalent way to our own: see Section 4) are equivalent to the usual term algebra construction.

Proposition 7 (Algebraic Theories and Term Algebras). *Let Σ be an ordinary signature. Then for all $\underline{n}, \underline{m} \in \underline{\mathbb{N}}$ 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.* ∎

The previous result states that each arrow $t_\Sigma : \underline{n} \to \underline{1}$ identifies an element t of the term algebra over the set $\{x_1, \ldots, x_n\}$: an arrow $\underline{n} \to \underline{m}$ is a m-tuple of such elements, and arrow composition is term substitution. Note that this correspondence *requires* that both ∇ and ! are natural: s-monoidal theories are in fact too concrete, since elements as $a_\Sigma; \nabla_{\underline{1}}; h_\Sigma$ and $(a_\Sigma \otimes a_\Sigma); h_\Sigma$, that intuitively represent the same term $h(a, a)$, are different.

The incremental description of algebraic theories has received little attention in the literature (see [Jac93, Laf95]), despite the relevant fact that, differently from the usual categorical free construction (as given in Section 4), all the elements of the class $\mathbf{A}(\Sigma)$ are inductively defined, making a much handier tool to deal with. In fact, the relevant point for our discussion is that, although their definitions are more involved than the classical, set-theoretical ones, algebraic (and *a fortiori* s-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 of the 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 description of term algebras (like *substitution*) implicitly enjoy. In this view, ! and ∇ describe respectively *garbage collection* and *sharing* (as discussed in [Cor96, CGM95]). As an example, let us consider the constant a: as a generator, the corresponding arrow is $a_\Sigma : \underline{0} \to \underline{1}$, while, when considered an element of the term algebra over $\{x_1, x_2\}$, the associated arrow is $!_{\underline{2}}; a_\Sigma : \underline{2} \to \underline{1}$, where $!_{\underline{2}}$ intuitively corresponds to the *garbaging* of the two variables. Also the difference between $a_\Sigma; \nabla_{\underline{1}}; h_\Sigma$ and $(a_\Sigma \otimes a_\Sigma); h_\Sigma$ has a similar justification: in the first element, the a is shared; in the latter, it is not. For our purposes, the difference between shared and unshared – discharged and undischarged – subterms does play a relevant part: the generalization offered by s-monoidal theories will be handy in Section 6, when dealing with *truly concurrent semantics* in the setting of *process algebras*.

3 Describing Systems

In this section we propose a general framework, in the vein of both the *rewriting logic* approach by Meseguer [Mes92] and the SOS *approach* by Plotkin [Plo81]. Intuitively, an *algebraic rewriting system* is just a set of rules, each of them carrying information (i.e., expressing some restrictions) on the possible behaviour of the terms to which they can be applied.

Definition 8 (Algebraic Rewriting Systems). An *algebraic rewriting system* (ARS) $\mathcal{R}$ is a tuple $\langle \Sigma_\sigma, \Sigma_\tau, N, R\rangle$, where $\Sigma_\sigma, \Sigma_\tau$ are hyper-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 $d \in L$, if $R(d) = \langle l, a, b, r\rangle$, then $l : \underline{n} \to \underline{m}, r : \underline{p} \to \underline{q}$ iff $a : \underline{n} \to \underline{p}, b : \underline{m} \to \underline{q}$. We usually write $d : l \xrightarrow[b]{a} r$. ∎

Also, a rewriting system is an *s-monoidal* (*monoidal*, *graph*) one if, in defining R, l and r are elements of $\mathbf{S}(\Sigma_\sigma)$ ($\mathbf{M}(\Sigma_\sigma)$ and $G(\Sigma_\sigma)$, respectively): the choice depends on *how expressive* we need our systems to be, as it will be made clear in the following sections. For example, a *context system* [LX90] is just a graph system, where $R : N \to \Sigma_\sigma \times G(\Sigma_\tau) \times G(\Sigma_\tau) \times \Sigma_\sigma$, 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). *Term rewriting systems* [Klo92], instead, are 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. Hence, thanks to Proposition 7, they are just a very particular case of algebraic rewriting systems, where Σ_τ is actually irrelevant.

Let us consider the ordinary signatures $\Sigma_{\sigma\epsilon}$ (already introduced) and $\Sigma_{\tau\epsilon} = \Sigma_1$, where $\Sigma_1 = \{u, v, w\}$. Our running example will be the algebraic rewriting system $\mathcal{R}_e = \langle \Sigma_{\sigma\epsilon}, \Sigma_{\tau\epsilon}, N_e, R_e\rangle$, where the function R_e is described by

$$R_e = \{d : a \xrightarrow[u]{id_0} b, d_1 : f \xrightarrow[v]{u} g, d_2 : f \xrightarrow[w]{v} f, d_3 : h \xrightarrow[w]{u \otimes v} !_{\underline{1}} \otimes g\}.$$

The intuitive meaning of the rules is: for d, the element a can be rewritten to b, producing an effect u; for d_1, f can be rewritten to g, producing an effect v, whenever there exists a suitable rewriting 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 subterm associated to x is rewritten with an effect u; and so on for the other rules.

In our view, a rewriting system $\mathcal{R}$ can be considered as a logical theory, and any rewriting –using rules in $\mathcal{R}$– as a sequent entailed by that theory. An *algebraic sequent* is then a 5-tuple $\langle \alpha, s, a, b, t\rangle$, where $s \to t$ is a rewrite step, α is a *proof term* (encoding of the causes of the step), a and b are respectively the input and output conditions, the *actions* 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 a set of *inference rules*.

Definition 9 (Algebraic Tile Logic). Let $\mathcal{R} = \langle \Sigma_\sigma, \Sigma_\tau, N, R\rangle$ be an ARS. We say that $\mathcal{R}$ *entails* the class $\mathbf{R}$ of *algebraic sequents* $\alpha : s \xrightarrow[b]{a} t$ obtained by a finite number of applications of the following inference rules:

BasicRules

$$(generators)\ \frac{d : s \xrightarrow[b]{a} t \in R}{d : s \xrightarrow[b]{a} t \in \mathbf{R}}$$

$$(h\text{-}refl.)\ \frac{s : \underline{n} \to \underline{m} \in \mathbf{S}(\Sigma_\sigma)}{id_s : s \xrightarrow[id_{\underline{m}}]{id_{\underline{n}}} s \in \mathbf{R}} \qquad (v\text{-}refl.)\ \frac{a : \underline{n} \to \underline{m} \in \mathbf{M}(\Sigma_\tau)}{id_a : id_{\underline{n}} \xrightarrow[a]{a} id_{\underline{m}} \in \mathbf{R}};$$

CompositionRules

$$(p\text{-}comp.)\ \frac{\alpha : s \xrightarrow[b]{a} t, \alpha' : s' \xrightarrow[b']{a'} t' \in \mathbf{R}}{\alpha \otimes \alpha' : s \otimes s' \xrightarrow[b \otimes b']{a \otimes a'} t \otimes t' \in \mathbf{R}}$$

$$(h\text{-}comp.)\ \frac{\alpha : s \xrightarrow[c]{a} t, \alpha' : s' \xrightarrow[b]{c} t' \in \mathbf{R}}{\alpha * \alpha' : s; s' \xrightarrow[b]{a} t; t' \in \mathbf{R}} \quad (v\text{-}comp.)\ \frac{\alpha : s \xrightarrow[b]{a} u, \alpha' : u \xrightarrow[b']{a'} t \in \mathbf{R}}{\alpha \cdot \alpha' : s \xrightarrow[b;b']{a;a'} t \in \mathbf{R}};$$

AuxiliaryRules

$$(permutating)\ \frac{a : \underline{n} \to \underline{m}, b : \underline{n}' \to \underline{m}' \in \mathbf{M}(\Sigma_\tau)}{\rho_{a,b} : \rho_{\underline{n},\underline{n}'} \xrightarrow[b \otimes a]{a \otimes b} \rho_{\underline{m},\underline{m}'} \in \mathbf{R}}$$

$$(duplicating)\ \frac{a : \underline{n} \to \underline{m} \in \mathbf{M}(\Sigma_\tau)}{\nabla_a : \nabla_{\underline{n}} \xrightarrow[a \otimes a]{a} \nabla_{\underline{m}} \in \mathbf{R}} \qquad (discharging)\ \frac{a : \underline{n} \to \underline{m} \in \mathbf{M}(\Sigma_\tau)}{!_a : !_{\underline{n}} \xrightarrow[id_{\underline{0}}]{a} !_{\underline{m}} \in \mathbf{R}}.$$

□

The different sets of rules are self-explaining. 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 $\mathbf{M}(\Sigma_\tau)$ can be rewritten to itself (showing no effect/using no trigger, so to say). Composition rules provide all the possible ways in which sequents can be composed, while auxiliary rules are the counterpart of the auxiliary operators given in the definition of algebraic theories. In fact, we will denote a tile logic as *monoidal* if the underlying rewriting theory is monoidal, and the sequents are generated by means of basic and composition rules only. Instead, a tile logic is *s-monoidal* if the underlying theory is s-monoidal, and all the three sets of rules are used.

Let us consider now the ARS $\mathcal{V}$ we previously defined. It entails the sequents

$$\dfrac{\dfrac{\dfrac{}{d : a \xrightarrow[u]{id_0} b} \quad \dfrac{}{d_1 : f \xrightarrow[v]{u} g}}{d * d_1 : a; f \xrightarrow[v]{id_0} b; g}\,(h\text{-}comp.) \qquad \dfrac{}{d_2 : f \xrightarrow[w]{v} f}}{(d * d_1) * d_2 : a; f; f \xrightarrow[w]{id_0} b; g; f}\,(h\text{-}comp.)$$

$$\dfrac{\dfrac{}{d : a \xrightarrow[u]{id_0} b} \qquad \dfrac{\dfrac{}{d : a \xrightarrow[u]{id_0} b} \quad \dfrac{}{d_1 : f \xrightarrow[v]{u} g}}{d * d_1 : a; f \xrightarrow[v]{id_0} b; g}\,(h\text{-}comp.)}{d \otimes (d * d_1) : a \otimes (a; f) \xrightarrow[u \otimes v]{id_0} b \otimes (b; g)}\,(p\text{-}comp.) \qquad \dfrac{}{d_3 : h \xrightarrow[w]{u \otimes v} !_{\underline{1}} \otimes g}}{(d \otimes (d * d_1)) * d_3 : (a \otimes (a; f)); h \xrightarrow[w]{id_0} b; g; g}\,(h\text{-}comp.)$$

(where the entailment is described in a natural deduction style). On a final remark, note that all the previous definitions can be "symmetrically" given: "transposed", so to say, in the sense that, given an ARS $\langle \Sigma_\sigma, \Sigma_\tau, N, R \rangle$, we could have used the signature Σ_τ as the one over which states are built. This means that the deduction rules of the logic could be given assuming that the auxiliary operators are on actions. For example, the duplicating rule would become

$$\dfrac{s : \underline{n} \to \underline{m} \in \mathbf{M}(\Sigma_\sigma)}{\nabla_s : s \xrightarrow[\nabla_{\underline{m}}]{\nabla_n} s \otimes s \in \mathbf{R}}.$$

This transposed view will become pivotal in Section 6.

The class $\mathbf{R}$ is too concrete, in the sense that many sequents that intuitively should represent the same rewrite have a different representation. An equivalence over sequents can then be considered as a way to abstract away from implementation details, identifying derivations that are *computationally* equivalent.

Definition 10 (Abstract Algebraic Sequents). Let $\mathcal{R} = \langle \Sigma_\sigma, \Sigma_\tau, N, R \rangle$ be an ARS. We say that it entails the class $\mathbf{R}_E$ of *abstract algebraic sequents*: equivalence classes of algebraic sequents entailed by $\mathcal{R}$ modulo the 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 $id_s \cdot \alpha = \alpha = \alpha \cdot id_t$ and $id_b * \alpha = \alpha = \alpha * id_a$ for all $\alpha : s \xrightarrow[b]{a} t$; the monoidality axioms

$$id_{s \otimes t} = id_s \otimes id_t \qquad id_{a \otimes b} = id_a \otimes id_b$$

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

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

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

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

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

$$(\alpha \otimes \alpha') * \rho_{b,b'} = \rho_{a,a'} * (\alpha' \otimes \alpha)$$

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

for all $\alpha : s \xrightarrow[b]{a} t, \alpha' : s' \xrightarrow[b']{a'} t' \in \mathbf{R}$. ∎

This axiomatization properly extends the one given for rewriting logic. Moreover, also for s-monoidal and monoidal sequents we can define a suitable abstraction: in the first case we simply assume that the naturality axioms

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

do not necessarily hold, while in the latter case we do not take into account any of the axioms involving the auxiliary operators. Note also that, as already happened for the theories of Section 2, even in this case we could have inductively defined identities, permutations and the other auxiliary operators starting from the basic cases, interpreting in a constructive way the monoidality axioms.

As an example, if we consider the ARS $\mathcal{V}$, from identity and monoidality axioms we have that

$$d \otimes (d * d_1) = (d \otimes d) * (id_u \otimes d_1)$$

hence the entailed proof terms

$$(d \otimes (d * d_1)) * d_3 \qquad ((d \otimes d) * (id_u \otimes d_1)) * d_3$$

are equivalent, despite the latter has a derivation unrelated to the one already shown for $(d \otimes (d * d_1)) * d_3$.

4 A Categorical Semantics for Rewriting Systems

The aim of this section is to provide an alternative characterization of the classes of abstract sequents we gave in Definition 10. First, we present a categorical description for the theories we defined in Section 2. Then, starting from there, we characterize a rewriting system as a suitable d-computad, able to generate by a free construction the different classes of abstract sequents. All the relevant categorical notions are given in the appendix, but note however that this section can be safely skipped at a first reading, without affecting the understanding of the rest of the paper.

Introduced in the early Sixties, algebraic theories [Law63, KR77] received a lot of attention during the Seventies from computer scientists: they were used to characterize in an alternative way the usual definition of *term algebra* over an ordinary signature. The following definition provides, for a given signature Σ, the categorical description of the associated algebraic theory $\mathbf{A}(\Sigma)$ by means of a suitable free construction.

Definition 11 (Algebraic Theories). Given an ordinary signature Σ, the associated *algebraic theory* is the cartesian category $\mathbf{A}(\Sigma)$ with finite products, freely generated from the graph with pairing G_Σ, such that

- its objects form the monoid $(\underline{\mathbb{N}}, \otimes, \underline{0})$ of underlined natural numbers: $\underline{0}$ is the neutral object and the sum is defined as $\underline{n} \otimes \underline{m} = \underline{n+m}$;
- for every operator $f \in \Sigma_n$, there is a basic arrow $f_\Sigma : \underline{n} \to \underline{1}$. $\square$

For an ordinary signature Σ, the relevant property of $\mathbf{A}(\Sigma)$ (already stated in Section 2) is that arrows from $\underline{n}$ to $\underline{m}$ are in one-to-one correspondence with m-tuples of elements of the free term algebra –over a set of n variables– associated to Σ. Even if usually defined for the ordinary case, the free construction of algebraic theories easily takes into account also the hyper-signatures we introduced in Section 2. Moreover, the free construction allows us to recover different theories, choosing different categories (hence, different adjoints).

Definition 12 (S-Monoidal Theories). Given a hyper-signature Σ_S, the associated *s-monoidal theory* is the s-monoidal category $\mathbf{S}(\Sigma_S)$, freely generated from the graph with pairing G_Σ, such that

- its objects form the monoid $(\underline{\mathbb{N}}, \otimes, \underline{0})$ of underlined natural numbers: $\underline{0}$ is the neutral object and the sum is defined as $\underline{n} \otimes \underline{m} = \underline{n+m}$;
- for every operator $f : \underline{n} \to \underline{m} \in S$, there is a basic arrow $f_\Sigma : \underline{n} \to \underline{m}$. $\square$

Similarly, alternative definitions hold also for graph and monoidal theories. And these categories form the basis for defining d-computads describing our rewriting systems.

Definition 13 (From Rewriting Systems to D-Computads). Let $\mathcal{R}$ be the ARS $\langle \Sigma_\sigma, \Sigma_\tau, N, R\rangle$. The associated algebraic d-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 d-cells such that

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

(where we avoided subscripts for the sake of readability). ∎

Of course, different kinds of d-computads (monoidal, algebraic, etc.) are defined, according to which underlying theories we consider. Moreover, there exists a suitable left-adjoint between the category of monoidal (s-monoidal, with finite products) d-computads and the category of monoidal (s-monoidal, with finite products) d-categories. Whatever d-computads we have, such an adjunction is pivotal in defining a model for rewriting theories: a double-category is freely generated from a d-computad, such that its cells represent (abstract) sequents.

Definition 14 (Spaces of Computations). Let $\mathcal{R}$ be an ARS, and let $\mathbf{C}(\mathcal{R})$ be its associated d-computad. Then the associated *algebraic d-theory* $\underline{\mathbf{A}(\mathcal{R})}$ is the d-category $F_{df}(\mathbf{C}(\mathcal{R}))$ with finite horizontal products. ∎

Let us now consider our running example $\mathcal{V}$: the d-computad $\mathbf{C}(\mathcal{V})$ has the following set of cells:

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

Some of the relevant cells (i.e., different from identities) of $\underline{\mathbf{A}(\mathcal{R})}$ are:

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

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

The following proposition states the precise correspondence, for a given ARS $\mathcal{R}$, between the cells of the associated algebraic d-theory, and the families of abstract derivations.

Proposition 15 (Correspondence between Abstract Models). *Let $\mathcal{R}$ be an* ARS. *Then there exists a bijective, operator-preserving function $\phi : \mathbf{R}_E \to \underline{\mathbf{A}(\mathcal{R})}$ between the class of abstract algebraic sequents entailed by $\mathcal{R}$ and the cells of the associated algebraic d-theory, such that*

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

∎

Since must ϕ preserve the operators, it has to be inductively defined over the structure of proof terms. The proof is quite easy: 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 horizontal 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 h-comp. of sequents to the horizontal composition of the associated cells, etc. preserves the axioms and it is also a bijection. Similar results hold also for s-monoidal and monoidal abstract sequents, simply choosing the corresponding d-categories in the free construction.

Note instead that the d-category $\underline{\mathbf{A}(\mathcal{R})}$ freely associated to the ARS $\mathcal{R}$ is just a very particular case of *model* for a rewriting system, one where the identifications between cells (hence, between sequents) are so to say the minimal needed to get a coherent structure. Actually, a very general notion of model can be provided: we give just the basic definition, referring the interested reader to [Gad96].

Definition 16 (Models of Algebraic D-Theories). Let $\underline{\mathbf{C}}$ be a cartesian d-category with finite products. A $\underline{\mathbf{C}}$-*model* for the algebraic d-theory associated to an ARS $\mathcal{R}$ is a cartesian d-functor $\mathcal{M} : \underline{\mathbf{A}(\mathcal{R})} \to \underline{\mathbf{C}}$, and a *model morphism* is a d-natural transformation between models. The category $\underline{\mathbf{C}}-\mathbf{Mod}_{\mathcal{R}}$ of $\underline{\mathbf{C}}$-models is the d-functor category $[\underline{\mathbf{A}(\mathcal{R})} \to \underline{\mathbf{C}}]$; its objects are $\underline{\mathbf{C}}$-models of $\mathcal{R}$, and its arrows are model morphisms. ∎

5 A Case Study: Process Algebras in the Interleaving View

It is quite common in *concurrency theory* to deal with formalisms relying on the notion of *side-effects* and *synchronization* in determining the actual behaviour of a system, features wich are quite difficult to recast in frameworks like (classical) term rewriting. *Process (Description) Algebras* [BK84, Hoa85, Mil89] offer a constructive way to describe *concurrent systems*, considered as structured entities (the *agents*) interacting by means of some synchronization mechanism. They define each system as a term of an algebra over a set of process constructors, building new systems from existing ones, on the assumption that algebraic

operators represent basic features of a concurrent system. In the following we present one of the better known example of process algebra, the *Calculus of Communicating Systems* (CCS), introduced by Milner in the early Eighties [Mil89], restricting ourselves, for the sake of exposition, to the case of *finite* CCS.

Definition 17 (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 $\overline{\mu}$ such that $\overline{\tau} = \tau$. Moreover, let $\alpha, \overline{\alpha}, \ldots$ range over $Act \backslash \{\tau\}$. A CCS *process* (also, *agent*) is a term freely 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* function, preserving involution and τ. Usually, we let $P, Q, R...$ 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 a unary operator 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 from a set of inference rules.

Definition 18 (Operational Semantics of CCS). The CCS *transition system* is the relation $T_{\text{CCS}} \subseteq Proc \times Act \times Proc$ inductively generated from the following set of axioms and inference rules

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

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

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

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

where $P \xrightarrow{\mu} Q$ means that $\langle P, \mu, Q \rangle \in T_{\text{CCS}}$. ∎

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

From an operational point of view, then, 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 *action* performed by the underlying transitions, 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 ordinary signature containing all the atomic actions of Act (i.e., $A_{\text{CCS}} = \{\mu : \underline{1} \to \underline{1} \mid \mu \in Act\}$), then both those features are easily described in the framework of tile logic by a suitable rewriting system.

Definition 19 (The CCS Rewriting System). The ARS $\mathcal{R}_{\text{CCS}}$ associated to CCS is the tuple $\langle \Sigma_{\text{CCS}}, A_{\text{CCS}}, N, R\rangle$, with the following set of rules:

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

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

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

$$\xi_l : || \xrightarrow[\mu]{\mu \otimes id_{\underline{1}}} || \qquad \xi_r : || \xrightarrow[\mu]{id_{\underline{1}} \otimes \mu} || \qquad \xi_s : || \xrightarrow[\tau]{\alpha \otimes \overline{\alpha}} ||$$

(where we omitted the subscripts for the sake of readibility). ∎

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 to the set of relabeling functions, since the corresponding rules are so. The effect μ indicates that the 'underlying' process is actually running, outputting the action μ. It is the rule act_μ that prefixes an idle process with the action μ, and then starts the execution, consuming that same action. There are three rules dealing with the parallel operator: ξ_s synchronizes two running processes, while ξ_l and ξ_r perform an asynchronous move, taking a running and an idle process. Note also that algebraicity (actually, just s-monoidality) is required only for the rules involving the sum, since one of the alternatives has to be discharged: as you may now guess, this remark will be useful in Section 6.

As an example of sequent construction, let us consider again the process $P = \alpha.\beta.nil$, executing sequentially first the action α, then the action β. The

computation is represented by the following sequent (whose entailment is described in a natural deduction style)

$$\dfrac{\dfrac{\dfrac{}{id_{\beta;nil} : nil;\beta \xrightarrow[id_{\underline{1}}]{id_0} nil;\beta} \quad \dfrac{}{act_\alpha : \alpha \xrightarrow[\alpha]{id_1} id_{\underline{1}}}}{id_{nil;\beta} * act_\alpha : nil;\beta;\alpha \xrightarrow[\alpha]{id_0} nil;\beta} \quad \dfrac{\dfrac{}{id_{nil} : nil \xrightarrow[id_{\underline{1}}]{id_0} nil} \quad \dfrac{}{act_\beta : \beta \xrightarrow[\beta]{id_1} id_{\underline{1}}}}{id_{nil} * act_\beta : nil;\beta \xrightarrow[\beta]{id_0} nil}}{(id_{nil;\beta} * act_\alpha) \cdot (id_{nil} * act_\beta) : nil;\beta;\alpha \xrightarrow[\alpha;\beta]{id_0} nil}$$

showing the importance of actions in expressing the ordering constraints: the process can execute α only if the underlying process is actually idle.

Note that the axioms impose an equivalence relation over sequents (i.e., over computations), and then offer a description that, even if more concrete than the one given by the set–theoretical relation entailed by a given transition system, is still somewhat "abstract": there are many derivations that are identified, corresponding to "essentially" equivalent CCS computations. There is however an obvious adequacy result, stated by the following theorem.

Proposition 20 (Computational Correspondence). *Let P, Q be* CCS *agents, and P_R, Q_R the associated elements of $\mathbf{A}(\Sigma_{\text{CCS}})$. Then the transition $P \xrightarrow{\mu} Q$ is entailed by the* CCS *operational semantics iff there exists an abstract algebraic sequent $\alpha : P_R \xrightarrow[\mu]{id_0} Q_R$ entailed by $\mathbf{R}_{\text{CCS}}$.* ∎

The correspondence is instead one-to-one if we consider the restriction $\mathbf{R}_p$ of $\mathbf{R}_{\text{CCS}}$ over $\mathbf{A}(\Sigma_{\text{CCS}}) \times G(A_{\text{CCS}}) \times G(A_{\text{CCS}}) \times \mathbf{A}(\Sigma_{\text{CCS}})$: i.e., if we consider the relation obtained simply dropping the proof terms from sequents. Or, equivalently, if we consider the class of abstract sequents obtained adding to E the axiom $(Bord)$

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

Proposition 21 (Interleaving Correspondence). *Let P, Q be* CCS *agents, and P_R, Q_R the associated elements of $\mathbf{A}(\Sigma_{\text{CCS}})$. Then the transition $P \xrightarrow{\mu} Q$ is entailed by the* CCS *operational semantics (i.e., $\langle P, \mu, Q\rangle \in T_{\text{CCS}}$) iff the abstract algebraic sequent (modulo the set of axioms $E \cup \{(Bord)\}$) $\alpha : P_R \xrightarrow[\mu]{id_0} Q_R$ is entailed by $\mathbf{R}_{\text{CCS}}$ (i.e., $\langle P_R, id_{\underline{0}}, \mu, Q_R\rangle \in \mathbf{R}_p$).* ∎

Again, we leave a more careful study of the properties of the axiomatization to Section 6. Let us take now instead a little detour, dealing with *formats*.

5.1 On Formats and Expressiveness

In the first part of this section we took into account the classical operational semantics of CCS, and proved a correspondence theorem with our tile model. The implicit claim was that our rewriting systems represent a generalization of SOS specifications. A natural question to be asked is if, given any SOS transition system, 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, monoidal systems (which generalize context systems) are fully adequate for those algebras in *basic deSimone format* [BD95], i.e., where all the rules are of the form

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

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

It is immediate to see that algebraic rewriting systems allow for dealing also with process algebras in *deSimone format* [DeS85], i.e, such that all its rules have the form

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

where each process variable appears *at most once* in the process context D (i.e., a term with undefined subterms, indicated by the occurrence of process variables), a_i, a *range over* $Act_{\mathcal{P}}$ and Pr is an $n+1$-ary relation over $Act_{\mathcal{P}}$. Equivalently, this means that each rule actually is just a schema, corresponding to an infinite set of rules, on which the relation Pr expresses some restrictions: a typical example is the rule for the restriction operator $\backslash_\alpha$ of CCS we gave in the previous section. Our ARS's are able to characterize specifications whose rules are in what we called the *algebraic format*.

Definition 22 (Rules in Algebraic Format). A process algebra $\mathcal{P}$ is in the *algebraic format* iff all its deduction rules have the form

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

where $f \in \Sigma_{\mathcal{P}}$; $I \subseteq \{1 \ldots n\}$; a_i, a range over $Act_{\mathcal{P}}$ and Pr is an $n+1$-ary relation over $Act_{\mathcal{P}}$; C, D are process contexts, containing *any number of times* each process variable. Moreover, 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 $D[Q_1 \dots Q_n]$, then it must appear also in $C[P_1 \dots P_n]$. For term rewriting, this restriction is analogous to the one assuming that for each rule $l \to r$ the set of free variables of r is contained in that of l. Proposition 24, however, holds for the more general case.

Definition 23 (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

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

where $b = a_1 \otimes \dots \otimes a_n$, such that $a_k = id_{\underline{1}}$ for $k \notin I$. ∎

In the previous definition we assumed that the conditions on actions are explicitly provided, so dealing with an eventually infinite set of rules. Note that this definition is meaningful thanks to Proposition 7, which implies that, for a given process algebra $\mathcal{P}$ in the algebraic format, any process context appearing in a rule can be univocally described by an element of the algebraic theory associated to $\Sigma_{\mathcal{P}}$. For example, axiom † is described by

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

Proposition 24 (Correspondence of Process Algebras and Tile Models). *Let P, Q be terms of the 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 $P \xrightarrow{a} Q$ is entailed by the operational semantics associated to $\mathcal{P}$ iff there exists an abstract algebraic sequent $d : P_{\mathcal{P}} \xrightarrow[a]{id_0} Q_{\mathcal{P}}$ in $\mathbf{R}_{\mathcal{P}}$.* ∎

The result is easy to prove: it exploits the property that any ARS associated to a given process algebra $\mathcal{P}$ is such that the effect of a rule can never be an identity (in particular, it is always a basic effect: an element of the signature associated to the actions of $\mathcal{P}$). Note that our discussion implicitly dealt with the "horizontal view" of rewriting systems, where the auxiliary operators were used on states. This is equivalent to assume that the relevant structure is the one on processes, while actions have a simpler one. As Section 6 will show, things are really different when a not-interleaving view in considered.

5.2 D-Computads and Process Algebras

As an easy exercise, we now explicitly describe the d-computad associated to CCS. Of course, that same description could be inferred from the results of the two previous sections, but we feel that the geometrical intuition may ease the grasp of the rewriting system construction.

Definition 25 (The CCS D-Computad). The d-computad $\mathbf{C}(\mathcal{R}_{\text{CCS}})$ associated to CCS is the tuple $\langle \mathbf{A}(\Sigma_{\text{CCS}}), G(A_{\text{CCS}}), S_{\text{CCS}} \rangle$, where S_{CCS} is the set of cells:

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

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

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

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

(where we omitted the subscripts for the sake of readability). ∎

There is one cell for each rule, and the horizontal arrows are used to describe processes, vertical ones to describe actions. The cell structure shows the symmetricity of our constructions: nothing prevents to enrich actions with the auxiliary operators, instead of states, simply "exchanging" the dimensions where they live. The d-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 $id_{\underline{1}}$).

As already happened for abstract sequents, the axioms of double-categories impose an equivalence relation over d-cells, equating many derivations corresponding to equivalent CCS derivations.

Proposition 26 (Computational Correspondence, II). *Let P, Q be* CCS *agents, and P_R, Q_R the associated elements of $\mathbf{A}(\Sigma_{\text{CCS}})$. Then the transition $P \xrightarrow{\mu} Q$ is entailed by the* CCS *operational semantics iff there exists a cell d in $F_{dc}(\mathbf{C}(\mathcal{R}_{\text{CCS}}))$ such that*

$$\begin{tikzcd} 0 \arrow[r, "P_R"] \arrow[d, "id"'] \arrow[dr, phantom, "d"] & 1 \arrow[d, "a"] \\ 0 \arrow[r, "Q_R"'] & 1 \end{tikzcd}$$

□

The analogous of the interleaving correspondence can be obtained simply considering the quotient category of $F_{dc}(\mathbf{C}(\mathcal{R}_{\text{CCS}}))$, where all the cells with the same "border" (so to say) are identified.

6 A Case Study: Process Algebras in the Truly Concurrent View

In Section 5 we presented the ordinary, interleaving operational semantics for CCS, while in this section we aim at showing an example of a truly-concurrent one, restricting ourselves, for the sake of simplicity, to a reduced, yet quite popular version of CCS. After introducing *prefixed* CCS, we describe a suitable *algebra of transitions* for this calculus, where the algebraic structure (not so differently from what we are aiming at with our tile model) conveys some information on the spatial "structure" of computations. Then we recall a truly concurrent semantics for CCS, obtained by a suitable set of axioms over transitions, equating those computations that represent the same set of "causally independent" events. Finally, we propose an alternative characterization of this semantics in terms of our tile model: we show for each agent P a (essentially one-to-one) correspondence between the class of concurrent computations originating from P, and the class of abstract sequents entailed by our logic with source an element P_f (univocally determined by P) of the monoidal theory associated to the process signature.

Definition 27 (Prefixed CCS). A *prefixed* CCS *process* is a term freely generated by the following syntax

$$P ::= nil, \sum_{i=1}^{n} \mu_i.P_i, P\backslash_\alpha, P_1||P_2$$

where $\sum_{i=1}^{n} \mu_i$ is a family of n-ary operators, ranging over $n \in \mathbb{N}$, $\mu_i \in Act$, and $\sum_{i=1}^{n} \mu_i.P_i$ is a shorthand for $\sum_{i=1}^{n} \mu_i(P_1, \ldots, P_n)$. Usually, we let P, Q, R... range over the set *PProc* of prefixed processes. ∎

Let Σ_{PCCS} be the signature associated to the prefixed version of CCS: we denote with η the intuitive embedding from $\mathbf{M}(\Sigma_{\text{PCCS}})$ to $\mathbf{M}(\Sigma_{\text{CCS}})$, associating to each prefixed agent $P = \sum_{i=1}^{n} \mu_i.P_i$ its "left-most" counterpart

$$\eta(P) = \mu_1.P_1 + (\ldots(\mu_{n-1}.P_{n-1} + \mu_n.P_n)\ldots).$$

Now, instead of providing a transition relation for prefixed CCS, we present its *algebra of transitions*. Similarly to what happens for sequents, each transition is represented by a tuple $t : P \xrightarrow{\mu} Q$, where P, Q are agents, μ is an action, t is a proof term. We extend the set of names with τ_s, which denotes the synchronization of two agents: μ still ranges over Act, ν ranges over $Act \cup \{\tau_s\}$.

Definition 28 (CCS Transitions). The class $\mathbf{Tr}^a_{\text{CCS}}$ of *asynchronous* CCS *transitions* is the one obtained by a finite number of applications of the following inference rules:

$$(p\text{-}sum)\ \frac{}{[\sum_{i=1}^{n} \mu_i.P_i\rangle_j : \sum_{i=1}^{n} \mu_i.P_i \xrightarrow{\mu_j} P_j}$$

$$(res)\ \frac{t : P \xrightarrow{\nu} Q}{t\backslash_\alpha : P\backslash_\alpha \xrightarrow{\nu} Q\backslash_\alpha}\ \text{for } \mu \notin \{\alpha, \overline{\alpha}\}$$

$$(com_l)\ \frac{t : P \xrightarrow{\nu} Q}{t \lhd R : P||R \xrightarrow{\nu} Q||R} \qquad (com_r)\ \frac{t : P \xrightarrow{\nu} Q}{R \rhd t : R||P \xrightarrow{\nu} R||Q}$$

The class $\mathbf{Tr}_{\text{CCS}}$ of CCS *transitions* is the one obtained by a finite number of applications of the previous rules of deduction, extended with the following one:

$$(com)\ \frac{t : P \xrightarrow{\alpha} Q, t' : P' \xrightarrow{\overline{\alpha}} Q'}{t|t' : P||P' \xrightarrow{\tau_s} Q||Q'}$$

∎

The incremental definition of the class $\mathbf{Tr}_{\text{CCS}}$ of transitions has been given in order to provide some partial results on the subclass of transitions where no synchronization is performed. First, let $\mathbf{Tr}^a_p \subseteq \mathbf{Tr}_p$ be the transition relations associated to $\mathbf{Tr}^a_{\text{CCS}}$ and $\mathbf{Tr}_{\text{CCS}}$ respectively, obtained dropping the proof term: via the embedding η (and identifying τ_s with τ), $\mathbf{Tr}_p$ is contained in T_{CCS}.

Now we define a quaternary relation, called the *concurrency relation* [FM90], between transitions: whenever the pairs $\langle t_1, t_2\rangle$ and $\langle t_3, t_4\rangle$ are related, they actually represents the same two "causally independent" events. There are many different ways to provide a truly concurrent operational semantics for process algebras: i) mapping to Petri nets or event structures by induction on the syntax of the agents [Win82, GM84]; ii) or on the structure of the proofs of transitions

[Old87, DDM88, GM94]; iii) defining the concurrent computations by assigning a residue structure to the CCS transition system [BC91]; and iv) defining the concurrent computations by direct axiomatization [FM90]. However, all the proposed approaches are essentially equivalent (see e.g. [GM94]) in the sense that they yield the same equivalence on CCS computations. Here we employ the axiomatic definition of [FM90], since it is the most direct.

Definition 29 (Concurrency Relation). Let $t_i : P_i \to Q_i$, $t : P \to Q$ and s_i with $i = 1 \ldots 4$ be CCS transitions, and R a prefixed agent. The *concurrency relation* $\chi \subseteq \mathbf{Tr}^2_{\text{CCS}} \times \mathbf{Tr}^2_{\text{CCS}}$ is the symmetric relation inductively generated from the following set of axioms and inference rules

$$\frac{}{t_1 \lhd P_2 \ \underline{\text{then}}\ Q_1 \rhd t_2 \ \chi \ P_1 \rhd t_2 \ \underline{\text{then}}\ t_1 \lhd Q_2}$$

$$\frac{t_1 \ \underline{\text{then}}\ t_2 \ \chi \ t_3 \ \underline{\text{then}}\ t_4}{t_1 \backslash_\alpha \ \underline{\text{then}}\ t_2 \backslash_\alpha \ \chi \ t_3 \backslash_\alpha \ \underline{\text{then}}\ t_4 \backslash_\alpha}$$

$$\frac{t_1 \ \underline{\text{then}}\ t_2 \ \chi \ t_3 \ \underline{\text{then}}\ t_4}{t_1 \lhd R \ \underline{\text{then}}\ t_2 \lhd R \ \chi \ t_3 \lhd R \ \underline{\text{then}}\ t_4 \lhd R} \qquad \frac{t_1 \ \underline{\text{then}}\ t_2 \ \chi \ t_3 \ \underline{\text{then}}\ t_4}{R \rhd t_1 \ \underline{\text{then}}\ R \rhd t_2 \ \chi \ R \rhd t_3 \ \underline{\text{then}}\ R \rhd t_4}$$

$$\frac{t_1 \ \underline{\text{then}}\ t_2 \ \chi \ t_3 \ \underline{\text{then}}\ t_4}{t_1 | t \ \underline{\text{then}}\ t_2 \lhd Q \ \chi \ t_3 \lhd P \ \underline{\text{then}}\ t_4 | t} \qquad \frac{t_1 \ \underline{\text{then}}\ t_2 \ \chi \ t_3 \ \underline{\text{then}}\ t_4}{t | t_1 \ \underline{\text{then}}\ Q \rhd t_2 \ \chi \ P \rhd t_3 \ \underline{\text{then}}\ t | t_4}$$

$$\frac{t_1 \ \underline{\text{then}}\ t_2 \ \chi \ t_3 \ \underline{\text{then}}\ t_4 \text{ and } s_1 \ \underline{\text{then}}\ s_2 \ \chi \ s_3 \ \underline{\text{then}}\ s_4}{t_1 | s_1 \ \underline{\text{then}}\ t_2 | s_2 \ \chi \ t_3 | s_3 \ \underline{\text{then}}\ t_4 | s_4}$$

where $t_1 \ \underline{\text{then}}\ t_2 \ \chi \ t_3 \ \underline{\text{then}}\ t_4$ means that $\langle\langle t_1, t_2\rangle, \langle t_3, t_4\rangle\rangle \in \chi$. ∎

Now, we define the class $\mathbf{Cp}_{\text{CCS}}$ of CCS computations as the one containing all the sequences of transitions, quotiented by a suitable axiomatization based on the concurrency relation just described.

Definition 30 (CCS Computations). The class $\mathbf{Cp}_{\text{CCS}}$ of CCS *concurrent computations* is the one containing all the transitions of $\mathbf{Tr}_{\text{CCS}}$, closed by finite applications of the inference rule

$$(conc) \ \frac{t_1 : P \to Q, t_2 : Q \to R \in \mathbf{Tr}_{\text{CCS}}}{t_1 \cdot t_2 : P \to R}$$

and subject to the axiom

$$(diamonds) \ \frac{t_1 \ \underline{\text{then}}\ t_2 \ \chi \ t_3 \ \underline{\text{then}}\ t_4}{t_1 \cdot t_2 = t_3 \cdot t_4}$$

identifying those sequential computations which correspond to the same concurrent computations. ∎

We define now two hyper-signatures that will be used in defining the rewriting system associated to the prefixed version of CCS: the *truly concurrent process signature* $\Sigma_{\text{TC}} = \Sigma_{\text{PCCS}} \cup \{l, r : 1 \to 1\}$ and the *synchronizing signature* $A_{\text{TC}} = A_{\text{CCS}} \cup \{\tau_c : 2 \to 2, d : 1 \to 2\}$.

Definition 31 (The Truly Concurrent CCS Rewriting System). The SRS $\mathcal{R}_{\text{TC}} = \langle A_{\text{TC}}, \Sigma_{\text{TC}}, N, R_{\text{TC}} \rangle$ associated to PCCS has the following set of rules

$$\langle \sum_{i=1}^{n} \mu_i \rangle_j : \sum_{i=1}^{n} \mu_i \xrightarrow[\mu_j]{\pi_j} id_1$$

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

$$\eta_l : l \xrightarrow[\mu]{\mu} l \qquad \eta_r : r \xrightarrow[\mu]{\mu} r$$

$$\delta : || \xrightarrow[d]{id_1 \otimes id_1} l \otimes r$$

$$\delta_\alpha : \backslash_\alpha \xrightarrow[d]{d} \backslash_\alpha \otimes \backslash_\alpha \qquad \delta_l : l \xrightarrow[d]{d} l \otimes l \qquad \delta_r : r \xrightarrow[d]{d} r \otimes r$$

$$\sigma : l \otimes r \xrightarrow[\tau_c]{\alpha \otimes \overline{\alpha}} l \otimes r$$

$$\sigma_\alpha : \backslash_\alpha \otimes \backslash_\alpha \xrightarrow[\tau_c]{\tau_c} \backslash_\alpha \otimes \backslash_\alpha \qquad \sigma_l : l \otimes l \xrightarrow[\tau_c]{\tau_c} l \otimes l \qquad \sigma_r : r \otimes r \xrightarrow[\tau_c]{\tau_c} r \otimes r$$

where $\pi_j = !_{\underline{j-1}} \otimes id_{\underline{1}} \otimes !_{\underline{n-j}}$ represents the projection of the j-th element, discarding all the others. With $\mathcal{R}_{\text{ATC}}$ we denote the SRS with only the δ rules, and with $\mathcal{R}_{\text{BTC}}$ the SRS without the σ and δ rules. ∎

The following result states that a PCCS agent can be expressed by a suitable element of the monoidal theory associated to Σ_{TC}.

Proposition 32 (Most Parallel Element). *Let Σ be an ordinary signature. Then for each element $s : \underline{n} \to \underline{m}$ of $\mathbf{A}(\Sigma)$ there exists a unique element s_m of $\mathbf{M}(\Sigma)$ such that $s = (\bigotimes_{i=1}^{n} r_i); \rho; s_m$, where r_i is either $!_{\underline{1}}$ or ∇_{k_i}, $k_i \in \mathbb{N}$.* ∎

Then, thanks to the monoidality axioms, for any ordinary signature Σ, each element $s : \underline{0} \to \underline{m}$ in $\mathbf{A}(\Sigma)$ has univocally associated an element s_m of $\mathbf{M}(\Sigma)$. This means that each *truly concurrent process* P (i.e., element of $\mathbf{A}(\Sigma_{\text{TC}})$) has univocally associated an element P_m of $\mathbf{M}(\Sigma_{\text{TC}})$. Let us define now instead a function $fork : \mathbf{M}(\Sigma_{\text{TC}}) \to \mathbf{M}(\Sigma_{\text{TC}})$, associating to each truly concurrent process P an element $fork(P) = P_f$ of $\mathbf{M}(\Sigma_{\text{TC}})$ where all the parallel operators on top have been (so to say) unwind: if a given process P has inductively associated the element $P_f = \bigotimes_{i=1}^{n} p_i$, with $p_i : \underline{0} \to \underline{1}$, then

$$(\sum_{i=1}^{n} \mu_i.P_i)_f = (\sum_{i=1}^{n} \mu_i.P_i)_m, \qquad (P||Q)_f = (l.P)_f \otimes (r.Q)_f,$$

$$(P\backslash_\alpha)_f = P_f; \bigotimes_{i=1}^{n} \backslash_\alpha, \qquad (l.P)_f = P_f; \bigotimes_{i=1}^{n} l, \qquad (r.P)_f = P_f; \bigotimes_{i=1}^{n} r.$$

where the first rule means that for each agent P with a prefix on top, no unwinding is actually possible, and then $P_f = P_m$. Let $\Sigma_d = \{d : 1 \to 1\}$: a canonical sequent relates P_f and P_m for any truly concurrent CCS agent P (simply agent from now on).

Proposition 33 (Forking Sequents). *Let P be a truly concurrent* CCS *agent, and P_m, P_f the associated elements of $\mathbf{M}(\Sigma_{\text{TC}})$. Then there exists a unique sequent Δ_P in $\mathbf{R}_{\text{ATC}}$ with source P_m and target P_f. Moreover, $\Delta_P : P_m \xrightarrow[\delta]{id_0} P_f$ with $\delta \in \mathbf{M}(\Sigma_d)$.* ∎

In the following, we will usually denote with Δ and δ elements of $\mathbf{R}_{\text{ATC}}$ and $\mathbf{M}(\Sigma_d)$, respectively.

Note that there are many sequents in $\mathbf{R}_{\text{TC}}$ which cannot represent any "observationally meaningful" computation, while they may be needed in the derivation of a meaningful sequent. This is the case of the discharging sequent $!_{P_m}$, discarding a whole process P, which is needed to express projection (on the j-th component) in the $\langle \sum_{i=1}^{n} \mu_i \rangle_j$ rule. We introduce now a suitable sub-class $\mathbf{R}_{\text{WTC}}$, which characterizes *well-formed* sequents, i.e., those whose effects describe meaningful observations.

Definition 34 (Well-Formed Sequents). Let $\mathbf{M}_S(A_{\text{TC}})$ be the *symmetric monoidal* sub-class of $\mathbf{S}(A_{\text{TC}})$, i.e., the one whose elements are obtained by the rules of Definition 5, except *(discharging)* and *(duplicating)*. Then the class $\mathbf{R}_{\text{WTC}}$ of *well-formed sequents* is the sub-class of $\mathbf{R}_{\text{TC}}$ containing those abstract s-monoidal sequents $\gamma : s \xrightarrow[a]{id} t$ such that $a \in \mathbf{M}_S(A_{\text{TC}})$. ∎

We have now all the basic definitions that allow us to adequately state the existence of a mapping $\psi : \mathbf{Cp}_{\text{CCS}} \to \mathbf{R}_{\text{WTC}}$, ensuring the correspondence between the two models.

Proposition 35 (Truly Concurrent Correspondence). *An injective mapping $\psi : \mathbf{Cp}_{\text{CCS}} \to \mathbf{R}_{\text{WTC}}$ between the class of* CCS *concurrent computations and the class of well-formed sequents entailed by $\mathcal{R}_{\text{TC}}$ can be defined by induction over the algebraic structure of transitions. Moreover, for each prefixed agent P such that $\gamma : P_f \xrightarrow[a]{id} s \in \mathbf{R}_{\text{WTC}}$, there exists a unique concurrent computation $t : P \to Q \in \mathbf{Cp}_{\text{CCS}}$, a forking sequent Δ and a symmetry ρ such that $\psi(t) = \gamma \cdot \Delta \cdot \rho$.* ∎

The first half of the previous proposition assures us that the mapping does not identify computations which cannot be equated via the quaternary relation χ. The second half assures us that each well-formed sequent, up to composition with a forking sequent and a simmetry, is in fact the image of a concurrent computation. In other words, while many sequents correspond to the same abstract computation, all of them differ only by additional forkings and symmetries.

The rest of the section provides a sketch of the proof, describing in full details the mapping ψ. Its definition is reminiscent of the one given in [GM94], even if there transitions were mapped on a concrete model (Petri nets [Rei85]): here instead we have a more general setting, subsuming that case. The mapping is given in two steps, starting from the asynchronous class $\mathbf{Tr}^a_{\text{CCS}}$: if we assume that a given transition $t : P \xrightarrow{\mu} Q$ has inductively associated the sequent $\psi_a(t) = p_f \otimes (\gamma \cdot \Delta_v) \otimes q_f$, with source P_f and target Q_f, where $\gamma : u_m \xrightarrow[\mu]{id} v_m$ and p, q, u, v are truly concurrent CCS agents, then we have

$$\psi_a([\sum_{i=1}^{n} \mu_i.P_i\rangle_j) = ([\bigotimes_{i=1}^{j-1} !_{(P_i)_m} \otimes id_{\underline{1}} \otimes \bigotimes_{j+1}^{n} !_{(P_i)_m}] * \langle \sum_{i=1}^{n} \mu_i\rangle_j) \cdot \Delta_{P_j}$$

$$\psi_a(t\backslash_\alpha) = (p\backslash_\alpha)_f \otimes [(\gamma * res_\alpha) \cdot \Delta_{v\backslash_\alpha}] \otimes (q\backslash_\alpha)_f$$

$$\begin{aligned}\psi_a(t \lhd R) &= (l.p)_f \otimes [(\gamma * \eta_l) \cdot \Delta_{l.v}] \otimes (l.q)_f \otimes (r.R)_f \\ &= (l.p)_f \otimes [(\gamma * \eta_l) \cdot \Delta_{l.v}] \otimes (q||R)_f \\ \psi_a(R \rhd t) &= (l.R)_f \otimes (r.p)_f \otimes [(\gamma * \eta_r) \cdot \Delta_{r.v}] \otimes (r.q)_f \\ &= (R||p)_f \otimes [(\gamma * \eta_r) \cdot \Delta_{r.v}] \otimes (r.q)_f\end{aligned}$$

where all the p_f, etc. actually denote the associated identity cell. Thanks to ψ_a, we are able to state the following result about the correspondence of the observational behaviour.

Proposition 36 (Computational Correspondence, III). *Let P, Q be prefixed* CCS *agents, and P_f, Q_f the associated elements of $\mathbf{M}(\Sigma_{\text{TC}})$. Then the transition $P \xrightarrow{\mu} Q$ is entailed by $\mathbf{Tr}^a_r$ iff there exists an abstract s-monoidal sequent $\alpha : P_f \xrightarrow[a]{id_0} Q_f$ entailed by $\mathbf{R}_{\text{BTC}}$, where $a = id \otimes (\mu; \delta) \otimes id$.* ∎

To finally extend the mapping to the whole class $\mathbf{Tr}_{\text{CCS}}$, note that whenever we have a transition $t|t'$, then neither t nor t' have occurrences of the *com* operator inside them, so that when building the sequent associated to $t|t'$ we can use ψ_a on its arguments. Let us start defining the ψ^r_a and ψ^l_a versions of ψ_a, informally moving the effect of $\psi_a(t)$ to the far right (far left, respectively) of the effect: if $\psi_a(t) = p_f \otimes (\gamma \cdot \Delta_v) \otimes q_f$, then

$$\psi^r_a(t) = (p_f \otimes \rho_{u_m,q_f}) \cdot [p_f \otimes q_f \otimes (\gamma \cdot \Delta_v)] \cdot (p_f \otimes \rho_{q_f,v_f})$$

$$\psi^l_a(t) = (\rho_{p_f,u_m} \otimes q_f) \cdot [(\gamma \cdot \Delta_v) \otimes p_f \otimes q_f] \cdot (\rho_{v_f,p_f} \otimes q_f)$$

where all the p_f, q_f denote the associated identities, and by the naturality axiom $\psi^r_a(t) = \psi^l_a(t) = \psi_a(t)$ holds. We can now finally define the mapping ψ over the whole class of transitions: we consider only the case $t|t'$, since the others are analogous to the corresponding cases for ψ_a. We have $\psi(t|t') = \rho_i \cdot \xi \cdot \rho_e$, where

$$\rho_i = [(l.p)_f \otimes \rho_{(l.u)_m,(l.q)_f} \otimes \rho_{(r.p')_f,(r.u')_m} \otimes (r.q')_f],$$

$$\xi = \{(l.p)_f \otimes (l.q)_f \otimes \{[(\gamma \otimes \gamma') * \sigma] \cdot (\Delta_{l.v} \otimes \Delta_{r.v'})\} \otimes (r.p')_f \otimes (r.q')_f\},$$

$$\rho_e = [(l.p)_f \otimes \rho_{(l.q)_f,(l.v)_f} \otimes \rho_{(r.v')_f,(r.p')_f} \otimes (r.q')_f].$$

Furthermore, we have that $\psi(t|t') : P_f \xrightarrow[a]{id} Q_f$, with $a = \rho; [id \otimes (\tau_c; \delta) \otimes id]; \rho'$. Due to the structure of $\psi(t|t')$, the following correspondence result holds.

Proposition 37 (Computational Correspondence, IV). *Let P, Q be prefixed* CCS *agents, and P_f, Q_f the associated elements of $\mathbf{M}(\Sigma_{\text{TC}})$. Then the transition $P \xrightarrow{\tau_s} Q$ is entailed by $\mathbf{Tr}_r$ iff there exists an abstract s-monoidal sequent $\alpha : P_f \xrightarrow[a]{id_0} Q_f$ entailed by $\mathbf{R}_{\text{TC}}$, where $a = \rho; [id \otimes (\tau_c; \delta) \otimes id]; \rho'$.* ∎

The previous results state an observational correspondence between the sequents entailed by our rewriting system and the CCS transitions. Finally, next result states that the mapping ψ is in fact injective.

Proposition 38 (Correspondence of Computations and Cells). *Let t_i, $i = 1 \ldots 4$ be* CCS *transitions. Then t_1 <u>then</u> t_2 χ t_3 <u>then</u> t_4 iff $\psi(t_1) \cdot \psi(t_2) = \psi(t_3) \cdot \psi(t_4)$.* ∎

The three previous lemmata are the basis from which Proposition 35 can be actually proved, by a structural induction over proof terms.

6.1 Further Remarks on Formats

Let us conclude our work with a final investigation on formats. The main problem with the algebraic format we introduced in Section 5.1 is the restriction we have on the hypotheses. Each process variable can be the premise of just one hypothesis, so that rules of this kind

$$\frac{P \xrightarrow{a} Q,\ P \xrightarrow{b} R}{P \xrightarrow{c} Q||R}$$

are forbidden. If we enrich over actions, we can take into account rules with a very basic structure on processes, but a complex set of hypotheses.

Definition 39 (Rules in Vertical Format). A process algebra $\mathcal{P}$ is in the *vertical format* iff all its deduction rules have the form

$$\frac{P_i \xrightarrow{a_{ij}} Q_{ij} \text{ for } i = 1 \dots n, j = 1 \dots m_i}{f(P_1, \dots, P_n) \xrightarrow{a} g(R_1, \dots, R_k)}$$

where $m_i \geq 0$, $f, g \in \Sigma_{\mathcal{P}}$ and $a_{ij}, a \in Act_{\mathcal{P}}$. Moreover, all the process variables P_i's, Q_{jk}'s are different, and $R_j \in \{P_1 \dots P_n, Q_{11} \dots Q_{nm_n}\}$. ∎

In the so-called *positive* GSOS *format* (see e.g. [ABV94]) rules have the form

$$\frac{P_i \xrightarrow{a_{ij}} Q_{ij} \text{ for } i = 1 \dots n, j = 1 \dots m_i}{f(P_1, \dots, P_n) \xrightarrow{a} D[R_1 \dots R_k]}$$

where D is a process context in which each process variable can appear *any number of times*. The only difference with our vertical format lies on the freedom allowed by that context: however, we feel this is a small one, since all the relevant cases we know of fall in the range of our format.

Definition 40 (From Process Algebras to Rewriting Systems, II). Let $\mathcal{P}$ be a process algebra $\langle \Sigma, A, R \rangle$ in the vertical format. Then the associated SRS is the tuple $\langle A_{\mathcal{P}}, \Sigma_{\mathcal{P}}, N, R_{\mathcal{P}} \rangle$, where N is an arbitrary set of names, and $R_{\mathcal{P}}$ is the set of rules such that

$$f \underset{a}{\xrightarrow{b}} g \in R_{\mathcal{P}} \qquad \text{iff} \qquad \frac{P_i \xrightarrow{a_{ij}} Q_{ij} \text{ for } i = 1 \dots n, j = 1 \dots m_i}{f(P_1, \dots, P_n) \xrightarrow{a} g(R_1, \dots R_k)}$$

where $b = \bigotimes_{i=1}^{n} (\nabla_{1+m_i}; \bigotimes_{j=0}^{m_i} (a_{i,j}; r_{i,j})); \rho$: $a_{i,0} = id_{\underline{1}}$ and $r_{i,j} = id_{\underline{1}}$ if $Q_{i,j} \in \{R_1 \dots R_k\}$ (P_i if $j = 0$), $r_{i,j} = !_{\underline{1}}$ otherwise. ∎

It is very important that we are dealing with s-monoidal theories. In fact, the absence of the naturality axioms is fundamental for defining "enriched" actions which express correctly copying and discarding of hypotheses. If the naturality axiom holds also for ∇, then $\nabla_{\underline{1}}; (a \otimes a) = a; \nabla_{\underline{1}}$ and it would become impossible to find suitable sequents making distinct the following rules

$$\frac{P \xrightarrow{a} Q, P \xrightarrow{a} R}{f(P) \xrightarrow{b} h(Q, R)} \qquad \frac{P \xrightarrow{a} Q}{f(P) \xrightarrow{b} h(Q, Q)}$$

As for !, the monoidality axiom states that $a; !_{\underline{1}} = !_{\underline{1}}$, so that it would become impossible to find suitable sequents making distinct the following rules

$$\frac{P \xrightarrow{a} Q, R \xrightarrow{b} S}{h(P, R) \xrightarrow{c} f(Q)} \qquad \frac{P \xrightarrow{a} Q}{h(P, R) \xrightarrow{c} f(Q)}$$

The symmetry ρ given in Definition 40 is informally defined as the "symmetry ordering the variables for g". Its importance in defining the trigger associated to a given rule is shown by the following, very simple case

$$\frac{P \xrightarrow{a} Q, R \xrightarrow{b} S}{h(P,R) \xrightarrow{c} h(S,Q)}$$

where the associated rule is $h \xrightarrow[c]{d} h$, with $d = (a \otimes b); \rho_{\underline{1},\underline{1}}$.

Thanks to Proposition 32, we can state our last correspondence proposition.

Proposition 41 (Correspondence of Process Algebras and Tile Models, II). *Let P, Q be terms of the process algebra $\mathcal{P} = \langle \Sigma, A, R \rangle$ in the vertical format, and $P_{\mathcal{P}}, Q_{\mathcal{P}}$ the associated most parallel elements of $M_{\Sigma_{\mathcal{P}}}$. Then the transition $P \xrightarrow{a} Q$ is entailed by the operational semantics associated to $\mathcal{P}$ iff there exists an abstract s-monoidal sequent $d : P_{\mathcal{P}} \xrightarrow[a]{id_0} Q_{\mathcal{P}}$ in $\mathbf{R}_{\mathcal{P}}$.* $\square$

Also for this result we use the fact that, for a given process algebra $\mathcal{P}$, then the effect of a rule can never be an identity.

References

[ABV94] L. Aceto, B. Bloom, F.W. Vaandrager, *Turning SOS Rules into Equations*, Information and Computation **111** (1), pp. 1-52.

[ADJ77] J.A. Goguen, J.W. Tatcher, E.G. Wagner, J.R. Wright, *Initial Algebra Semantics and Continuous Algebras*, Journal of the ACM **24** (1), 1977, pp. 68-95.

[BC91] G. Boudol, I. Castellani, *Flow Models of Distributed Computations: Three Equivalent Semantics for CCS*, INRIA Report 1484 (July 1991). To appear in Information and Computation.

[BD95] E. Badouel, P. Darondeau, *Trace Nets and Process Automata*, Acta Informatica **32** (7), 1995. pp. 647-679.

[BE74] A. Bastiani, C. Ehresmann *Multiple Functors I: Limits Relative to Double Categories*, Cahiers de Topologie ed Géométrie Différentielle **15** (3), 1974, pp. 545-621.

[BK84] J.A. Bergstra, J.W. Klop, *Process Algebra for Synchronous Communication*, Information and Computation **60**, 1984, pp. 109-137.

[Bou85] G. Boudol, *Computational Semantics of Term Rewriting Systems*, in Algebraic Methods in Semantics, eds. M.Nivat and J. Reynolds, Cambridge University Press, 1985.

[Bur91] A. Burroni, *Higher Dimensional Word Problem*, in CTCS'91, LNCS 530, pp. 94–105.

[BW90] M. Barr, C. Wells, *Category Theory for Computing Science*, Prentice Hall Series in Computer Science, 1990.

[CG95] A. Corradini, F. Gadducci, *CPO Models for Infinite Term Rewriting*, in Proc. AMAST'95, LNCS 936, 1995, pp. 368–384.

[CGM95] A. Corradini, F. Gadducci, U. Montanari, *Relating Two Categorical Models of Concurrency*, in Proc. RTA'95, LNCS 914, 1995, pp. 225–240.

[CM92] A. Corradini, U. Montanari, *An Algebraic Semantics for Structured Transition Systems and its Application to Logic Programs*, Theoretical Computer Science **103**, 1992, pp. 51-106.

[Cor93] A. Corradini, *Term Rewriting in CT_Σ*, in Proc. TAPSOFT'93, LNCS 668, 1993, pp. 468–484.

[Cor96] A. Corradini, *Term Rewriting, in Parallel*, submitted.

[DDM88] P. Degano, R. De Nicola, U. Montanari, *A Distributed Operational Semantics For CCS Based On Condition/Event Systems*, Acta Informatica **26**, 1988, pp. 59-91.

[DeS85] R. De Simone *Higher Level Synchronizing Devices in MEIJE-SCCS*, Theoretical Computer Science **37** (3), pp. 245–267, 1985.

[DP93] R. Dawson, R. Paré, *General Associativity and General Composition for Double-Categories*, in Cahiers de Topologie et Géométrie Différentielle Catégoriques, **25** (1), 1993, pp. 57-79.

[EPS93] M.C.J.D. van Eekelen, M.J. Plasmeijer, M.R. Sleep, *Term Graph Rewriting, Theory and Practice*, John Wiley & Sons, 1993.

[FM90] G. Ferrari, U. Montanari, *Towards the Unification of Models for Concurrency*, in Proc. CAAP'90, LNCS 431, 1990, pp. 162-176.

[Gad96] F. Gadducci, *On the Algebraic Approach to Concurrent Term Rewriting*, Ph.D Thesis, TD 02/96, Università di Pisa, Pisa, 1996.

[GM84] U. Goltz, A. Mycroft *On the Relationships of CCS and Petri Nets*, in Proc. ICALP'84, LNCS 172, 1984, pp. 196-208.

[GM94] R. Gorrieri, U. Montanari, *On the Implementation of Concurrent Calculi in Net Calculi: Two Case Studies*, Theoretical Computer Science **141**, 1994, pp. 195-252.

[GM95] F. Gadducci, U. Montanari, *Enriched Categories as Models of Computations*, in *Proc. Fifth Italian Conference on Theoretical Computer Science, ICTCS'95*, World Scientific, 1996, pp. 1-24.

[GM96] F. Gadducci, U. Montanari, *SOS Contexts as Cells in Double-Categories*, draft.

[HL91] G. Huet, J.J. Lévy, *Computations in Orthogonal Rewriting Systems*, in Computational Logic, Essays in Honour of A. Robinson, eds. J.L. Lassez and G. Plotkin, MIT Press, 1991, chapter 11.

[Hoa85] C.A.R. Hoare, *Communicating Sequential Processes*, Prentice Hall, 1985.

[Jac93] B. Jacobs, *Semantics of Weakening and Contraction*, Annals of Pure and Applied Logic **69**, 1994, pp 73-106.

[Kel76] R. Keller, *Formal Verifications of Parallel Programs*, Communications of ACM **7**, 1976, pp. 371-384.

[Kel82] G.M. Kelly, *Basic Concepts of Enriched Category Theory*, London Mathematical Society, LN Series 64, 1982.

[Klo92] J.W. Klop, *Term Rewriting Systems*, in Handbook of Logic in Computer Science, Vol. I, eds. S. Abramsky, D. Gabbay, and T. Maibaum, Oxford University Press, 1992, pp. 1-116.

[KR77] A. Kock, G.E. Reyes, *Doctrines in Categorical Logic*, in Handbook of Mathematical Logic, ed. John Bairwise, North Holland, 1977, pp. 283-313.

[KS74] G.M. Kelly, R.H. Street, *Review of the Elements of 2-categories*, Lecture Notes in Mathematics 420, 1974, pp. 75-103.

[Laf95] J. Lafont, *Equational Reasoning with 2-dimensional Diagrams* in Term Rewriting, French Spring School of Theoretical Computer Science, Font-Romeu, May 1993, LNCS 909, 1995, pp. 170-195.

[Law63] F. W. Lawvere, *Functorial Semantics of Algebraic Theories*, Proc. National Academy of Science **50**, 1963, pp. 869-872.

[LX90] K.G. Larsen, L. Xinxin, *Compositionality Through an Operational Semantics of Contexts*, in Proc. ICALP'90, LNCS 443, 1990, pp. 526-539.

[Mes90] J. Meseguer, *Rewriting as a Unified Model of Concurrency*, SRI Technical Report, CSL-93-02R, 1990. See in particular the appendix on *Functorial Semantics of Rewrite Systems.*

[Mes92] J. Meseguer, *Conditional Rewriting Logic as a Unified Model of Concurrency*, in Selected Papers of 2th Workshop on Concurrency and Compositionality, Theoretical Computer Science **96**, 1992, pp. 73-155.

[Mil89] R. Milner, *Communication and Concurrency*, Prentice-Hall, 1989.

[ML71] S. MacLane, *Categories for the Working Mathematician*, Springer, 1971.

[Old87] E.-R. Olderog, *Operational Petri Net Semantics for CCSP*, in Advances in Petri Nets 1987, LNCS 266, 1987, pp. 196-223.

[Plo81] G. Plotkin, *A Structural Approach to Operational Semantics*, Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[Pow89] A.J. Power, *An Abstract Formulation for Rewrite Systems*, in Proc. CTCS'89, LNCS 389, 1989, pp. 300-312.

[Rei85] W. Reisig, *Petri Nets*, Springer-Verlag, 1985.

[RS87] D.E. Rydeheard and J.G. Stell, *Foundations of Equational Deduction: A Categorical Treatment of Equational Proofs and Unification Algorithm*, in Proc. CTCS'87, LNCS 283, 1987, pp. 114-339.

[Str92] R.H. Street, *Categorical Structures*, in Handbook of Algebra, eds M. Hazewinkel et al., Elsevier, preprint 1992.

[Win82] G. Winskel *Event Structures for CCS and Related Languages*, in Proc. ICALP'82, LNCS 140, 1982, pp. 561-576.

Appendix: Some Categorical Preliminaries

In the first part we briefly recall some results on cartesian and monoidal categories: for a good presentation of the basic notions of category theory used in the paper we refer the interested reader to [ML71]. The rest of the section is devoted to provide an introduction to *double-categories*, not-so-common a device in the computer science community.

6.2 From Monoidal to Cartesian

We recollect some notions we already introduced on *graphs*, extending them with the definition of *graph morphism.*

Definition 42 (Graphs and Graph Morphisms). A *graph* G is a 4-tuple $\langle O_G, A_G, \delta_0, \delta_1 \rangle$: O_G, A_G are classes whose elements are called respectively *objects* and *arrows* (ranged over by $a, b, \ldots$ and $f, g, \ldots$), and $\delta_0, \delta_1 : A_G \to O_G$ are functions, called respectively *source* and *target.* A *graph morphism* $\tau : G_1 \to G_2$ between graphs G_1, G_2 is a couple of functions $\tau_A : A_1 \to A_2$, $\tau_O : O_1 \to O_2$ preserving source and target. A graph G is *reflexive* if there exist 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 O_G of objects forms a monoid; it is *monoidal* if it is reflexive and also its class of arrows forms a monoid, such that, if 0 is the neutral eleemnt of O_G, then $id(0)$ is the neutral element of A_G. A graph morphism $\tau : G_1 \to G_2$ between reflexive (with pairing, monoidal) graphs G_1, G_2 is *reflexive (pairing preserving, monoidal)* if its components preserve identity (the monoidal operator, both). ∎

Given a graph $G = \langle O_G, A_G, \delta_0, \delta_1 \rangle$, let $G \times_0 G = \langle O_G, A'_G, \delta'_0, \delta'_1 \rangle$ be the graph whose nodes are the same of G, while its arcs are the pairs $\langle f, g \rangle$ of *pairwise compatible* arcs of G, i.e., the subset of $A_G \times A_G$ such that $\delta_1(f) = \delta_0(g)$, and $\delta'_0(\langle f, g \rangle) = \delta_0(f), \delta'_1(\langle f, g \rangle) = \delta_1(g)$. A *category* $\mathbf{C}$ is a pair $\langle C, ; \rangle$, where C is a reflexive graph, called the *underlying graph* of $\mathbf{C}$, and the *composition* operator ; is a reflexive graph morphism $; : C \times_0 C \to C$: ; is associative, and moreover $id(a); f = f = id(b)$ for all $f : a \to b \in C$. A functor $F : \mathbf{C} \to \mathbf{D}$ is instead a reflexive graph morphism $F : C \to D$ preserving composition. Now we define the notion of *transformation*, an intuitive (and quite standard) generalization of the well-known notion of *natural transformation*.

Definition 43 ((Natural) Transformations). Given two functors $F, G : \mathbf{C} \to \mathbf{D}$, a *transformation* $\eta : F \Rightarrow G$ is a function $\alpha : O_C \to A_D$ such that $\eta(a) \in \mathbf{D}[F_O(a), G_O(a)]$; it is *natural* if for all $f : a \to b \in A_C$ the following diagram commutes:

$$\begin{tikzcd} F(a) \arrow[r, "\eta(a)"] \arrow[d, "F(f)"'] & G(a) \arrow[d, "G(f)"] \\ F(b) \arrow[r, "\eta(b)"'] & G(b) \end{tikzcd}$$

∎

One of the main features of categories is the possibility to define *universal constructions*: i.e., to characterize some elements as those satisfying "in a unique way" suitable properties.

Definition 44 (Products and Terminal Objects). Let $\mathbf{C}$ be a category. An object 1 in $\mathbf{C}$ is *terminal* if for every object a in $\mathbf{C}$ there exists a unique arrow $!_a : a \to 1$. The *product* of a pair $\langle a, b \rangle$ of objects in $\mathbf{C}$ is a triple $\pi_{a,b} = \langle a \times b, \pi_0 : a \times b \to a, \pi_1 : a \times b \to b \rangle$ such that, given the pair of arrows $\langle f : c \to a, g : c \to b \rangle$, there is a unique arrow $\langle f, g \rangle : c \to a \times b$ making the following diagram to commute:

$$\begin{tikzcd} & c \arrow[dl, "f"'] \arrow[d, "\langle f,g \rangle"] \arrow[dr, "g"] & \\ a & a \times b \arrow[l, "\pi_0"] \arrow[r, "\pi_1"'] & b \end{tikzcd}$$

∎

Usually, however, the *universal property* characterizes an object only *up-to-isomorphism*: for example, there may exist many elements satisfying the terminal object property, but they are isomorphic by a canonical isomorphism. A universal construction is *strict* if it is uniquely determined, in the sense that the canonical isomorphism is actually an identity; more generally, we say that a universal construction is *on-the-nose* if each class of elements satisfying the conditions of the universal construction by a canonical representative.

Definition 45 (Cartesian Categories). A category $\mathbf{C}$ is *cartesian* if it has terminal object 1 and product $\pi_{a,b}$ for each pair of objects a, b. A functor $F : \mathbf{C} \to \mathbf{D}$ between cartesian categories is *cartesian* if preserves products and terminal objects; i.e., if the canonical morphisms $\langle F(\pi_0), F(\pi_1)\rangle : F(a \times_C b) \to F(a) \times_D F(b)$ and $!_{F(1_C)} : F(1_C) \to 1_D$ are isomorphisms. ∎

Given $\mathbf{C}$, $\mathbf{D}$ categories, their *product category* $\mathbf{C} \times \mathbf{D}$ has as objects pairs $\langle c, d\rangle$ where $c \in O_C, d \in O_D$, as arrows pairs $\langle f, g\rangle$ where $f \in A_C, g \in A_D$, and the various operators are defined pointwise.

Definition 46 (Symmetric Monoidal Categories). A *monoidal category*[3] $\mathbf{C}$ is a triple $\langle \mathbf{C}_0, \otimes, e\rangle$ where $\mathbf{C}_0$ is a category, $e \in \mathbf{C}_0$ and $\otimes : \mathbf{C}_0 \times \mathbf{C}_0 \to \mathbf{C}_0$ is a functor, satisfying the coherence axioms: $(f \otimes g) \otimes h = f \otimes (g \otimes h)$ and $f \otimes e = e \otimes f = f$ for all arrows $f, g, h \in \mathbf{C}_0$ (where the identity of e is denoted with the object itself). A *symmetric monoidal category* is a 4-tuple $\langle \mathbf{C}_0, \otimes, e, \rho\rangle$ where $\langle \mathbf{C}_0, \otimes, e\rangle$ is a monoidal category, and $\rho : - \otimes + \Rightarrow + \otimes -$ is a natural isomorphism such that $\rho(e, a) = a$ for all $a \in \mathbf{C}_0$, and satisfying the axioms

$$\begin{tikzcd} a \otimes b \otimes c \arrow[r, "a \otimes \rho(b,c)"] \arrow[dr, "\rho(a \otimes b, c)"'] & a \otimes c \otimes b \arrow[d, "\rho(a,c) \otimes b"] \\ & c \otimes a \otimes b \end{tikzcd} \qquad \begin{tikzcd} a \otimes b \arrow[r, "\rho(a,b)"] \arrow[dr, "a \otimes b"'] & b \otimes a \arrow[d, "\rho(b,a)"] \\ & a \otimes b \end{tikzcd}$$

A *monoidal functor* $F : \mathbf{C} \to \mathbf{C}'$ is a pair $\langle F_0, \phi\rangle$ where $F_0 : \mathbf{C}_0 \to \mathbf{C}'_0$ is a functor such that $F_0(e) = e'$, while $\phi : F_0(- \otimes +) \Rightarrow F_0(-) \otimes' F_0(+)$ is a natural isomorphism such that $\phi(e, a) = \phi(a, e) = a$ and satisfying:

$$\begin{tikzcd} F(a \otimes b \otimes c) \arrow[r, "\phi(a, b \otimes c)"] \arrow[d, "\phi(a \otimes b, c)"'] & F(a) \otimes' F(b \otimes c) \arrow[d, "F(a) \otimes' \phi(b,c)"] \\ F(a \otimes b) \otimes' F(c) \arrow[r, "\phi(a,b) \otimes' F(c)"'] & F(a) \otimes' F(b) \otimes' F(c) \end{tikzcd}$$

(where we omitted subscripts for the sake of readability). A monoidal functor between symmetric monoidal categories is *symmetric* if moreover:

[3] Actually, we are defining here what are called *strict* monoidal categories in the literature. We allowed ourselves some freedom, in order to ease the notation.

$$\begin{tikzcd}
F(a \otimes b) \arrow[r, "\phi(a,b)"] \arrow[d, "F(\rho(a,b))"'] & F(a) \otimes' F(b) \arrow[d, "\rho'(F(a),F(b))"] \\
F(b \otimes a) \arrow[r, "\phi(b,a)"'] & F(b) \otimes' F(a)
\end{tikzcd}$$

A *monoidal (natural) transformation* $\alpha : F \Rightarrow G$ between monoidal functors is a (natural) transformation such that $\alpha(e) = e$ and satisfying:

$$\begin{tikzcd}
F(a \otimes b) \arrow[r, "\phi(a,b)"] \arrow[d, "\alpha(a\otimes b)"'] & F(a) \otimes' F(b) \arrow[d, "\alpha(a)\otimes'\alpha(b)"] \\
G(a \otimes b) \arrow[r, "\phi'(a,b)"'] & G(a) \otimes' G(b)
\end{tikzcd}$$

It is *symmetric* between symmetric functors if it satisfies also

$$\begin{tikzcd}
F(a \otimes b) \arrow[r, "F(\rho(a,b))"] \arrow[d, "\alpha(a\otimes b)"'] & F(b \otimes a) \arrow[d, "\alpha(b\otimes a)"] \\
G(a \otimes b) \arrow[r, "G(\rho(a,b))"'] & G(b \otimes a)
\end{tikzcd}$$

$\square$

All the coherence properties required for monoidal functors and natural transformations can be summed up saying that, in the richer context, all the underlying definitions must preserve also the new relevant structure. In the following, a (symmetric) monoidal functor is usually indicated with the underlying (symmetric) functor whenever the natural isomorphism associated is an identity. Now we "enrich" monoidal categories, to fill the gap with cartesian categories.

Definition 47 (S-Monoidal Categories). An *s-monoidal category* $\mathbf{C}$ is a 6-tuple $\langle \mathbf{C}_0, \otimes, e, \rho, \nabla, ! \rangle$ where $\langle \mathbf{C}_0, \otimes, e, \rho \rangle$ is a symmetric monoidal category, and $\nabla : Id \Rightarrow \langle - \otimes -, - \otimes \rho(+,-) \otimes + \rangle$ and $! : Id \Rightarrow e$ are two symmetric monoidal transformations satisfying:

$$\begin{tikzcd}
a \arrow[r, "\nabla(a)"] \arrow[d, "\nabla(a)"'] & a \otimes a \arrow[d, "a\otimes\nabla(a)"] \\
a \otimes a \arrow[r, "\nabla(a)\otimes a"'] & a \otimes a \otimes a
\end{tikzcd}
\qquad
\begin{tikzcd}
a \arrow[r, "\nabla(a)"] \arrow[dr, "a"'] & a \otimes a \arrow[d, "a\otimes!(a)"] \\
 & a = a \otimes e
\end{tikzcd}
\qquad
\begin{tikzcd}
a \arrow[r, "\nabla(a)"] \arrow[dr, "\nabla(a)"'] & a \otimes a \arrow[d, "\rho(a,a)"] \\
 & a \otimes a
\end{tikzcd}$$

An *s-monoidal functor* is a symmetric monoidal functor such that $\nabla'(F(a)) = F(\nabla(a)); \phi(a,a)$ and $!(F(a)) = F(!(a))$. $\square$

The monoidality of the transformations amounts to say that $\nabla(a \otimes b) = (\nabla(a) \otimes \nabla(b)); (a \otimes \rho(b,a) \otimes b)$ and $!(a \otimes b) = !(a) \otimes !(b)$ hold. Different versions

of the following result seems to have become part of the categorical "folklore" in the last years (see e.g. [Bur91, Jac93, Laf95]).

Proposition 48 (Cartesianity as Enriched Monoidality). *Let* $\mathbf{C}$ *be the s-monoidal category* $\langle \mathbf{C}_0, \otimes, e, \rho, \nabla, ! \rangle$. *If both* ∇ *and* $!$ *are natural, then* $\mathbf{C}$ *is cartesian.* ∎

Using a non-standard notation, we say that an s-monoidal category such that both ∇ and $!$ are natural is a cartesian category with *finite products*. An s-monoidal functor between cartesian categories with finite products is cartesian.

We say that a category $\mathbf{C}$ is *reflective* inside a category $\mathbf{D}$ if $\mathbf{C}$ is a subcategory of $\mathbf{D}$, the inclusion functor has a left-adjoint and the co-unit is a natural isomorphism. A very particular case is represented by *forgetful* functors: i.e., those functors that "forgets" part of the structure of the source category: they usually have a left-adjoint that simply "adds" the relevant structure with a "free" construction. An intuitive example is represented by the inclusion functor of **Cat** in **Gr**, the category of small graphs and graph-morphisms: the left-adjoint simply adds the identity and the composition functions, requiring these functions to satisfy the axioms of categories. In the paper, a particular importance is played by the following chain of adjunctions:

$$\mathbf{GR}_p \rightleftarrows \mathbf{GR}_m \rightleftarrows \mathbf{M\text{-}Cat} \rightleftarrows \mathbf{SM\text{-}Cat} \rightleftarrows \mathbf{FC\text{-}Cat}$$

relating the category $\mathbf{GR}_p$ ($\mathbf{GR}_m$) of graphs with pairing (monoidal) and graph morphisms preserving pairing (monoidal), the category **M-Cat** (**SM-Cat**, **FC-Cat**) of monoidal (s-monoidal, cartesian with finite products) categories and monoidal (s-monoidal) functors.

6.3 Internal and Double-Categories

Basically, an *internal category* of $\mathbf{C}$ (see [BW90]; also denoted a *cat-object* in $\mathbf{C}$) $\mathbf{D}_C$ is just a category such that its classes of arrows and objects are objects of $\mathbf{C}$, while its composition, identity, source and target morphisms are arrows in $\mathbf{C}$. An *internal functor* $F : \mathbf{D}_C \to \mathbf{E}_C$ between categories internal to $\mathbf{C}$ is a pair of arrows $F_A : A_{D_C} \to A_{E_C}, F_O : O_{D_C} \to O_{E_C}$ between, respectively, the objects in $\mathbf{C}$ representing the classes of objects of $\mathbf{D}_C$ and $\mathbf{E}_C$, and the objects in $\mathbf{C}$ representing the classes of arrows of $\mathbf{D}_C$ and $\mathbf{E}_C$. As an example, a category internal to **Set**, the category of small sets and functions, is just a locally small category $\mathbf{C}$. An *internal transformation* $\eta : F \Rightarrow G : \mathbf{D}_C \to \mathbf{E}_C$ is an arrow $O_{D_C} \to A_{E_C}$, satisfying the requirement $\eta ;_C \delta_0 = F_O$ and $\eta ;_C \delta_1 = G_O$, where δ_0, δ_1 are arrows in $\mathbf{C}$ representing the source and target functions in $\mathbf{E}_C$, while $;_C$ is the composition function in $\mathbf{C}$.

Actually, the definition is slightly more involved: it is necessary, in order to correctly define categories internal to $\mathbf{C}$, that $\mathbf{C}$ has pullbacks, so that for the category $\mathbf{D}_C$ the composition operator $;_{D_C}$ is defined as an arrow in $\mathbf{C}$ from the pullback $A_{D_C} \times_0 A_{D_C}$ (along $\langle (\delta_1)_{D_C}, (\delta_0)_{D_C} \rangle$) to A_{D_C}. Then we can define the

vertical composition of two internal transformations $\eta : F \Rightarrow G, \gamma : G \Rightarrow H$ as the arrow $[\eta, \gamma];_C (;_{E_C})$, or informally as $\eta;_{E_C} \gamma$; we have *naturality* as the axiom $[F_A, (\delta_1)_{E_C};_C \eta];_C (;_{E_C}) = [(\delta_0)_{E_C};_C \eta, G_A];_C (;_{E_C})$; and so on. An interesting example is already given by a simple category like **Cpo**, the category of small CPO's and continuous functions, used in [CG95] to describe *continuous algebras* [ADJ77, Cor93]. Also a 2-category $\underline{\mathbf{C}}$ can be described as a internal object to **Cat**, such that the categories $\mathbf{O}_C$ and $\mathbf{A}_C$ have the same set of objects. In general, an internal object to **Cat** is a *double-category* (see [BE74]; also [DP93] for some recent results on *pasting*). It represents an intuitive generalization of a 2-category, and it admits the following *naïve* presentation, adapted from [KS74].

Definition 49 (Double-Categories). A double-category $\underline{\mathbf{C}}$ (also d-category) consists of a collection $\{a, b, c, \ldots\}$ of *objects*, or 0-cells, a collection $\{f, g, h, \ldots\}$ of *horizontal arrows*, or horizontal 1-cells, a collection $\{x, y, z, \ldots\}$ of *vertical arrows*, or vertical 1-cells, and a collection $\{\alpha, \beta, \gamma, \ldots\}$ of *double-cells*, also d-cells. Objects and horizontal arrows form a category, the *horizontal 1-category* $\mathbf{C}$, with identity id_a for each object a; also objects and vertical arrows form a category, the *vertical 1-category*, with identity id^a for each object a. D-cells are assigned *horizontal source* and *target* (which are vertical 1-cells), written as $\alpha : x \Rightarrow_h y$, and *vertical source* and *target* (which are horizontal 1-cells), written as $\alpha : f \Rightarrow_v g$; furthermore, these 1-cells must be *compatible*, i.e., they must 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, d-cells can be composed both vertically ($\alpha *_v \beta$) and horizontally ($\gamma *_h \delta$): given $\alpha : x \Rightarrow_h y$ and $\beta : y \Rightarrow_h z$, then

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

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

Under each of these laws d-cells form a category, the *horizontal category* $\mathbf{C}_h$ and the *vertical category* 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, then $(\alpha *_v \gamma) *_h (\beta *_v \delta) = (\alpha *_h \beta) *_v (\gamma *_h \delta)$;
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

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

   must coincide.

Given $\underline{\mathbf{C}}, \underline{\mathbf{D}}$ d-categories, a *d-functor* $F : \underline{\mathbf{C}} \to \underline{\mathbf{D}}$ is a 4-tuple of functions mapping objects to objects, horizontal (vertical) arrows to horizontal (vertical) arrows and d-cells to d-cells, preserving identities and compositions of all kinds. $\square$

We denote by **D-Cat** the category of d-categories and d-functors. It includes **2-Cat** as a subcategory, since each 2-category is a double-category such that the vertical 1-category is a discrete one, i.e., all its arrows are identities.

Since a d-category is a cat-object in **Cat**, a d-functor $F : \underline{\mathbf{C}} \to \underline{\mathbf{D}}$ can be equivalently defined as a triple $\langle F, F_h, F_c \rangle$ of functors, such that $F : \mathbf{C} \to \mathbf{D}$ is a functor between the horizontal 1-categories, $F_h : \mathbf{C}_h \to \mathbf{D}_h$ is a functor between the horizontal categories and $F_C : \mathbf{C}_h \times_0 \mathbf{C}_h \to \mathbf{D}_h \times_0 \mathbf{D}_h$ between their pullbacks, preserving vertical compositions and identities of all kind.

Let $F, G : \underline{\mathbf{C}} \to \underline{\mathbf{D}}$ be d-functors: a *d-transformation* $\eta : F \Rightarrow G$ is a functor $\eta : \mathbf{C} \to \mathbf{D}_h$ from the horizontal 1-category $\mathbf{C}$ to the horizontal category $\mathbf{D}_h$, satisfying $\eta; \delta_0 = F$ and $\eta; \delta_1 = G$, where ; is composition in **Cat**, and δ_0, δ_1 are functors from $\mathbf{D}_h$ to $\mathbf{D}$. A d-transformation is *natural* if it satisfies the intuitive naturality requirement. We say that a d-category is cartesian with finite products if it is a cat-object in **FC-Cat** — the category of small categories with finite products and s-monoidal functors — and the associated functors representing source, target, composition and identity strictly preserve products (hence, we are considering a suitable subcategory of **Cat(FC-Cat)**).

We give now an equivalent, yet more explicit definition of horizontal product. Let $\underline{\mathbf{C}}$ be a d-category: we say that $\underline{\mathbf{C}}$ is cartesian (with *horizontal products*) if $\mathbf{C}_h$ and $\mathbf{C}$ have products, and they preserve composition in both vertical categories. This requirement is equivalent to imposing the functoriality of horizontal product with respect to vertical composition[4]: that is, whenever the composite is well defined, then $\langle \alpha *_v \beta, \gamma *_v \delta \rangle = \langle \alpha, \gamma \rangle *_v \langle \beta, \delta \rangle$. For instance, let a, b and c, d be 0-cells such that their products in the horizontal 1-category are the triples $\langle a \times b, p_0, p_1 \rangle$ and $\langle c \times d, p'_0, p'_1 \rangle$: we say that two objects $x : a \to c$ and $y : b \to d$ in $\mathbf{C}_h$ have product $x \times y : \langle a \times b \to c \times d, \pi_0, \pi_1 \rangle$, where

$$\begin{tikzcd} a \times b \arrow[r, "p_0"] \arrow[d, "x \times y"'] \arrow[dr, phantom, "\pi_0"] & a \arrow[d, "x"] \\ c \times d \arrow[r, "p'_0"'] & c \end{tikzcd} \qquad \begin{tikzcd} a \times b \arrow[r, "p_1"] \arrow[d, "x \times y"'] \arrow[dr, phantom, "\pi_1"] & b \arrow[d, "y"] \\ c \times d \arrow[r, "p'_1"'] & d \end{tikzcd}$$

if, given any two double cells

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

then there exists a unique double-cell

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

such that $\alpha = \langle \alpha, \beta \rangle *_h \pi_0$, $\beta = \langle \alpha, \beta \rangle *_h \pi_1$ or, equivalently, such that the following identities hold:

[4] Or that both source and target functors between $\mathbf{C}_h$ and $\mathbf{C}$ are cartesian.

$$\alpha = \begin{tikzcd} e \arrow[r, "\langle f,h\rangle"] \arrow[d, "z"'] \arrow[dr, phantom, "\langle\alpha,\beta\rangle"] & a \times b \arrow[r, "p_0"] \arrow[d, "x \times y"] \arrow[dr, phantom, "\pi_0"] & a \arrow[d, "x"] \\ m \arrow[r, "\langle g,i\rangle"'] & c \times d \arrow[r, "p'_0"'] & c \end{tikzcd} \qquad \beta = \begin{tikzcd} e \arrow[r, "\langle f,h\rangle"] \arrow[d, "z"'] \arrow[dr, phantom, "\langle\alpha,\beta\rangle"] & a \times b \arrow[r, "p_1"] \arrow[d, "x \times y"] \arrow[dr, phantom, "\pi_1"] & b \arrow[d, "y"] \\ m \arrow[r, "\langle g,i\rangle"'] & c \times d \arrow[r, "p'_1"'] & d \end{tikzcd}$$

In the following, we will denote with **DFC-Cat** the category of d-categories with finite horizontal products, and s-monoidal d-functors.

Definition 50 (Double-Computads). A *d-computad* is a triple $\langle \mathbf{C}, \mathbf{C}_h, \tau \rangle$ such that $\mathbf{C}$, $\mathbf{C}_h$ are categories, and $\tau = \langle \tau_0, \tau_1, id_\tau \rangle$ is a triple of functors $\tau_i : \mathbf{C}_h \to \mathbf{C}, id_\tau : \mathbf{C} \to \mathbf{C}_h$ such that $id_\tau ; \tau_0 = id_\tau ; \tau_1 = id_C$. A *d-morphism* is a pair $\langle F, F_h \rangle$ of functors $F : \mathbf{C} \to \mathbf{D}$, $F_h : \mathbf{C}_h \to \mathbf{D}_h$ between the underlying categories, such that τ, τ_i are preserved. ∎

In other words, a *d-computad* is a reflexive graph internal to **Cat**. It can also be described in a slightly more restrictive way (but for our purposes there will be no loss of generality) as a triple $\langle \mathbf{C}, D, S \rangle$, where $\mathbf{C}$ is a category, D is a monoidal graph (and its set of nodes is the same as the set of objects of $\mathbf{C}$) and S is a set of d-cells, each of which has assigned two pairs of compatible arrows in $\mathbf{C}$ and D as *horizontal* and *vertical source* and *target*, respectively. Given two d-computads $\langle \mathbf{C}, D, S \rangle$ and $\langle \mathbf{C}', D', S' \rangle$, a *d-morphism* is a triple $\langle F, G, h \rangle$ such that $F : \mathbf{C} \to \mathbf{C}'$ is a functor, $G : D \to D'$ is a monoidal graph morphism, and $h : S \to S'$ is a function (such that if $\alpha : f \to g \in A_C$ and $\beta : x \to y \in A_D$, then $h(\alpha) : F(f) \to F(g) \in A_{C'}$ and $h(\beta) : G(x) \to G(y) \in A_{D'}$), preserving identities and compositions of all kind.

Let **D-Comp** be the category of d-computads and d-morphisms: there exists an obvious forgetful functor U_d : **D-Cat** $\to$ **D-Comp**, with a left-adjoint F_d. This adjoint composes the cells of a d-computad in all the possible ways, both horizontally and vertically, imposing further equalities in order to satisfy the axioms of a d-category. Of course, analogous adjoints exist also when d-computads have a more enriched structure. As an example, we indicate with **DFC-Comp** the category of d-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} : **DFC-Cat** $\to$ **DFC-Comp**, with a left-adjoint F_{df}. With $\langle U_{ds}, F_{ds} \rangle$ we indicate instead the corresponding adjoint pair between s-monoidal d-computads and s-monoidal d-categories.