Electronic Notes in Theoretical Computer Science 4 (1996)

Tiles, Rewriting Rules and CCS [1]

Fabio Gadducci and Ugo Montanari

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

Abstract

In [12] we introduced the *tile model*, a framework encompassing 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 and of concurrency theory, and our formalism recollects many properties of these sources. For example, it provides a compositional way to describe both the states and the sequences of transitions performed by a given system, stressing their distributed nature. Moreover, 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. In this work we narrow our scope, presenting a restricted version of our tile model and focussing our attention on its expressive power. To this aim, we recall the basic definitions of the *process algebras* paradigm [3,24], centering the paper on the recasting of this framework in our formalism.

1 Introduction

It is not an overstatement to say that, in recent 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* [16] 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

[1] Research supported in part by Progetto Speciale del CNR "Strumenti per la Specifica e la Verifica di Proprietà Critiche di Sistemi Concorrenti e Distribuiti"; and in part by the Information Technology Promotion Agency, Japan, as a part of the Industrial Science and Technology Frontier Program "New Models for Software Architecture" sponsored by NEDO (New Energy and Industrial Technology Development Organization).

transition relation needs to be inductively defined according to that structure. This is the case of formalisms like *Petri nets* [30], where a state is a multiset of basic components, and each of them may evolve simultaneously (i.e., *in parallel*); or *term rewriting systems* [17], where states are terms of a given algebra, and rewrites are freely obtained from a set of deduction rules. Furthermore, we may need to consider formalisms relying on the use of *synchronization* and *side-effects* in determining the actual behaviour of a system. Maybe, the most important breakthrough is represented by the so-called sos approach [28]: 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* [22], 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* [9,6], where, in order to give a faithful account of the spatial distribution of a system, also transitions are equipped with an algebraic structure.

In [12] we introduced the *tile model*, as an attempt to encompass the properties of the already mentioned formalisms. As it happened for *rewriting logic* [23], the underlying idea of the 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 basic features of the model, like its compositional and spatial properties. In particular, there are three composition rules. First, they allow 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 in 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, and the possibility of expressing restrictions and synchronization will be fundamental when applying our paradigm to the operational description of *distributed systems*.

The tile model also admits a sofisticated characterization by means of *double-categories* [19], structures that may be roughly described as the superposition of a *vertical* and a *horizontal* category. In [12] it is shown that, starting from a rewriting system, with a free contruction (by means of a suitable adjunction) a double-category can be obtained whose "arrows" are in a one-to-one correspondence with the sequents entailed by that system. This result generalizes the analogous property for term rewriting [29,5], and it underlines the wide applicability of our model [11]. Along this line, in this paper we decided to narrow our attention: instead of describing in full details our formalism, for which we refer the interested reader to [12], we aim at analyzing its expressive power. The focus of the paper, then, is the recasting of the *process algebras* paradigm [3,14,24] in our model, which can be considered as a real benchmark for any general framework (se e.g. [26]). In particular, we deal with a suitable case study, the *Calculus of Communicating Systems* (also CCS, [24]), considered as the standard representative of the paradigm. CCS offers a constructive way to describe *concurrent systems*, considered as structured entities (the *agents*) interacting by means of some synchronization mechanism. Each system is then defined as a term of an algebra over a set of process constructors: new systems are built from existing ones, on the assumption that algebraic operators represent basic features of a concurrent system. The structure over agents allows for an immediate definition of the operational semantics of the language by means of the SOS approach: the dynamic bahaviour of an agent is then described by a suitable *labelled transition system*, where each transition step is a triple $\langle s, \mu, t \rangle$, with μ the observation associated to the transition itself. Finally, a further abstraction is obtained with the associated notion of *bisimulation*: an equivalence over agents equating those with the same observable behaviour.

The paper has the following structure. In Section 2.1 we introduce a formalization of term algebras, providing a concrete description which underlines the assumptions implicitly made in the ordinary notion. In Section 2.2 we introduce our rewriting systems, equipping them with a logic that describes the classes of derivations entailed by a system using (possibly abstract) sequents. In Section 3 we recall the basic definitions of CCS, its operational semantics and the associated strong bisimulation equivalence, along with its finite axiomatization. Finally, in Section 4 we show how the process algebras paradigm can be recovered in our framework. In particular, in Section 4.1 we describe a rewriting system which faithfully recovers the ordinary SOS semantics of CCS; in Section 4.2 we introduce the notion of *tile bisimulation*, in order to recast a suitable notion of observational equivalence in our formalism: this enables us to recover also CCS bisimilarity; finally, in Section 4.3 we turn the finite axiomatization of bisimilarity in a confluent rewriting system, providing each class of bisimilar agents with a canonical representative.

2 A Summary of the Tile Model

In this section we describe the basic features of the *tile model*, within a presentation biased towards the *process algebras* framework we deal with in Sections 3 and 4. For a comprehensive introduction we refer the reader to [12].

2.1 Building States

We open this section recalling some definitions from graph theory, that will be used to introduce *algebraic theories* [21,18]. Developed in the early Sixties, these theories received a lot of attention during the Seventies from computer scientists as a suitable characterization of the ordinary notion of term algebra.

Definition 2.1 (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* when equipped with 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 with pairing 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. $\square$

We can think of a signature Σ as a 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 $f \in \Sigma_n$. The usual notion of term can be formalized along this intuition, which allows to recover also alternative structures.

Definition 2.2 (graph theories). Given a signature Σ, the associated *graph theory* $G(\Sigma)$ 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:

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

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

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

Identities could be given 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 the theories. The solution we chose is equivalent, yet easier to describe, and it is used for all the auxiliary operators introduced in the next definitions.

Definition 2.3 (monoidal theories). Given a signature Σ, the associated *monoidal theory* $\mathbf{M}(\Sigma)$ 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:

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

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

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 $\underline{n}, \underline{m} \in \underline{\mathbb{N}}$. □

Further enriching the auxiliary structure, we are finally able to present the more expressive kind of theories we deal with in our paper, *algebraic theories.*

Definition 2.4 (algebraic theories). Given a signature Σ, the associated *algebraic theory* $\mathbf{A}(\Sigma)$ 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:

$$(\mathit{generators})\ \frac{f : \underline{n} \to \underline{1} \in \Sigma}{f : \underline{n} \to \underline{1} \in \mathbf{S}(\Sigma)} \qquad (\mathit{sum})\ \frac{s : \underline{n} \to \underline{m}, t : \underline{n'} \to \underline{m'}}{s \otimes t : \underline{n} \otimes \underline{n'} \to \underline{m} \otimes \underline{m'}}$$

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

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

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

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

$$id_{\underline{n} \otimes \underline{m}} = id_{\underline{n}} \otimes id_{\underline{m}} \qquad \rho_{\underline{n} \otimes \underline{m}, \underline{p}} = (id_{\underline{n}} \otimes \rho_{\underline{m},\underline{p}});(\rho_{\underline{n},\underline{p}} \otimes id_{\underline{m}})$$
$$!_{\underline{n} \otimes \underline{m}} = !_{\underline{n}} \otimes !_{\underline{m}} \qquad \nabla_{\underline{n} \otimes \underline{m}} = (\nabla_{\underline{n}} \otimes \nabla_{\underline{m}});(id_{\underline{n}} \otimes \rho_{\underline{n},\underline{m}} \otimes id_{\underline{m}})$$
$$!_{\underline{0}} = \nabla_{\underline{0}} = \rho_{\underline{0},\underline{0}} = id_{\underline{0}} \qquad \rho_{\underline{0},\underline{n}} = \rho_{\underline{n},\underline{0}} = id_{\underline{n}}$$

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* axioms

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

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

As for identities, also permutation and the other auxiliary operators could be inductively extended to all $\underline{n} \in \mathbb{N}$ starting from the basic cases, interpreting in a constructive way the monoidality axioms.

Let us consider the 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 $\mathbf{A}(\Sigma_{\sigma\epsilon})$ are $a_\Sigma; f_\Sigma : \underline{0} \rightarrow \underline{1}$, $f_\Sigma; g_\Sigma : \underline{1} \rightarrow \underline{1}$, $a_\Sigma; \nabla_{\underline{1}}; (f_\Sigma \otimes id_{\underline{1}}); h_\Sigma : \underline{0} \rightarrow \underline{1}$, intuitively corresponding to the terms $f(a), g(f(x))$ and $h(f(a), a)$, respectively, for a given variable x. In fact, a classical result we already anticipated proves that algebraic theories are equivalent to the ordinary construction (as it can be found e.g. in [2]) for term algebras.

Proposition 2.5 (algebraic theories and term algebras). *Let Σ be a signature. Then for all $\underline{n}, \underline{m} \in \mathbb{N}$ 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.* $\square$

The previous result states that each arrow $t_\Sigma : \underline{n} \rightarrow \underline{1}$ identifies an element t of the term algebra over the set $\{x_1, \ldots, x_n\}$: an arrow $\underline{n} \rightarrow \underline{m}$ is an m-tuple of such elements, and arrow composition is term substitution. Note that this correspondence *requires* that ∇ and $!$ are natural: if this were not the case, we get *s-monoidal theories* [10,12]. In these more concrete structures, such 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. In fact, in the PhD thesis [10] of the first author it is shown that a fundamental property of correspondence holds between s-monoidal theories and *term graphs* (as defined e.g. in the introductory chapter of [8]): each arrow $t_\Sigma : \underline{n} \rightarrow \underline{m}$ 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.

The incremental description of algebraic theories has received little attention in the literature (see [15,20]), despite the relevant fact that, differently from the usual categorical construction, 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 theories allow for a characterization of terms which is far more general, and at the same time more concrete, than the one allowed by the usual formalization 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 ordinary description (like *substitution*) implicitly enjoy. In this view, $!$ and ∇ represent respectively *garbage collection* and *sharing* (as discussed in [7,5]). As an example, let us consider the constant a: as a generator, the corresponding arrow is $a_\Sigma : \underline{0} \rightarrow \underline{1}$, while, when considered as an element of the term algebra over $\{x_1, x_2\}$, the associated arrow is $!_{\underline{2}}; a_\Sigma : \underline{2} \rightarrow \underline{1}$, 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 not play a relevant part, while instead s-monoidal theories hold a fundamental röle in [12], when dealing with *truly concurrent semantics* in the setting of *process algebras*.

2.2 Describing Systems

In this section we recall the basic formulation of our framework, inspired both from the *rewriting logic* approach by Meseguer [23] and the SOS approach by Plotkin [28]. Intuitively, an *algebraic rewriting system* is just a set of rules, each of them carrying information (i.e., expressing some conditions) on the possible behaviours of the terms to which they can be applied.

Definition 2.6 (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 signatures, N is a set of rule names and R is a function $R : N \to \mathbf{A}(\Sigma_\sigma) \times G(\Sigma_\tau) \times G(\Sigma_\tau) \times \mathbf{A}(\Sigma_\sigma)$ such that for all $d \in N$, 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 $d : l \xrightarrow[b]{a} r$. □

A *context system* [22] is just a very simple ARS, 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* [17], 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 2.5, they are just a very particular case of algebraic rewriting systems, where Σ_τ is empty: in the following, we will refer to these systems as *horizontal rewriting systems* (also HRS's), and a rule will be simply denoted as $d : l \longrightarrow r$. In fact, an HRS is what is called an *unconditional rewriting theory* in [23]. It is actually less general, since in this paper we decided to consider rewriting systems built over signatures Σ instead that over equational theories (Σ, E), even if the extension of ARS's to deal with them is quite straightforward.

Let us consider the 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

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

(where ι is a shorthand for the identitiy arrow $id_{\underline{0}} \in \mathbf{M}(\Sigma_{\tau\epsilon})$). 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.

An ARS $\mathcal{R}$ can be considered as a logical theory, and any rewriting –using rules in $\mathcal{R}$– as a sequent entailed by the 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. We say that *s rewrites to t via α* (using a *trigger* a and producing an *effect* b) if we obtain the sequent $\alpha : s \xrightarrow[b]{a} t$ by finitely many applications of a set of *inference rules.*

Definition 2.7 (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 the *algebraic sequents* $\alpha : s \xrightarrow[b]{a} t$ obtained by finitely many applications of the following inference rules:

basic rules

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

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

composition rules

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

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

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

auxiliary rules

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

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

□

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 for algebraic theories.

Let us consider the ARS $\mathcal{R}_e$ we previously defined. It entails the sequent

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

where ι is a shorthand for $id_{\underline{0}}$ and the entailment is described in a natural deduction style. It also entails the sequent

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

where the entailment is still described in a natural deduction style, but without using the rule names.

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 *computationally* equivalent derivations.

Definition 2.8 (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

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

(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 unconditional rewriting logic in [23]. Note also that, as already happened for the theories of Section 2.1, 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 $\mathcal{R}_e$, 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, even if the latter has a derivation unrelated to the one already shown for $(d \otimes (d * d_1)) * d_3$.

3 Operational Semantics for CCS

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* [3,14,24] 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. We present here one of the better known example of process algebra, the *Calculus of Communicating Systems* (CCS), introduced by Milner in the early Eighties [24], restricting ourselves, for the sake of exposition, to the case of *finite* CCS.

Definition 3.1 (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 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 $P, Q, R, \ldots$ range over the set *Proc* of processes. □

In the following, we indicate as Σ_{ccs} the signature associated with CCS processes (for example, *nil* is a constant, μ 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 3.2 (operational semantics of CCS. The ccs *transition system* is the relation $T_{ccs} \subseteq Proc \times Act \times Proc$ 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 $\langle P, \mu, Q\rangle \in T_{ccs}$. □

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* $\langle P, \mu, Q\rangle \in T_{ccs}$. As an example, to infer that from $P = (\alpha.nil + \beta.nil)||\overline{\alpha}.nil$ we can deduct $P \xrightarrow{\alpha} Q = nil||\overline{\alpha}.nil$, three different rules must be applied. Moreover, a process P can be rewritten into Q if there exists a *computation* from P to Q, i.e., a chain $P = P_0 \xrightarrow{\mu_1} P_1 \ldots P_{n-1} \xrightarrow{\mu_n} P_n = Q$ of one-step reductions.

The operational semantics we just defined is however too intensional, and more abstract semantics have been introduced by defining suitable *behavioural equivalences*, which identify processes exhibiting the same *observational behaviour*. Most of them are defined on the basic notion of *bisimulation* [27]: intuitively, two processes P, Q are *bisimilar* if, whenever P performs an action μ evolving to a state P', then also Q may execute that same action, evolving to a state Q' which is still bisimilar to P'.

Definition 3.3 (bisimulation equivalence). A symmetric equivalence relation $\sim_b \subseteq Proc \times Proc$ is a *bisimulation* if, whenever $P \sim_b Q$ for generic P, Q processes, then for any transition $P \xrightarrow{\mu} P'$ there exists a corresponding transition $Q \xrightarrow{\mu} Q'$ with $Q \sim_b Q'$. The maximal bisimulation equivalence is called *strong bisimulation*, and denoted by $\sim$. □

It is well-known that strong bisimilarity for ccs is also a congruence, and that it can be described by an equational theory over Σ_{ccs} [24]. In [3], the authors defined a *finitary* equational theory for an observational equivalence over their *Algebra of Communicating Processes*, introducing auxiliary operators. An obvious extension of their formalism can be adapted to get a finite description of strong bisimilarity for ccs, introducing three auxiliary operators, which intuitively split the parallel operator into three distinct cases, corresponding to left, right and synchronous composition of the sub-agents. On the other hand, Moller [25] has proved that bisimilarity cannot be finitely axiomatized without resorting to auxiliary operators. So, let Σ_{eccs} be the signature obtained extending Σ_{ccs} with the operators $\{\lfloor, \rfloor, | : 2 \to 1\}$.

Definition 3.4 (B-K axioms). Let P, Q be ECCS processes. The *Bergstra-Klop* (also *B-K*) *axiomatization* is given by the following axioms for the parallel, relabelling and restriction operators

$$P||Q = ((P\lfloor Q) + (P\rfloor Q)) + (P|Q);$$

$$(\mu.P)\lfloor Q = P\rfloor(\mu.Q) = \mu.(P||Q);$$

$$(\mu.P)|(\mu'.Q) = \begin{cases} \tau.(P||Q) & \text{if } \mu' = \overline{\mu} \text{ and } \mu \neq \tau, \\ nil & \text{otherwise;} \end{cases}$$

$$(\mu.P)\backslash_\alpha = \begin{cases} \mu.(P\backslash_\alpha) & \text{if } \mu \notin \{\alpha, \overline{\alpha}\}, \\ nil & \text{otherwise;} \end{cases}$$

$$(\mu.P)[\Phi] = \Phi(\mu).(P[\Phi]);$$

$$nil\lfloor P = P\rfloor nil = nil|P = P|nil = nil\backslash_\alpha = nil[\Phi] = nil;$$

extended with the *Hennessy-Milner* (also *H-M*) axioms for the choice operator

$$P + P = P + nil = P \quad P + Q = Q + P \quad (P + Q) + R = P + (Q + R).$$

We usually write $P \sim_{BK} Q$ if P and Q are in the same equivalence class with respect to the B-K axioms. □

The H-M axioms simply state the associativity, commutativity, identity and idempotency of the non-deterministic operator (see [13]). The importance of the B-K axioms is given by their soundness and completness with respect to the bisimulation equivalence, as stated in the following result. ¿From our point of view, however, equally relevant is the fact that these axioms can be easily turned into rewriting rules, obtaining a confluent rewriting system, that identifies bisimilar agents: more on this in the next section.

Proposition 3.5 (B-K axioms and strong bisimulation). *Let P, Q be* CCS *processes. Then $P \sim Q$ iff $P \sim_{BK} Q$.* □

4 Operational Semantics from Rewriting Systems

In this section we show how the CCS operational semantics can be recovered by suitable rewriting systems. In particular, in Section 4.1 we define an algebraic rewriting system $\mathcal{R}_{ccs}$ which faithfully corresponds to the CCS transition system T_{ccs}. Then, in Section 4.2 we define the notion of *tile bisimulation*, roughly identifying sequents with the same effect: when applied to the sequents entailed by $\mathcal{R}_{ccs}$, it provides a recasting of strong bisimilarity for CCS processes. Finally, in Section 4.3 we describe a horizontal rewriting system $\mathcal{R}_{BK}$, that derives, for each element of a class of bisimilar CCS processes, a canonical representative of the class itself.

4.1 Using Tiles for CCS

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

Definition 4.1 (the CCS rewriting system). The ARS $\mathcal{R}_{ccs}$ associated to CCS is the tuple $\langle \Sigma_{ccs}, A_{ccs}, N, R\rangle$, 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 process is actually "running", outputting the action μ. For example, the rule act_μ prefixes an idle process with the action μ, and then starts the execution, consuming that same action. There are also three rules dealing with the parallel operator: ξ_s synchronizes two running processes, while ξ_l and ξ_r perform an asynchronous move, taking a running and an idle process.

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 sequent

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

whose two-steps entailment is the following

$$\frac{\overline{id_{nil;\beta} : nil; \beta \xrightarrow[\iota]{\iota} nil; \beta} \quad \overline{act_\alpha : \alpha \xrightarrow[\alpha]{\iota} id_{\underline{1}}}}{id_{nil;\beta} * act_\alpha : nil; \beta; \alpha \xrightarrow[\alpha]{\iota} nil; \beta}$$

$$\frac{\overline{id_{nil} : nil \xrightarrow[\iota]{\iota} nil} \quad \overline{act_\beta : \beta \xrightarrow[\beta]{\iota} id_{\underline{1}}}}{id_{nil} * act_\beta : nil; \beta \xrightarrow[\beta]{\iota} nil}$$

(where ι is a shorthand for both $id_{\underline{0}}$ and $id_{\underline{1}}$, since no confusion can arise), showing the importance of effects in expressing the ordering constraints: P can execute α only if the underlying process $P' = \beta.nil$ is actually idle.

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

$$\dfrac{\dfrac{\dfrac{\dfrac{}{id_{nil} : nil \xrightarrow[\iota]{\iota} nil} \quad \dfrac{}{act_\alpha : \alpha \xrightarrow[\alpha]{\iota} id_{\underline{1}}}}{id_{nil} * act_\alpha : nil;\alpha \xrightarrow[\alpha]{\iota} nil} \quad \dfrac{}{res_\beta : \backslash_\beta \xrightarrow[\alpha]{\alpha} \backslash_\beta}}{(id_{nil} * act_\alpha) * res_\beta : nil;\alpha;\backslash_\beta \xrightarrow[\alpha]{\iota} nil;\backslash_\beta} \quad \dfrac{}{res_\gamma : \backslash_\gamma \xrightarrow[\alpha]{\alpha} \backslash_\gamma}}{((id_{nil} * act_\alpha) * res_\beta) * res_\gamma : nil;\alpha;\backslash_\beta;\backslash_\gamma \xrightarrow[\alpha]{\iota} nil;\backslash_\beta;\backslash_\gamma}$$

where the basic sequent act_α has been provided with a suitable context.

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 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 4.2 (computational correspondence). *Let P, Q be* ccs *agents, and P_R, Q_R the associated elements of $\mathbf{A}(\Sigma_{ccs})$. Then the transition $P \xrightarrow{\mu} Q$ is entailed by the* ccs *transition system T_{ccs} iff an abstract algebraic sequent $\alpha : P_R \xrightarrow[\mu]{id_{\underline{0}}} Q_R$ is entailed by $\mathbf{R}_{ccs}$.* □

The correspondence is instead one-to-one if we consider the restriction $\mathbf{R}_p$ of $\mathbf{R}_{ccs}$ over $\mathbf{A}(\Sigma_{ccs}) \times G(A_{ccs}) \times G(A_{ccs}) \times \mathbf{A}(\Sigma_{ccs})$: i.e., the relation obtained by dropping the proof terms from sequents. Or, equivalently, if we take into account the class of abstract sequents modulo the set of axioms E', obtained adding to E the axiom

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

Proposition 4.3 (interleaving correspondence). *Let P, Q be* ccs *agents, and P_R, Q_R the associated elements of $\mathbf{A}(\Sigma_{ccs})$. Then the transition $P \xrightarrow{\mu} Q$ is entailed by the* ccs *transition system T_{ccs} (i.e., $\langle P, \mu, Q\rangle \in T_{ccs}$) iff the abstract algebraic sequent (modulo the set of axioms E') $\alpha : P_R \xrightarrow[\mu]{id_{\underline{0}}} Q_R$ is entailed by $\mathbf{R}_{ccs}$ (i.e., $\langle P_R, id_{\underline{0}}, \mu, Q_R\rangle \in \mathbf{R}_p$).* □

4.2 Recovering Bisimulation for Tiles

It seems quite reasonable that the notion of bisimulation could be extended to deal with our framework. In this section we introduce *tile bisimulation*, showing its (intuitive) correspondence with strong bisimilarity for ccs processes.

Definition 4.4 (tile bisimulation). Let $\mathcal{R} = \langle \Sigma_\sigma, \Sigma_\tau, N, R\rangle$ be an ars. A symmetric equivalence relation $\equiv_b \subseteq \mathbf{A}(\Sigma_\sigma) \times \mathbf{A}(\Sigma_\sigma)$ is a *tile bisimulation* for $\mathcal{R}$ if, whenever $s \equiv_b t$ for generic s, t elements of $\mathbf{A}(\Sigma_\sigma)$, then for any abstract sequent $\alpha : s \xrightarrow[b]{a} s'$ entailed by $\mathcal{R}$ there exists a corresponding one $\beta : t \xrightarrow[b]{a} t'$ with $s' \equiv_b t'$. The maximal tile bisimulation equivalence is called *strong tile bisimulation*, and denoted by $\equiv_{st}$. □

This is an obvious generalization of Definition 3.3, due to the more concrete representation of states and the richer structure on effects shown by sequents with respect to ccs transitions. But of course there is a complete coincidence between bisimilarity over ccs processes and tile bisimilarity over the corresponding elements of $\mathbf{A}(\Sigma_{ccs})$.

Proposition 4.5 (bisimulation corrispondence). *Let P, Q be* ccs *agents, and P_R, Q_R the associated elements of $\mathbf{A}(\Sigma_{ccs})$. Then $P \sim Q$ iff $P_R \equiv_{st} Q_R$.* $\Box$

We need now to develop a concept analogous to congruence. Usually, an equivalence is a congruence whenever it preserves the operators. In our case, this "operator preserving" property can be restated in terms of parallel and horizontal composition.

Definition 4.6 (tile functoriality). Let $\mathcal{R} = \langle \Sigma_\sigma, \Sigma_\tau, N, R \rangle$ be an ars. A symmetric equivalence relation $\equiv_f \subseteq \mathbf{A}(\Sigma_\sigma) \times \mathbf{A}(\Sigma_\sigma)$ is *functorial* for $\mathcal{R}$ if, whenever $s \equiv_f t, s' \equiv_f t'$ for generic s, s', t, t' elements of $\mathbf{A}(\Sigma_\sigma)$, then $s; s' \equiv_f t; t'$ (whenever defined) and $s \otimes s' \equiv_f t \otimes t'$. $\Box$

It is not in general true that a tile bisimulation equivalence is also functorial. The following results provide a characterization of such a property in terms of *tile decomposition*.

Definition 4.7 (tile decomposition). Let $\mathcal{R}$ be an ars. We say that $\mathcal{R}$ verifies the *(tile) decomposition property* if *i*) whenever it entails an abstract sequent $\alpha : s; t \xrightarrow[b]{a} u$, then it entails also two sequents $\beta : s \xrightarrow[c]{a} s'$ and $\gamma : t \xrightarrow[b]{c} t'$ with $u = s'; t'$; and *ii*) whenever it entails an abstract sequent $\alpha : s \otimes t \xrightarrow[b]{a} u$, then it entails also two sequents $\beta : s \xrightarrow[b_1]{a_1} s'$ and $\gamma : t \xrightarrow[b_2]{a_2} t'$ with $u = s' \otimes t'$, $a = a_1 \otimes a_2$ and $b = b_1 \otimes b_2$. $\Box$

A very simple system not verifying the (tile) decomposition property is given by $\mathcal{R}_a = \langle \Sigma_{an}, \Sigma_a, N_a, R_a \rangle$, where $\Sigma_{an} = \{nil : 0 \to 1, a : 1 \to 1\}$, $\Sigma_a = \{a_1 : 1 \to 1, a_2 : 1 \to 1\}$ and

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

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

Proposition 4.8 (decomposition and bisimulation). *Let $\mathcal{R}$ be an* ars. *If it verifies the decomposition property, then the associated strong tile bisimulation is functorial.* $\Box$

The converse is not true. In fact, the tile bisimulation associated to the ars $\mathcal{R}_a$ is functorial, and it is freely generated from the basic classes $\{nil\}, \{id_{\underline{0}}\}, \{id_{\underline{1}}\}, \{a, a; a, \ldots\} = \{a^n | n \geq 1\}$, but the system does not verify the decomposition property. Note also the importance of $a_2 \in \Sigma_a$, which is responsible for the non-equivalence of $id_{\underline{1}}$ and a: on the contrary, that equivalence would have destroyed functoriality.

While it may be difficult to check out if a given rewriting system is "decomposable", the following proposition provides an easy syntactical property that implies decomposition.

Proposition 4.9 (basic source and decomposition). *Let $\mathcal{R}$ be the* ARS $\langle \Sigma_\sigma, \Sigma_\tau, N, R\rangle$ *such that, for all $d : s \xrightarrow[b]{a} t \in R$, $s \in \Sigma_\sigma$ (hence, the source is a basic operator). Then $\mathcal{R}$ satisfies the decomposition property.* □

Proof (sketch). The proof can be carried out in two steps.

First, each abstract sequent α can be decomposed into the vertical composition $\alpha_1 \cdot \alpha_2 \cdot \ldots \cdot \alpha_n$ of "concrete" sequents α_i such that the operator $\cdot$ does not appear in any of them. These kind of sequents are called *one-step*, and they can be obtained without using the *v-comp* rule.

Then, let us suppose that $\alpha : s \xrightarrow[b]{a} t$ is one-step. Now, since the source of each rule must be a basic operator, we have that the structure of α exactly mirrors the one of its source s. And since also the axioms of algebraic sequents mirror those of algebraic theories, the result holds. □

In fact, both $\mathcal{R}_{ccs}$ and $\mathcal{R}_e$ verify this "basic source" property, hence the decomposition one, so that the following corollary holds.

Corollary 4.10 (strong bisimulation is functorial). *The strong tile bisimulation $\equiv_{st}$ associated to $\mathcal{R}_{ccs}$ is functorial.*

Thanks to Proposition 4.5, this result implies that strong bisimilarity for CCS processes is also a congruence. In fact, if an equivalence is functorial it preserves contexts, and *a fortiori* also operators. As an example, let P, Q be CCS agents, P_R, Q_R the associated elements of $\mathbf{A}(\Sigma_{ccs})$, and let us assume that $P \sim Q$. Hence $P_R \equiv_{st} Q_R$ (also $Q_R \equiv_{st} P_R$ by symmetry) and, by functoriality, $(P_R \otimes Q_R); || \equiv_{st} (Q_R \otimes P_R); ||$, so that also $P||Q \sim Q||P$ holds.

In general, it should be worthy to identify suitable "formats" for the rules such that, given a rewriting system $\mathcal{R}$, then whenever its rules fit a format then $\mathcal{R}$ is decomposable. An analogous work has been done on process algebras: see e.g. [1] for more details on the so-called GSOS format. For our tile model, some preliminary considerations can be found in [12].

4.3 B-K Axioms as Rewriting Rules

The aim of this section is to show that the axiomatization given in Definition 3.4 can be turned into a horizontal rewriting system, which is adequate for bisimilarity, in the sense that, given two CCS processes, they are bisimilar iff they may evolve to the same element.

Definition 4.11 (B-K axioms as rewriting rules). The HRS $\mathcal{R}_{BK}$ associated to the B-K axioms is the tuple $\langle \Sigma_{eccs}, \emptyset, N_{BK}, R_{BK}\rangle$, with the following set of rules:

$$dec : || \longrightarrow \nabla_{\underline{2}}; ((\nabla_{\underline{2}}; (\lfloor \otimes \rfloor); +) \otimes |); +$$

$$\sigma_l : (\mu \otimes id_{\underline{1}}); \lfloor \longrightarrow ||; \mu \quad \sigma_{le} : (nil \otimes id_{\underline{1}}); \lfloor \longrightarrow !_{id_{\underline{1}}}; nil$$

$$\sigma_r : (id_{\underline{1}} \otimes \mu); \rfloor \longrightarrow ||; \mu \quad \sigma_{re} : (id_{\underline{1}} \otimes nil); \rfloor \longrightarrow !_{id_{\underline{1}}}; nil$$

$$\sigma : (\alpha \otimes \overline{\alpha}); | \longrightarrow ||; \tau \quad \sigma_e : (\mu \otimes \mu'); | \longrightarrow !_{\underline{2}}; nil \text{ for } \mu' \neq \overline{\mu} \text{ or } \mu = \tau$$

$$\sigma_{nl} : (nil \otimes id_{\underline{1}}); | \longrightarrow !_{\underline{1}}; nil \quad \sigma_{nr} : (id_{\underline{1}} \otimes nil); | \longrightarrow !_{\underline{1}}; nil$$

$$res_\alpha : \mu; \backslash_\alpha \longrightarrow \backslash_\alpha; \mu \text{ for } \mu \notin \{\alpha, \overline{\alpha}\}$$

$$res_e : \alpha;\backslash_\alpha \longrightarrow !_{\underline{1}};nil \qquad res_n : nil;\backslash_\alpha \longrightarrow nil$$
$$rel_\Phi : \mu;\Phi \longrightarrow \Phi;\Phi(\mu) \qquad rel_n : nil;\Phi \longrightarrow nil$$

together with the rules for for the choice operator

$$idem : \nabla_{\underline{1}};+ \longrightarrow id_{\underline{1}} \qquad idnil : (id_{\underline{1}} \otimes nil);+ \longrightarrow id_{\underline{1}}$$
$$comm : + \longrightarrow \rho_{\underline{1},\underline{1}};+ \qquad assoc : (id_{\underline{1}}\otimes;+);+ \longrightarrow (+ \otimes id_{\underline{1}});+$$

(where we omitted the subscripts for the sake of readibility). With $\mathcal{R}_{BBK}$ we denote the HRS obtained without the rules for the choice operator; with $\mathcal{R}_I$ the one containing only the rules *idem* and *idnil* and finally with $\mathcal{R}_{AC}$ the one containing only the rules *comm* and *assoc*. □

The system we defined is convergetnt but not terminating: it is well-known that the axioms for associativity and commutativity of an operator cannot be in general turned into terminating rules. In fact, it is easy to see that

$$+ \longrightarrow \rho_{\underline{1},\underline{1}};+ \longrightarrow \rho_{\underline{1},\underline{1}};\rho_{\underline{1},\underline{1}};+ = + \longrightarrow \rho_{\underline{1},\underline{1}};+ \longrightarrow \dots$$

However, let $\Sigma_{sccs} = \{nil, \mu, +\} \subseteq \Sigma_{ccs}$ be the signature of *sequential* CCS (SCCS) processes: next result shows that $\mathcal{R}_{BK}$ is still adequate with respect to strong bisimulation.

Proposition 4.12 (bisimulation as normal form, I). *Let P, Q be* ECCS *processes. Then $P \sim_{BK} Q$ iff there exists a* SCCS *process S such that $\mathcal{R}_{BK}$ entails two sequents $\alpha : P \longrightarrow S$ and $\beta : Q \longrightarrow S$.* □

Notice that the normalization procedure is totally orthogonal to the usual notion of transition in the SOS framework. In fact, let us consider the CCS processes $P = \alpha.\beta.nil$ and $Q = ((\alpha.nil)\backslash_\beta)\backslash_\gamma$: the associated computations evolving from them have been shown in the previous section. Note instead that, from a normalization point of view, P cannot move. Instead, Q *sequentially* executes two different *res* operations (one causally dependent from the other), and finally it evolves to $\alpha.nil$, as shown by the following sequents

$$\dfrac{\dfrac{\dfrac{}{id_{nil} : nil \longrightarrow nil} \quad \dfrac{}{res_\beta : \alpha;\backslash_\beta \longrightarrow \backslash_\beta;\alpha}}{id_{nil} * res_\beta : nil;\alpha;\backslash_\beta \longrightarrow nil;\backslash_\beta;\alpha} \quad \dfrac{}{id_{\backslash_\gamma} : \backslash_\gamma \longrightarrow \backslash_\gamma}}{(id_{nil} * res_\beta) * id_{\backslash_\gamma} : nil;\alpha;\backslash_\beta;\backslash_\gamma \longrightarrow nil;\backslash_\beta;\alpha;\backslash_\gamma}$$

$$\dfrac{\dfrac{}{id_{nil;\backslash_\beta} : nil;\backslash_\beta \longrightarrow nil;\backslash_\beta} \quad \dfrac{}{res : \alpha;\backslash_\gamma \longrightarrow \backslash_\gamma;\alpha}}{id_{nil;\backslash_\beta} * res_\gamma : nil;\backslash_\beta;\alpha;\backslash_\gamma \longrightarrow nil;\backslash_\beta;\backslash_\gamma;\alpha}$$

$$\dfrac{\dfrac{}{res_e : nil;\backslash_\beta \longrightarrow nil} \quad \dfrac{}{id_{\backslash_\gamma;\alpha} : \backslash_\gamma;\alpha \longrightarrow \backslash_\gamma;\alpha}}{res_e * id_{\backslash_\gamma;\alpha} : nil;\backslash_\beta;\backslash_\gamma;\alpha \longrightarrow nil;\backslash_\gamma;\alpha}$$

$$\dfrac{\dfrac{}{res_e : nil;\backslash_\gamma \longrightarrow nil} \quad \dfrac{}{id_\alpha : \alpha \longrightarrow \alpha}}{res_e * id_\alpha : nil;\backslash_\gamma;\alpha \longrightarrow nil;\alpha}$$

such that, by the axioms of Definition 2.8, we have

$$((id_{nil} * res_\beta) * id_{\backslash_\gamma}) \cdot (res_e * res_\gamma) \cdot (res_e * id_\alpha) : nil;\alpha;\backslash_\beta;\backslash_\gamma \longrightarrow nil;\alpha.$$

The above denotation of the abstract sequent in the example suggests a reduction where two steps are executed in parallel. Proposition 4.12 also has a stronger formulation, which is stated by the following result.

Proposition 4.13 (bisimulation as normal form, II). *Let P, Q be* eccs *processes. Then $P \sim_{BK} Q$ iff there exists a* sccs *process S such that the sequents $\alpha : P \longrightarrow_{BBK} P' \longrightarrow_{AC} P'' \longrightarrow_{I} S$ and $\beta : Q \longrightarrow_{BBK} Q' \longrightarrow_{AC} Q'' \longrightarrow_{I} S$ are entailed by $\mathcal{R}_{BK}$ (where $P \longrightarrow_{BBK} P'$ means that it is entailed by $\mathcal{R}_{BBK}$, and so on).* □

Since both $\mathcal{R}_{BBK}$ and $\mathcal{R}_I$ are terminating, then, if we considered an equational extension of our tile model, each class of bisimilar processes would be provided with a normal form, modulo associativity and commutativity.

References

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

[2] 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.

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

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

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

[6] 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.

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

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

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

[10] F. Gadducci, *On the Algebraic Approach to Concurrent Term Rewriting*, PhD Thesis, Università di Pisa, Pisa. Technical Report TD-96-02, Department of Computer Science, University of Pisa, 1996.

[11] 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.

[12] F. Gadducci, U. Montanari, *The Tile Model*, Technical Report TR-96-27, Department of Computer Science, University of Pisa, 1996.

[13] M. Hennessy, R. Milner, *Algebraic Laws for Nondeterminism and Concurrency*, Journal of the ACM **32** (1), 1985, pp. 137-161.

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

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

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

[17] 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.

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

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

[20] 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.

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

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

[23] J. Meseguer, *Conditional Rewriting Logic as a Unified Model of Concurrency*, Theoretical Computer Science **96**, 1992, pp. 73-155.

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

[25] F. Moller, *The Non-Existence of Finite Axiomatizations for CCS Congruences*, in Proc. LICS'90, IEEE Press, 1990, pp. 142-153.

[26] N. Martí-Oliet, J. Meseguer, *Rewriting Logic as a Logical and Semantic Framework*, SRI Technical Report, CSL-93-05, 1993.

[27] D. Park, *Concurrency and Automata on Infinite Sequences*, in Proc. Fifth G-I Conference, LNCS 104, 1981, pp. 167-183.

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

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

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