

Tile Transition Systems as Structured Coalgebras*

Andrea Corradini¹, Reiko Heckel², Ugo Montanari¹

¹ Dipartimento di Informatica, Università degli Studi di Pisa,
Corso Italia, 40, I - 56125 Pisa, Italia, {andrea, ugo}@di.unipi.it

² Universität GH Paderborn, FB 17 Mathematik und Informatik,
Warburger Str. 100, D-33098 Paderborn, Germany, reiko@uni-paderborn.de

Abstract. The aim of this paper is to investigate the relation between two models of concurrent systems: tile rewrite systems and coalgebras. Tiles are rewrite rules with side effects which are endowed with operations of parallel and sequential composition and synchronization. Their models can be described as monoidal double categories. Coalgebras can be considered, in a suitable mathematical setting, as dual to algebras. They can be used as models of dynamical systems with hidden states in order to study concepts of observational equivalence and bisimilarity in a more general setting.

In order to capture in the coalgebraic presentation the algebraic structure given by the composition operations on tiles, coalgebras have to be endowed with an algebraic structure as well. This leads to the concept of *structured coalgebras*, i.e., coalgebras for an endofunctor on a category of algebras.

However, structured coalgebras are more restrictive than tile models. Those models which can be presented as structured coalgebras are characterized by the so-called *horizontal decomposition property*, which, intuitively, requires that the behavior is compositional in the sense that all transitions from complex states can be derived by composing transitions out of component states.

1 Introduction

Tile logic [16, 26] relies on certain rewrite rules with side effects, called *basic tiles*, reminiscent of *SOS rules* [30] and *context systems* [21]. Related models are *structured transition systems* [11], as well as *rewriting logic* [22, 23] which extends to concurrent systems with state changes the body of theory developed within the algebraic semantics approach. Tile logic has been conceived with similar aims and similar algebraic structure as rewriting logic, and it extends rewriting logic

* Research partially supported by MURST project *Tecniche Formali per Sistemi Software*, by CNR Integrated Project *Metodi per Sistemi Connessi mediante Reti*, by TMR Network *GETGRATS* and by Esprit WGs *APPLIGRAPH*, *CONFER2* and *COORDINA*.

(in the unconditional case), since it takes into account state changes with side effects and synchronization.

We now briefly introduce tile logic. A tile A is a sequent which has the form:

$$A: s \xrightarrow[a]{a} s'$$

and states that the *initial configuration* s of the system evolves to the *final configuration* s' producing an *effect* b . However s is in general *open* (not closed) and the rewrite step producing the effect b is actually possible only if the sub-components of s also evolve producing the *trigger* a . Both trigger a and effect b are called *observations*, and model the interaction, during a computation, of the system being described with its environment. More precisely, both system configurations are equipped with an *input* and an *output interface*, and the trigger just describes the evolution of the input interface from its initial to its final configuration. Similarly for the effect. It is convenient to visualize a tile as a two-dimensional structure (see Figure 1), where the horizontal dimension corresponds to the extension of the system, while the vertical dimension corresponds to the extension of the computation. Actually, we should also imagine a third dimension (the thickness of the tile), which models parallelism: configurations, observations, interfaces and tiles themselves are all supposed to consist of several components in parallel.

The initial configuration of a tile A can also be called $\text{north}(A)$, and likewise $\text{south}(A)$, $\text{west}(A)$ and $\text{east}(A)$ stands for final configuration, trigger and effect, respectively.

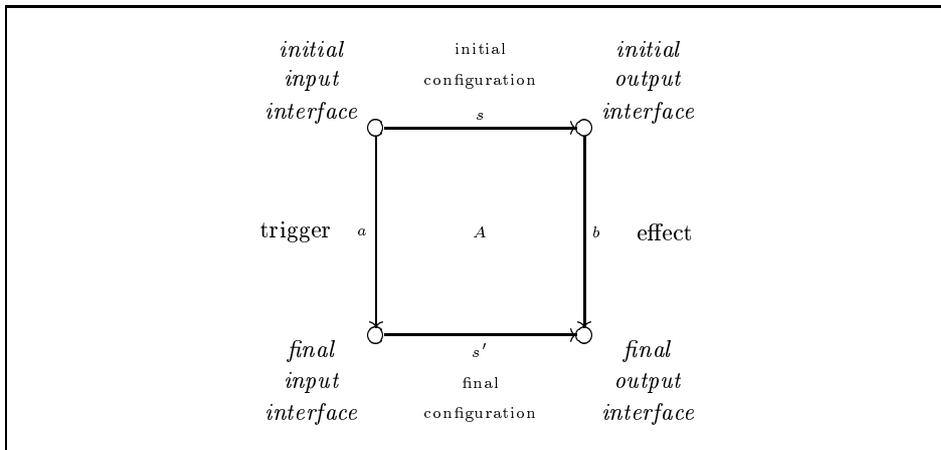


Fig. 1: A tile.

Configurations and observations are sometimes terms over a horizontal and a vertical signature, but for lots of applications to distributed systems it is convenient to employ various kinds of graphs, diagrams and charts [1]. Also suitable structural axioms can be imposed on terms or graphs.

Tiles are equipped with inference rules expressing three operations of composition: parallel ($-\otimes-$), horizontal ($-\ast-$), and vertical ($- \cdot -$) composition. Similarly, both configurations and observations are assumed to be equipped with operations of parallel and sequential composition, and interfaces with parallel composition only.

The operation of parallel composition is self explanatory. Vertical composition models sequential composition of transitions and computations. Horizontal composition corresponds to synchronization: the effect of the first tile acts as trigger of the second tile, and the resulting tile expresses the synchronized behavior of both.

In tile logic, a *tile rewrite system* provides signatures for configurations and observations and a set of *basic* tiles. Proofs start from basic tiles and apply the composition rules in all possible ways. The structure of tiles entailed in this way is specified by *proof terms*, built from the basic tiles used in the derivation and from the composition operations performed on them, up to certain structural axioms. Following the usual Curry-Howard analogy, proof terms *are* tiles, and their horizontal and vertical sources and targets are their *types*. However, quite often, proof terms are not relevant, and thus are omitted, or equivalently an additional normalizing axiom is introduced stating that two proof terms are the same whenever they have the same sources and targets. This is the case throughout in the paper.

Tile models are monoidal double categories. A *double category* [13] consists of four collections: objects, horizontal arrows, vertical arrows and cells, which correspond respectively to interfaces, configurations, observations and tiles of tile logic. Horizontal arrows with objects form the horizontal 1-category, and cells with vertical arrows form the horizontal 2-category. Contemporary horizontal composition of horizontal arrows and cells corresponds to horizontal composition of tile sequents. Similarly for the vertical dimension. Monoidal double categories have an additional operation which applies to the four collections above and which on tile sequents corresponds to parallel composition. If the tile logic is *flat*, i.e., it forgets about proof terms, it is appropriate to consider only flat models, too. A double category is flat if all the cells with the same sources and targets are identified.

Tile rewrite systems are interpreted as computads, i.e., algebraic structures consisting again of objects, horizontal and vertical arrows, and cells, but where only 1-categories are defined. In the monoidal version of tile logic, configurations and observations are arrows of strict monoidal categories generated by two hyper-signatures. A more direct representation of configurations and observations relies on certain hypergraphs analogous to Petri sequential processes [1]. In the initial model, the tiles entailed by a tile rewrite system can now be interpreted as cells. More precisely, the initial model can be constructed as the monoidal double category freely generated by the computad corresponding to the tile rewrite system [26].

Additional operations and axioms can be imposed on proof terms, configurations and observations whenever extra structure is required. Correspondingly,

tile models are enriched and the construction of the initial models adapted. Symmetric monoidal, cartesian [3] and cartesian closed versions [4] of tiles have been defined. Moreover, different structures for configurations, observations and proof terms can be introduced to tailor the logic and the models to the specific needs of the applications. An expressive specification language for this purpose is *membership equational logic* [24, 26], which has also been used to map tile logic into rewriting logic for implementation purposes [2]. Of course these enriched tile models are still special cases of the basic models of monoidal double categories. Hence, the results of this paper apply.

As it should be clear from the informal introduction above, the main intended application area of tile logic are distributed, interactive, open systems. In fact, tiles allow to model directly distribution (via the use of graphs to represent configurations), interaction (via triggers and effects) and openness (via the instantiation of free variables in configurations). In addition, tiles introduce rich forms of compositionality and induction based on their operations. Examples of applications are CCS with localities [2], π -calculus [14] and coordination of distributed systems [29].

The operational semantics of interactive systems is usually given in terms of labeled transition systems, and their abstract semantics in terms of sets of traces, or up to bisimilarity. When compositionality is an issue, semantic equivalence must be shown to be a congruence with respect to composition operations. The labeled transition systems associated to tile models are straightforwardly defined: horizontal arrows are states, tiles are transitions and pairs (trigger, effect) are labels. *Tile bisimilarity* is then defined in the standard way. It is also natural to define *tile congruences* as those equivalences of states, i.e., of horizontal arrows, which are *functorial*, i.e., which preserve their monoidal structure (identities, parallel composition and sequential composition).

The problem of finding sufficient conditions on tile rewrite systems to ensure that bisimilarity is a congruence in the initial model has been considered in [16]. The problem is divided in two parts. First a semantic condition, called *decomposition property* is defined on tile models, and it is shown that decomposition implies that bisimilarity is a congruence. The horizontal decomposition property is defined as follows. Whenever the vertical source h (which is a horizontal arrow) of a cell A can be sequentially decomposed as $h = h_1; h_2$, then also the cell itself must be horizontally decomposable as $A = A_1 * A_2$, where h_1 and h_2 are the vertical sources of A_1 and A_2 respectively. A similar condition is required for parallel composition. Informally, decomposition means that, given any computation A of a system h , and any subsystem h' of h , a computation A' of h' should exist which is a subcomputation of A . The second part consists in providing a syntactic condition, called *basic source property*, on tile rewrite systems: all rewrite rules must have just signature operators (i.e., basic graphs) as initial configurations and effects. It is possible to see that the basic source property ensures the decomposition property in the initial model, provided that there are no structural axioms in the specification.

The aim of this paper is to recast tile models enjoying a slightly stronger decomposition property (which we call *reflective* decomposition) as *structured coalgebras*. The use of coalgebras for the specification of dynamical systems with a hidden state space is receiving more and more attention in the last years, as a valid alternative to algebraic methods based on observational equivalences [31]. Given an endofunctor F on a category \mathbf{C} , a coalgebra is an arrow $f: X \rightarrow F(X)$ of \mathbf{C} and a coalgebra morphism from f to f' is an arrow $h: X \rightarrow X'$ of \mathbf{C} with $h \circ f' = f \circ F(h)$. Under certain conditions on \mathbf{C} and F , a category of coalgebras admits a final object, which can be considered informally as the minimal realization of the union of all the coalgebras in the category.

Ordinary labeled transition systems (with finite or countable branching) can be represented as coalgebras for a suitable functor on \mathbf{Set} . Furthermore, the unique morphism to the final coalgebra induces an equivalence which turns out to be exactly bisimilarity. Thus a first (rather straightforward) result of this paper is to show that tile models, seen as transition systems, can be considered as coalgebras and that their bisimilarity can be derived coalgebraically.

However, this representation forgets about the algebraic structure on horizontal arrows, which are seen just as forming a family of sets. As a consequence, the property that bisimilarity is a congruence, which is essential for making abstract semantics compositional, is not reflected in the structure of the model.

The problem of integrating coalgebras and algebras obtaining a model equipped with both structures has been tackled in [32], and an alternative but equivalent approach based on *structured coalgebras* is presented in [8, 10]. Here, the endofunctor determining the coalgebraic structure is lifted from \mathbf{Set} to the category of Γ -algebras, for some algebraic theory Γ . Morphisms between coalgebras in this category are both Γ -homomorphisms and coalgebra morphisms, and thus the unique morphism to the final coalgebra, which always exists, induces a (coarsest) bisimulation congruence on any coalgebra.

The second result of this paper is to show that by taking as Γ the theory of monoidal categories, a necessary and sufficient condition for the lifting to occur is reflective horizontal decomposition. Thus we obtain in another way that decomposition implies bisimilarity to be a congruence. Reflective decomposition additionally requires that cells with vertical sources which are horizontal identities must be identities for the horizontal composition of cells. It is easy to see that the basic source property implies also this extra condition for the initial model.

The paper is organized as follows. After introducing monoidal double categories in Section 2, Section 3 defines tile rewrite systems and their models, bisimilarity and functoriality, as well as the horizontal decomposition and basic source properties. Section 4 provides the necessary background on structured coalgebras. Section 5 presents tile transition systems as coalgebras over families of sets, while Section 6 provides the lifting of the algebraic structure. Finally in Section 7, as a case study, we show how the *calculus of communicating systems* (CCS) [27] can be recast in the tile framework as well as in terms of structured coalgebras.

2 Monoidal Double Categories

A *double category* is an internal category in the category of categories. Equivalently, double categories can be specified by a theory which is the tensor product¹ of the theory of categories with itself. The theory of *monoidal* double categories can be obtained as the tensor product of the theory of categories (twice) with the theory of monoids. Thus, one can argue that if the desired model of computation must have operations of parallel and horizontal composition (to build more parallel and larger systems) and of vertical composition (to build longer computations), then monoidal double categories are the most natural answer.

Here we give a more direct presentation of double categories, as done by Kelly and Street [19].

Definition 1 (double category). A double category \mathcal{D} consists of a collection a, b, c, \dots of objects, a collection h, g, f, \dots of horizontal arrows, a collection v, u, w, \dots of vertical arrows and a collection A, B, C, \dots of cells.

Objects and horizontal arrows form the horizontal 1-category \mathcal{H} (see Figure 2), with identity id^a for each object a , and composition $_*$.

$$a \xrightarrow{h} b \quad * \quad b \xrightarrow{g} c = a \xrightarrow{h * g} c \qquad a \xrightarrow{id^a} a$$

Fig. 2: Composition and identities in the horizontal 1-category.

Objects and vertical arrows form also a category, called the vertical 1-category \mathcal{V} (see Figure 3), with identity id_a for each object a , and composition \cdot (sometimes we will refer to both id^a and id_a either with the object name a or with id_a)

Cells are assigned horizontal source and target (which are vertical arrows) and vertical source and target (which are horizontal arrows); furthermore sources and targets must be compatible, in the sense that, given a cell A , with vertical source h , vertical target g , horizontal source v , and horizontal target u , then h and v have the same source, g and u have the same target, the target of h is equal to the source of u , and the target of v is equal to the source of g . These constraints can be represented by the diagram in Figure 4, for which we use the notation $A : h \xrightarrow[v]{u} g$.

In addition, cells can be composed both horizontally ($_*$) and vertically (\cdot) as follows: given $A : h \xrightarrow[v]{u} g$, $B : f \xrightarrow[w]{u} k$, and $C : g \xrightarrow[s]{z} h'$, then

¹ Tensor product (see for instance [20]) is a well-known construction for ordinary algebraic (Lawvere) theories. It can be extended to theories with partial operations (e.g. PMEqtl [26]) with essentially the same properties.

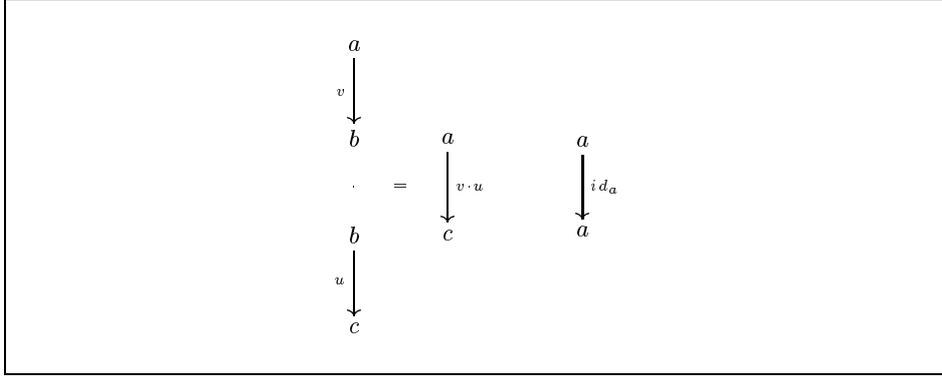


Fig. 3: Composition and identities in the vertical 1-category.

$A * B : (h * f) \xrightarrow{w} (g * k)$, and $A \cdot C : h \xrightarrow{v \cdot z} h'$ are cells. Both compositions can be pictured by pasting the diagrams in Figure 5. Moreover, given a fourth cell $D : k \xrightarrow{s} f'$, horizontal and vertical compositions verify the following exchange law (see also Figure 6):

$$(A \cdot C) * (B \cdot D) = (A * B) \cdot (C * D)$$

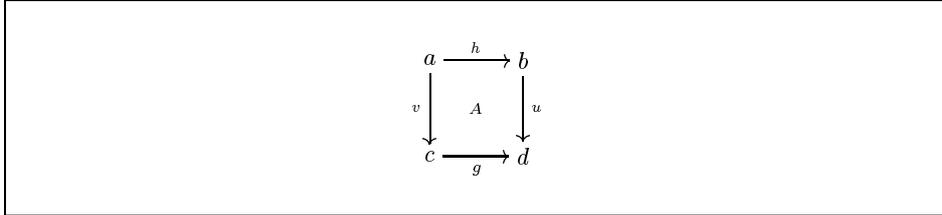


Fig. 4: Graphical representation of a cell.

Under these rules, cells form both a horizontal category \mathcal{D}^* and a vertical category \mathcal{D}' , with respective identities $1_v : a \xrightarrow{v} c$ and $1^h : h \xrightarrow{a} h$. Given $1^h : h \xrightarrow{a} h$ and $1^g : g \xrightarrow{b} g$, the equation $1^h * 1^g = 1^{h * g}$ must hold (and similarly for vertical composition of horizontal identities), as illustrated in Figure 7. Furthermore, horizontal and vertical identities of identities coincide, i.e., $1_{id_a} = 1^{id_a}$ and are denoted by the simpler notation 1_a (or just a).

A flat double category satisfies the additional condition that two cells with the same horizontal and vertical sources and targets are the same cell.

Remark 1. As a matter of notation, sometimes we will use \cdot ; \cdot to denote the composition on both the horizontal and vertical 1-categories.

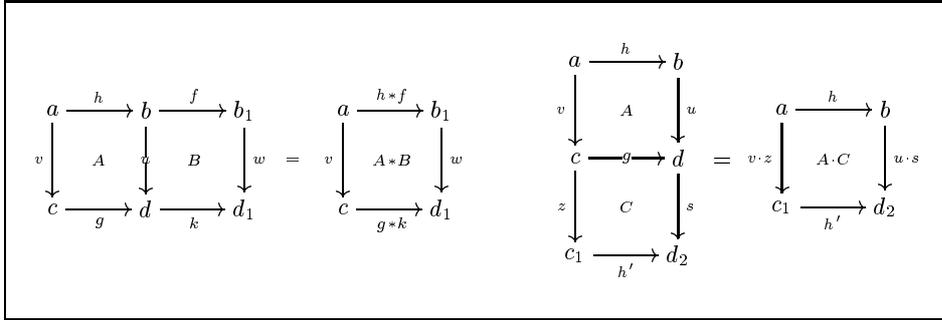


Fig. 5: Horizontal and vertical composition of cells.

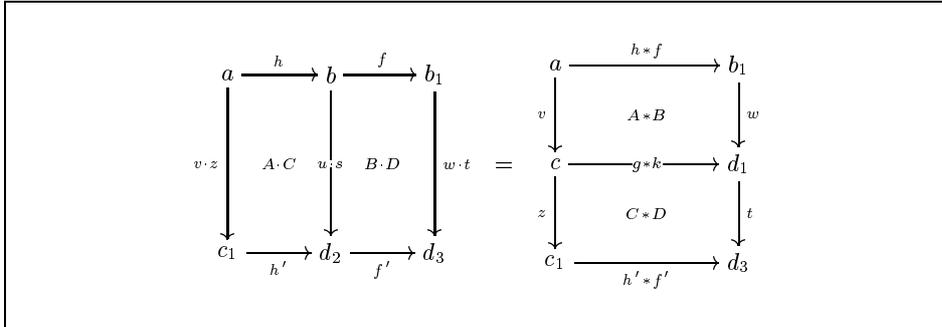


Fig. 6: The exchange law.

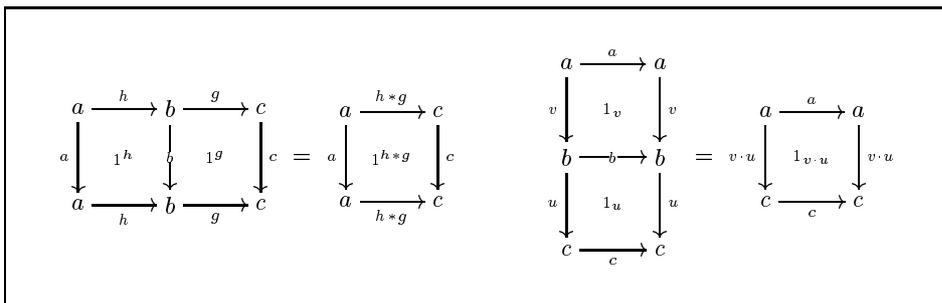


Fig. 7: Composition of identities.

Definition 2 (double functor). Given two double categories \mathcal{D} and \mathcal{E} , a double functor $F : \mathcal{D} \rightarrow \mathcal{E}$ is a 4-tuple of functions mapping objects to objects, horizontal and vertical arrows to horizontal and vertical arrows, and cells to cells, preserving identities and compositions of all kinds.

A monoidal double category, for the strict case, is defined as follows.

Definition 3 (strict monoidal double category). A strict monoidal double category, *sMD* in the following, is a triple $(\mathcal{D}, \otimes, e)$, where:

- \mathcal{D} is the underlying double category,
- $\otimes : \mathcal{D} \times \mathcal{D} \rightarrow \mathcal{D}$ is a double functor called the tensor product, and
- e is an object of \mathcal{D} called the unit object,

such that the following diagrams commute:

$$\begin{array}{ccc}
 \mathcal{D} \times \mathcal{D} \times \mathcal{D} & \xrightarrow{\otimes \times 1} & \mathcal{D} \times \mathcal{D} \\
 \downarrow 1 \times \otimes & & \downarrow \otimes \\
 \mathcal{D} \times \mathcal{D} & \xrightarrow{\otimes} & \mathcal{D}
 \end{array}
 \qquad
 \begin{array}{ccccc}
 \mathcal{D} & \xrightarrow{\langle 1, e \rangle} & \mathcal{D} \times \mathcal{D} & \xleftarrow{\langle e, 1 \rangle} & \mathcal{D} \\
 & \searrow 1 & \downarrow \otimes & \swarrow 1 & \\
 & & \mathcal{D} & &
 \end{array}$$

where double functor $1 : \mathcal{D} \rightarrow \mathcal{D}$ is the identity on \mathcal{D} , the double functor $e : \mathcal{D} \rightarrow \mathcal{D}$ (with some abuse of the notation) is the constant double functor which associates the object e and identities on e respectively to each object and each morphism/cell of \mathcal{D} , and $\langle _, _ \rangle$ denotes the pairing of double functors induced by the cartesian product of double categories. These equations state that the tensor product $_ \otimes _$ is associative on both objects, arrows and cells, and that e is the unit for $_ \otimes _$.

A monoidal double functor is a double functor which preserves tensor product and unit object. We denote by **fsMDCat** the category of flat monoidal double categories and monoidal double functors.

3 Tile Rewrite Systems

We now introduce (the flat monoidal versions of) tile rewrite systems and tile logic. Informally, a tile rewrite system is a set of double cells which, by horizontal, vertical and parallel composition, freely generate a monoidal double category. In the flat, monoidal version, the 1-categories of horizontal and vertical arrows are the strict monoidal categories freely generated by a horizontal and a vertical (hyper-) signature, which share the same set of sorts. The resulting monoidal double category is flat, i.e., two cells with the same horizontal and vertical source and target are identified.

Definition 4 (many-sorted hyper-signature). Given a set S of sorts, a (many-sorted, hyper) signature is an $S^* \times S^*$ -indexed family of sets $\Sigma = \{\Sigma_{n,m}\}_{(n,m) \in S^* \times S^*}$, where S^* denotes the free monoid on set S . Each $f \in \Sigma_{n,m}$ is denoted by $f : n \rightarrow m$.

Definition 5 (monoidal category freely generated by a signature). Given a signature Σ , $\mathbf{M}(\Sigma)$ is the strict monoidal category freely generated by Σ .

Definition 6 (tile rewrite systems). A monoidal tile rewrite system \mathcal{T} is a quadruple $\langle S, \Sigma_h, \Sigma_v, R \rangle$, where Σ_h, Σ_v are signatures on the same set S of sorts, and $R \subseteq \mathbf{M}(\Sigma_h) \times \mathbf{M}(\Sigma_v) \times \mathbf{M}(\Sigma_v) \times \mathbf{M}(\Sigma_h)$ is the set of rewrite rules, such that for all $\langle h, v, u, g \rangle \in R$, we have $h : n \rightarrow m, g : k \rightarrow l$ if and only if $v : n \rightarrow k, u : m \rightarrow l$.

For $\langle h, v, u, g \rangle \in R$ we use the notation $h \xrightarrow[v]{u} g$, or we depict it as a *tile*

$$\begin{array}{ccc} n & \xrightarrow{h} & m \\ v \downarrow & & \downarrow u \\ k & \xrightarrow{g} & l \end{array}$$

thus making explicit the source and target of each operator.

The rules of a tile rewrite system can be considered as its basic sequents. In the following, we say that h *rewrites to* g , using a *trigger* v and producing an *effect* u , if the (flat) sequent $h \xrightarrow[v]{u} g$ can be obtained by finitely many applications of certain inference rules.

Definition 7 (tile sequents). Let $\mathcal{T} = \langle S, \Sigma_h, \Sigma_v, R \rangle$ be a monoidal tile rewrite system. We say that \mathcal{T} entails the tile sequent $h \xrightarrow[v]{u} g$, written $\mathcal{T} \vdash h \xrightarrow[v]{u} g$, if and only if it can be obtained by a finite number of applications of the inference rules given in Table 1.

Basic rules provide the generators of the sequents, together with suitable identity arrows, whose intuitive meaning is that an element of $\mathbf{M}(\Sigma_h)$ may stay idle during a rewrite, showing no effect and using no trigger. Similarly, a horizontal identity may be rewritten to itself whenever trigger and effect are equal. Composition rules express the way in which sequents can be combined, either sequentially (*vert*), or executing them in parallel (*par*), or nesting one inside the other (*hor*).

It is easy to see that the tiles entailed by a tile rewrite system are the cells of a double category.

Proposition 1 (from tile rewrite systems to double categories). Given a monoidal tile rewrite system $\mathcal{T} = \langle S, \Sigma_h, \Sigma_v, R \rangle$, the flat monoidal double category $F_{\mathcal{T}}(\mathcal{T})$ has $\mathbf{M}(\Sigma_h)$ as horizontal 1-category, $\mathbf{M}(\Sigma_v)$ as vertical 1-category, and the flat tile sequents entailed by \mathcal{T} as double cells.

The models of a tile rewrite systems are defined as follows.

Basic Sequents. Generators and Identities:	
$(gen) \frac{h \xrightarrow{v} g \in R}{h \xrightarrow{u} g}$	
$(v-ref) \frac{v : n \rightarrow k \in \mathbf{M}(\Sigma_v)}{id_n \xrightarrow{v} id_k}$	$(h-ref) \frac{h : n \rightarrow m \in \mathbf{M}(\Sigma_h)}{h \xrightarrow{id_n} h}$
Composed Sequents. Parallel, Horizontal and Vertical compositions:	
$(vert) \frac{h \xrightarrow{v_1} f, f \xrightarrow{v_2} g}{h \xrightarrow{v_1; v_2} g}$	
$(par) \frac{h_1 \xrightarrow{v_1} g_1, h_2 \xrightarrow{v_2} g_2}{h_1 \otimes h_2 \xrightarrow{v_1 \otimes v_2} g_1 \otimes g_2}$	$(hor) \frac{h_1 \xrightarrow{v} g_1, h_2 \xrightarrow{w} g_2}{h_1; h_2 \xrightarrow{v} g_1; g_2}$

Table 1: Inference rules for monoidal tile sequents.

Proposition 2 (models of tile rewrite systems). *Given a tile rewrite system \mathcal{T} , its category of flat models is the comma category $(F_T(\mathcal{T}) \downarrow \mathbf{fsMDCat})$.*

Notice that the models are themselves double categories and that $F_T(\mathcal{T})$ is initial in the category of models of \mathcal{T} . However, in the following we will not be interested in morphisms relating only the models of a single tile rewrite system, since abstraction based on bisimilarity may relate several of them. Thus in the following we will consider *generic* models, i.e., just monoidal double categories.

We now introduce the ordinary notions of transition system and bisimilarity.

Definition 8 (labeled transition systems). *Let L be a fixed set of labels. A (nondeterministic) labeled transition system (over L), briefly *LTS*, is a structure $TS = \langle S, \longrightarrow_{TS} \rangle$, where S is a set of states, and $\longrightarrow_{TS} \subseteq S \times L \times S$ is a labeled transition relation. As usual, we write $s \xrightarrow{l} s'$ for $\langle s, l, s' \rangle \in \longrightarrow_{TS}$.*

A transition system morphism $f : TS \rightarrow TS'$ is a function $f : S \rightarrow S'$ which “preserves” the transitions, i.e., such that $s \xrightarrow{l} s'$ implies $f(s) \xrightarrow{l} f(s')$. We will denote by \mathbf{LTS}_L the category of LTS over L and corresponding morphisms.

Definition 9 (bisimilarity). *Given a LTS $TS = \langle S, \longrightarrow_{TS} \rangle$, an equivalence relation \equiv on S is a bisimulation if, whenever $s_1 \equiv s_2$, then for any transition $s_1 \xrightarrow{l} s'_1$ there exists a corresponding transition $s_2 \xrightarrow{l} s'_2$ with $s'_1 \equiv s'_2$. The maximal bisimulation is called bisimilarity, and denoted by \sim_{TS} .*

Operational bisimilarity of double categories is defined in the straightforward way hinted at in the Introduction.

Definition 10 (bisimilarity for double categories). Let \mathcal{D} be a double category, and let $TS_{\mathcal{D}}$ be the labeled transition system where labels are pairs (v, u) of vertical arrows, states are horizontal arrows h , and $h \xrightarrow{(v, u)}_{TS} g$ if and only if there is a cell $A: h \xrightarrow[v]{u} g$. Two horizontal arrows are operationally bisimilar, $h_1 \sim_{op} h_2$, iff $h_1 \sim_{TS_{\mathcal{D}}} h_2$.

Definition 11 (functorial equivalence relation). Let \mathcal{D} be a monoidal double category. An equivalence relation $h_1 \cong_{\mathbf{f}} h_2$ on the horizontal arrows is functorial for \mathcal{D} if, whenever $h \cong_{\mathbf{f}} g, h' \cong_{\mathbf{f}} g'$ for generic horizontal arrows h, h', g, g' , then $h; h' \cong_{\mathbf{f}} g; g'$ (whenever defined) and $h \otimes h' \cong_{\mathbf{f}} g \otimes g'$.

In other words, we are requiring that the *quotient category* of the horizontal 1-category of \mathcal{D} is well-defined, and it is monoidal. In general, it is not true that operational bisimilarity is also functorial. The following results (adapted from [16]) provide a characterization of such a property in terms of *horizontal decomposition*. The results hold for any monoidal double category, in particular for a flat one.

Definition 12 (horizontal tile decomposition). Let \mathcal{D} be a monoidal double category. We say that it is horizontally decomposable (or that it verifies the horizontal decomposition property) if

1. whenever there is a cell $A: h_1; h_2 \xrightarrow[v]{u} g$, then there are also cells $A_1: h_1 \xrightarrow[v]{u} g_1$ and $A_2: h_2 \xrightarrow[w]{u} g_2$ with $A = A_1 * A_2$;
2. whenever there is a cell $A: h_1 \otimes h_2 \xrightarrow[v]{u} g$, then there are also cells $A_1: h_1 \xrightarrow[v_1]{u_1} g_1$ and $A_2: h_2 \xrightarrow[v_2]{u_2} g_2$ with $A = A_1 \otimes A_2$.

Category \mathcal{D} verifies the reflective horizontal decomposition property if, in addition, for each cell $A: f \xrightarrow[v]{u} g$, $f = a$ implies $A = 1_v: a \xrightarrow[v]{v} a$, and $f = e$ implies $A = e: e \xrightarrow[e]{e} e$.

Proposition 3 (decomposition implies that bisimilarity is functorial). Let \mathcal{D} be a monoidal double category. If it verifies the horizontal decomposition property, then operational bisimilarity \sim_{op} is functorial.

The notions of bisimilarity, functoriality and (reflective) horizontal decomposition can be defined also for a tile rewrite system \mathcal{T} : it is enough to check if they hold for $F_{\mathcal{T}}(\mathcal{T})$. Notice in particular that for \mathcal{T} the operational bisimilarity \sim_{op} is a relation on configurations.

Given a tile rewrite system \mathcal{T} , it may be difficult to check if it is decomposable (and thus if its operational bisimilarity is functorial). We can provide a syntactical property of \mathcal{T} that implies reflective decomposition.

Proposition 4 (basic source property and decomposition). *Let $\mathcal{T} = \langle S, \Sigma_h, \Sigma_v, R \rangle$ be a tile rewrite system such that, for all $h \xrightarrow[u]{v} g \in R$, $h \in \Sigma_h$ and $u \in \Sigma_v$ (hence, both initial configuration and effect are just operators). Then $F_T(\mathcal{T})$ verifies the reflective horizontal decomposition property.*

4 Coalgebras and Structured Coalgebras

As recalled in the introduction, the use of coalgebras for the specification of dynamical systems with a hidden state space is receiving more and more attention in the last years, as a valid alternative to algebraic methods based on observational equivalences [31, 32].

In this section we first introduce the standard way to represent labeled transition systems as coalgebras for a suitable powerset functor [31], and then we discuss how this encoding can be lifted to a more structured framework, where the coalgebraic representation keeps the relevant algebraic structure of the states and transition of the encoded system. Let us start introducing the formal definition of coalgebra for a functor.

Definition 13 (coalgebras). *Let $B : \mathcal{C} \rightarrow \mathcal{C}$ be an endofunctor on a category \mathcal{C} . A coalgebra for B or B -coalgebra is a pair $\langle A, a \rangle$ where A is an object of \mathcal{C} and $a : A \rightarrow B(A)$ is an arrow. A B -cohomomorphism $f : \langle A, a \rangle \rightarrow \langle A', a' \rangle$ is an arrow $f : A \rightarrow A'$ of \mathcal{C} such that*

$$f; a' = a; B(f). \quad (1)$$

The category of B -coalgebras and B -cohomomorphisms will be denoted $B\text{-Coalg}$. The underlying functor $U : B\text{-Coalg} \rightarrow \mathcal{C}$ maps an object $\langle A, a \rangle$ to A and an arrow f to itself.

Let $P_L : \mathbf{Set} \rightarrow \mathbf{Set}$ be the functor defined as $X \mapsto \mathcal{P}(L \times X)$ where L is a fixed set of labels and \mathcal{P} denotes the powerset functor. Then coalgebras for this functor are one-to-one with labeled transition systems over L [31].

Proposition 5 (labeled transition systems as coalgebras). *Category $P_L\text{-Coalg}$ is isomorphic to the sub-category of \mathbf{LTS}_L containing all its objects, and all the morphisms $f : TS \rightarrow TS'$ which also “reflect” transitions, i.e., such that if $f(s) \xrightarrow{l}_{TS'} t$ then there is a state $s' \in S$ such that $s \xrightarrow{l}_{TS} s'$ and $f(s') = t$.*

It is instructive to spell out the correspondence just stated. For objects, a transition system $\langle S, \longrightarrow \rangle$ is mapped to the coalgebra $\langle S, \sigma \rangle$ where $\sigma(s) = \{\langle l, s' \rangle \mid s \xrightarrow{l} s'\}$, and, vice versa, a coalgebra $\langle S, \sigma : S \rightarrow P_L(S) \rangle$ is mapped to the system $\langle S, \longrightarrow \rangle$, with $s \xrightarrow{l} s'$ if $\langle l, s' \rangle \in \sigma(s)$. For arrows, by spelling out condition (1) for functor P_L , we get

$$\forall s \in S. \{\langle l, t \rangle \mid f(s) \xrightarrow{l} t\} = \{\langle l, f(s') \rangle \mid s \xrightarrow{l} s'\},$$

and by splitting this set equality in the conjunction of the two inclusions, one can easily see that inclusion “ \supseteq ” is equivalent to $s \xrightarrow{l} s' \Rightarrow f(s) \xrightarrow{l} f(s')$, showing that f is a transition system morphism, while the left-to-right inclusion is equivalent to $f(s) \xrightarrow{l} t \Rightarrow \exists s'. s \xrightarrow{l} s' \wedge f(s') = t$, meaning that f is a “zig-zag” morphism, i.e., that it reflects transitions.

The property of “reflecting behaviors” enjoyed by cohomomorphisms plays a fundamental rôle, for example, for the characterization of bisimulation relations as spans of cohomomorphisms, for the relevance of final coalgebras, and for various other results of the theory of coalgebras [31]. Given two coalgebras $\langle A, a \rangle$ and $\langle A', a' \rangle$, a *coalgebraic bisimulation* on them is a coalgebra $\langle A \times A', r \rangle$ having as carrier the cartesian product of the carriers, and such that the projections $\pi : A \times A' \rightarrow A$ and $\pi' : A \times A' \rightarrow A'$ are cohomomorphisms. Interestingly, it is easy to check that two states of a labeled transition system S are bisimilar (in the standard sense, see Definition 9) if and only if there is a coalgebraic bisimulation on S (regarded as a P_L -coalgebra) which relates them.

An even easier definition of categorical bisimilarity can be given if there exists a final coalgebra. In this case, two elements of the carrier of a coalgebra are bisimilar iff they are mapped to the same element of the final coalgebra by the unique cohomomorphism. Unfortunately, due to cardinality reasons, the functor P_L used for the coalgebraic representation of transition systems does not admit a final coalgebra [31]. One satisfactory, alternative solution consists of replacing the powerset functor \mathcal{P} on **Set** by the *countable* powerset functor \mathcal{P}_c , which maps a set to the family of its countable subsets. Then defining the functor $P_L^c : \mathbf{Set} \rightarrow \mathbf{Set}$ by $X \mapsto \mathcal{P}_c(L \times X)$ one has that coalgebras for this endofunctor are in one-to-one correspondence with transition systems with *countable degree*, i.e., systems where for each state $s \in S$ the set $\{\langle s', l \rangle \mid s \xrightarrow{l} s'\}$ is countable, the correspondence being defined exactly as in Proposition 5. Unlike functor P_L , the functor P_L^c admits cofree and final coalgebras.

Proposition 6 (final and cofree P_L^c -coalgebras). *The obvious underlying functor $U : P_L^c\text{-Coalg} \rightarrow \mathbf{Set}$ has a right adjoint $R : \mathbf{Set} \rightarrow P_L^c\text{-Coalg}$ associating with each set X a cofree coalgebra over X . As a consequence, the category $P_L^c\text{-Coalg}$ has a final object, which is the cofree coalgebra $R(\mathbf{1})$ over a final set $\mathbf{1}$.*

We shall stick to this functor throughout the rest of the paper, and since there is no room for confusion the superscript c will be understood.

Often transition systems come equipped with some algebraic structure on states, transitions, and/or labels, which plays a relevant role in the corresponding theory. For example, in calculi of the family of process algebras, like CCS [27] and the π -calculus [28], the agents (states) are closed under certain operations that can be interpreted as either structural (like parallel composition) or behavioural (like prefixing and nondeterministic choice). The same algebraic structure can be extended to the collection of transitions, in a way that is determined by the SOS rules which specify the operational semantics of the calculus. This more structured framework makes it possible to investigate the compositionality

properties of relevant equivalences on agents: one typical interesting question is whether bisimilarity is a congruence with respect to the operations defined on states.

Also the *structured transition systems*, studied for example in [11], are equipped with an (essentially) algebraic structure on both states and transitions. Here the operators are interpreted as structural ones, basic transitions are regarded as local changes on a distributed state, and the algebra on transitions ensures that basic transitions can be fired in any context and also in parallel. It has been shown that programs of many computational formalisms (including, among others, P/T Petri nets in the sense of [25], term rewriting systems, term graph rewriting [7], graph rewriting [15, 9, 18], Horn Clause Logic [6]) can be encoded as *heterogeneous graphs* having as collection of nodes algebras with respect to a suitable algebraic specification, and usually a poorer structure on arcs (often they are just a set). *Structured* transition systems are defined instead as graphs having a similar algebraic structure both on nodes and on arcs. A free construction associates with each program its induced structured transition system, from which a second free construction is used to generate the free model, i.e., a structured category which lifts the algebraic structure to the transition sequences. This induces an equivalence relation on the computations of a system, which is shown to capture some basic properties of true concurrency. Moreover, since the construction of the free model is a left adjoint functor, it is compositional with respect to operations on programs expressible as colimits.

Last but not least, also the tile transition systems introduced in the previous section have a rich algebraic structure on states, which are the arrows of a monoidal category, and the same structure is also defined on transitions, which are the tiles. Clearly, the general results presented at the end of the previous section, relating basic source and reflective horizontal decomposition properties, make an essential use of this algebraic structure.

For all the systems mentioned above (process algebra, structured transition systems and tile rewrite systems) the coalgebraic representation using functor P_L (for a suitable L) introduced in Proposition 5 is not completely satisfactory, because by definition the carrier is just a set and therefore the algebraic structure on both states and transitions is lost. This calls for the introduction of *structured coalgebras*, i.e., coalgebras for an endofunctor on a category $Alg(\Gamma)$ of algebras for a signature (or algebraic specification) Γ which is determined by the structure of states. Since it is natural to require that the structured coalgebraic representation of a system is compatible with the unstructured, set-based one, the following notion will be relevant.

Definition 14 (lifting). *Given endofunctors $B : \mathcal{C} \rightarrow \mathcal{C}$, $B' : \mathcal{C}' \rightarrow \mathcal{C}'$ and a functor $V : \mathcal{C}' \rightarrow \mathcal{C}$, B' is called a lifting of B along V , if $B'V = V B$.*

In particular, if $V^\Gamma : Alg(\Gamma) \rightarrow \mathbf{Set}$ is the underlying set functor, one will consider typically a functor $B' : Alg(\Gamma) \rightarrow Alg(\Gamma)$ which is a lifting of P_L along V^Γ .

The structured coalgebraic representation of transition systems has been studied in [32] for the case of CCS and other process algebra whose operational

semantics is given by SOS rules in the DeSimone format, and in [8] for structured transition systems. In the first case the lifting of P_L is determined by the SOS rules, while in the second one it is uniquely determined by the specification Γ . In both cases, as well as for the case of tile transition systems addressed in the next sections, the following interesting fact applies [32, 8].

Proposition 7 (bisimilarity is a congruence in structured coalgebras). *Let Γ be an algebraic specification, L be a Γ -algebra of labels, and $B_L^\Gamma : \mathbf{Alg}(\Gamma) \rightarrow \mathbf{Alg}(\Gamma)$ be a lifting of $P_L : \mathbf{Set} \rightarrow \mathbf{Set}$. If $\langle S, \sigma \rangle$ is a B_L^Γ -coalgebra and $\langle S, \longrightarrow \rangle$ its corresponding structured LTS, then bisimilarity on $\langle S, \longrightarrow \rangle$ is a congruence with respect to the operators in Γ .*

The statement follows by the observation that the right adjoint $R : \mathbf{Set} \rightarrow P_L\text{-Coalg}$ of Proposition 6 lifts to a right adjoint $R^\Gamma : \mathbf{Alg}(\Gamma) \rightarrow B_L^\Gamma\text{-Coalg}$ for the forgetful functor U^Γ , with $V^\Gamma; R = R^\Gamma; V_B^\Gamma$ (see [32]), as shown in the following diagram.

$$\begin{array}{ccc}
 P_L\text{-Coalg} & \begin{array}{c} \xrightarrow{F_B^*} \\ \xleftarrow{V_B^*} \end{array} & B_L^\Gamma\text{-Coalg} \\
 \begin{array}{c} \uparrow R \\ \downarrow U \end{array} & & \begin{array}{c} \uparrow R^* \\ \downarrow U^* \end{array} \\
 \mathbf{Set} & \begin{array}{c} \xrightarrow{F^*} \\ \xleftarrow{V^*} \end{array} & \mathbf{Alg}(\Gamma)
 \end{array}$$

Now, since R^Γ and V_B^Γ are both right adjoints, $B_L^\Gamma\text{-Coalg}$ inherits a final object $R^\Gamma(\mathbf{1})$ from $\mathbf{Alg}(\Gamma)$ which is then preserved by V_B^Γ . Hence, bisimilarity induced by the final morphism to $R^\Gamma(\mathbf{1})$ in $B_L^\Gamma\text{-Coalg}$ is determined by the underlying sets and functions, that is, its definition does not use the algebraic structure of states and transitions. Since the final morphisms in $B_L^\Gamma\text{-Coalg}$ are Γ -homomorphisms, it follows that bisimilarity is a congruence.

In other words, a structured transition system can be represented as a structured coalgebra only if bisimilarity is a congruence. This property certainly holds, for example, for specifications in GSOS format, which are considered in [32]. Certain structures are used there, called *bialgebras*, which combine aspects of algebras and coalgebras: bialgebras can be regarded as an alternative, equivalent presentation of structured coalgebras [8]. A specification in GSOS format is shown to satisfy a certain diagram called *pentagonal law*, which ensures the existence both of an algebra of transition systems and of an algebraic structure on their states. The pentagonal law also makes sure that bisimilarity is a congruence, showing that GSOS specifications perfectly fit in the structured coalgebraic framework.

A rather more general specification format is considered in [10], namely, the *algebraic* format [16], where the premise of a rule consists entirely of transitions on variables, and which generalizes rules in deSimone format by allowing complex terms in the source of the conclusion of a rule. In that paper, we first studied under which conditions transition systems can be represented as structured coalgebras on an environment category of algebras. It turned out that the conditions

which guarantee a coalgebraic presentation are very similar to the ones which ensure that bisimilarity is a congruence. Essentially they require that the behavior of the system is compositional, in the sense that all transitions from complex states can be derived using the rules from transitions out of component states. Thus one could say that what was considered a methodological convenience, i.e., that in the SOS approach each language construct is defined separately by a few clauses, is in fact mandatory to guarantee a satisfactory algebraic structure.

Next we proposed a general procedure which can be applied also to SOS specifications not satisfying the above property. More precisely, given any SOS specification in algebraic format, we defined its *context closure*, i.e., another specification including also the possible *context transitions*, which are transitions resulting in the addition of some context and labelled by it. We proved that bisimilarity for the context closure corresponds to *dynamic bisimilarity* for the original specification, which is by definition the coarsest bisimulation which is a congruence: as a consequence the context closure of a system is always representable as a structured coalgebra. This result is particularly relevant for *open systems*, for which dynamic bisimilarity seems to be the right notion of equivalence, since it takes into account not only experiments based on communications with the external world, but also experiments consisting of the additions of new components.

A different point of view is taken in [8], where it is argued that the property that bisimilarity is a congruence is too restrictive for structured transition systems, because it implies that basic transitions are defined only on atomic states. As a simple example, let us introduce the structured transition system associated with a simple P/T Petri net N consisting of places $S = \{a, b, c\}$ and of a single transition $T = \{t : a \oplus b \rightarrow c\}$ (consuming one token of place a and one of b and producing one in c). According to [25], the relevant algebraic structure is that of commutative monoids: the markings of the net can be regarded as elements of the free commutative monoid S^\oplus , and the structured transition system associated with N , denoted $TS(N)$, is obtained by adding idle transitions for each place and by extending the parallel composition operation \oplus in an obvious way to transitions. Now, let us assume that idle transitions are not visible (formally, they are labeled with the unit of the monoid of labels). Then it is easily seen that markings a and b are bisimilar (only idle transitions are possible) and clearly b and b are bisimilar as well. However, we have $a \oplus b \xrightarrow{t} c$, while only the idle transition is possible from $b \oplus b$. This shows that these two states are not bisimilar. Therefore one single basic transition having as source a composed state is sufficient to show that bisimilarity is not a congruence. As a consequence, in [8] the notion of *lax coalgebra* is introduced, which weakens the standard definition in order to allow for a full, coalgebra-like representation of structured transition systems.

5 Double and Tile Transition Systems as Coalgebras

In the previous section we have shown that labeled transition systems can be represented as coalgebras for an endofunctor on the category of sets. In this section, a similar representation for tile models (i.e., monoidal double categories) shall be developed.

In analogy to labeled transition systems, *double transition systems* are defined as flat monoidal double categories over a fixed vertical 1-category \mathcal{V} of observations. In Proposition 5 it is shown that coalgebra morphisms correspond to morphisms between labeled transition systems which (preserve and) reflect transitions. Below, an analogous restriction on monoidal double functors is introduced.

Definition 15 (double and tile transition systems). *Given a monoidal category of observations \mathcal{V} , a double transition system over \mathcal{V} is a flat monoidal double category \mathcal{D} which has \mathcal{V} as vertical 1-category.*

A morphism between double transition systems \mathcal{D} and \mathcal{D}' over \mathcal{V} is a monoidal double functor $F : \mathcal{D} \rightarrow \mathcal{D}'$ which acts on \mathcal{V} as identity. It reflects transitions if

$$\forall f \in \mathcal{H}. F(f) \xrightarrow{\frac{a}{b}} g' \in \mathcal{D}' \implies \exists g \in \mathcal{H}. f \xrightarrow{\frac{a}{b}} g \wedge F(g) = g'$$

We denote by $\mathbf{fsMDCat}_{\mathcal{V}}$ the category of double transition systems over \mathcal{V} and transition reflecting morphisms.

Given a tile rewrite system $\mathcal{T} = \langle S, \Sigma_h, \Sigma_v, R \rangle$, its associated tile transition system is its free model $F_{\mathcal{T}}(\mathcal{T})$, seen as a double transition system over $\mathbf{M}(\Sigma_v)$.

The endofunctor whose coalgebras represent double transition systems is only slightly more complex than functor P_L defined in Section 4. Since the states of a double transition system are arrows of a category (the horizontal 1-category \mathcal{H}), they are typed by their source and target objects. Consequently, the carrier of the corresponding coalgebra shouldn't be just a set but a family of sets indexed by pairs of objects of \mathcal{V} . The endofunctor $P_{\mathcal{V}}$ defined below on the category $\mathbf{Set}^{|\mathcal{V}^2|}$ is therefore a many-sorted version of P_L as defined in Section 4.

Definition 16 (endofunctor $P_{\mathcal{V}}$ for double transition systems). *Given a monoidal category $\mathcal{V} = \langle V, \otimes, e \rangle$, the functor $P_{\mathcal{V}} : \mathbf{Set}^{|\mathcal{V}^2|} \rightarrow \mathbf{Set}^{|\mathcal{V}^2|}$ is defined for every $|\mathcal{V}| \times |\mathcal{V}|$ -indexed set S by*

$$P_{\mathcal{V}}(S)(n, m) = \mathcal{P}_c \left(\bigcup_{n', m' \in |\mathcal{V}|} \mathcal{V}(n, n') \times \mathcal{V}(m, m') \times S(n', m') \right)$$

On arrows of $\mathbf{Set}^{|\mathcal{V}^2|}$, i.e., $|\mathcal{V}| \times |\mathcal{V}|$ -indexed indexed families of functions, the functor is defined analogously.

Notice that, according to the interpretation of tiles as cells, two vertical arrows are provided as observations. Moreover, transitions do not necessarily preserve the type of the state because there may be cells whose vertical source and target (which are horizontal arrows) are arrows of different type.

Proposition 8 (double transition systems as coalgebras). *Given \mathcal{V} and $P_{\mathcal{V}}$ as above, there exists a functor $Clg : \mathbf{fsMDCat}_{\mathcal{V}} \rightarrow P_{\mathcal{V}}\text{-Coalg}$ such that for every countably branching double transition system $\mathcal{D} \in \mathbf{fsMDCat}_{\mathcal{V}}$ with horizontal 1-category \mathcal{H} and all $f, g \in \mathcal{H}$*

$$f \sim_{op} g \text{ iff } \phi(f) = \phi(g)$$

where $\phi : Clg(\mathcal{D}) \rightarrow 1_{P_{\mathcal{V}}}$ is the unique morphism into the final $P_{\mathcal{V}}$ -coalgebra (which exists by similar arguments as in Proposition 6).

Proof. For a double transition system \mathcal{D} as above, the corresponding $P_{\mathcal{V}}$ -coalgebra is $Clg(\mathcal{D}) = \langle (\mathcal{H}(n, m)_{n, m \in |\mathcal{V}|}, \delta(n, m)_{n, m \in |\mathcal{V}|}) \rangle$ with

$$\delta(n, m)(f) = \{ \langle a, b, g \rangle \mid f \xrightarrow[b]{a} g \in \mathcal{D} \} \quad (2)$$

for all $f \in \mathcal{H}(n, m)$. Notice that $|\mathcal{H}| = |\mathcal{V}|$. \square

Thus, the use of many-sorted coalgebras allows us to retain the typing of states of double transition systems by pairs of objects. However, the algebraic structure given by the operations of the horizontal 1-category \mathcal{H} is lost. This problem is solved in the next section by lifting many-sorted to structured coalgebras.

6 Lifting the Algebraic Structure

We follow the outline of Section 4: first, we specify explicitly the algebraic structure on states and transitions. Then, we lift the endofunctor $P_{\mathcal{V}}$ to the corresponding category of algebras. Finally we show that double transition systems which satisfy the reflective horizontal decomposition property can be actually represented as structured coalgebras for the lifted functor.

In a double transition system \mathcal{D} , the algebraic structure of states is given by the monoidal category structure of the horizontal 1-category \mathcal{H} . Since we have fixed a monoidal category of observations \mathcal{V} as vertical 1-category, we know in advance the monoid of objects V . This allows to regard a monoidal category as a total algebra for the following signature (given with respect to the fixed monoid $V = \langle O, \otimes, e \rangle$).

signature $\underline{\text{MonCat}}(V) =$

sorts

(n, m) for all $n, m \in O$

operations

$;\!_{n, m, k} : (n, m)(m, k) \rightarrow (n, k)$

for all $n, m, k \in O$

$n : \rightarrow (n, n)$

for all $n \in O$

$e : \rightarrow (e, e)$

$\otimes_{n, m, n', m'} : (n, m)(n', m') \rightarrow (n \otimes n', m \otimes m')$ for all $n, m, n', m' \in O$

Algebras for this signature satisfying the usual laws of strict monoidal categories are the objects of the category \mathbf{MonCat}_V . Its arrows are given by $\mathbf{MonCat}(V)$ -homomorphisms, i.e., strict monoidal functors preserving the monoid of objects V . We usually omit the subscripts at operations when this causes no confusion.

As anticipated above, in order to represent a double transition system as a coalgebra in the category \mathbf{MonCat}_V , we have to provide a lifting of the endofunctor P_V defined on (families of) sets to the category of algebras. In the definition below, the resulting functor is called \hat{P}_V .

Definition 17 (lifting endofunctor P_V to \mathbf{MonCat}_V). *The functor $\hat{P}_V : \mathbf{MonCat}_V \rightarrow \mathbf{MonCat}_V$ is defined as follows. Given $M = \langle S, ;, id, \otimes, e \rangle \in |\mathbf{MonCat}_V|$, the algebra*

$$\hat{P}_V(M) = PM = \langle P_V(S), ;^{PM}, id^{PM}, \otimes^{PM}, e^{PM} \rangle$$

is given by

$$\begin{aligned} S;^{PM} T &= \{ \langle a, c, g; h \rangle \mid \langle a, b, g \rangle \in S, \langle b, c, h \rangle \in T \} \\ id^{PM} &= \{ \langle a, a, m \rangle \mid a : n \rightarrow m \in \mathcal{V} \} \\ S \otimes T &= \{ \langle a \otimes a', b \otimes b', g \otimes g' \rangle \mid \langle a, b, g \rangle \in S, \langle a', b', g' \rangle \in T \} \\ e^{PM} &= \{ \langle e, e, e \rangle \} \end{aligned}$$

On arrows of \mathbf{MonCat}_V , the functor is defined like P_V .

Next we show that a double transition system can be represented as a \hat{P}_V -coalgebra iff it satisfies the reflective decomposition property. Denote by $\mathbf{fsMDCat}_V^\Delta$ the full subcategory of $\mathbf{fsMDCat}_V$ whose objects are countably branching double transition systems satisfying the reflective decomposition property.

Proposition 9 (double transition systems as structured coalgebras). *Let \mathcal{D} be a countably branching double transition system and \mathcal{H} be its horizontal 1-category. Then, $\hat{C}lg(\mathcal{D}) = \langle \mathcal{H}, \delta \rangle$ with δ defined like in (2) is a \hat{P}_V -coalgebra if and only if \mathcal{D} satisfies the reflective horizontal decomposition property.*

Moreover, this translation extends to a functor $\hat{C}lg : \mathbf{fsMDCat}_V^\Delta \rightarrow \hat{P}_V\text{-Coalg}$.

Thus, under the assumption of the reflective horizontal decomposition property, we actually retain the horizontal structure of double transition systems in the coalgebraic presentation. What is lost in any case is the vertical structure which is, however, not relevant for the notion of bisimilarity.

More precisely, we can show that the category $\mathbf{fsMDCat}_V^\Delta$ of horizontally decomposable double transition systems is isomorphic to the full subcategory of $\hat{P}_V\text{-Coalg}$ whose objects are coalgebras $\langle \mathcal{H}, \delta \rangle$ with transitive and reflexive transition “relations”. This means, writing $f \xrightarrow[a]{a} g$ for $\langle a, b, g \rangle \in \delta(f)$, that they have to satisfy the rules (*vert*) and (*v-refl*) of Table 1.

7 An Application to CCS

As a case study, we present in this section a version without recursion of the *calculus of communicating systems* (shortly, CCS) introduced by Robin Milner [27]. The same presentation has appeared in [26]. Essentially the same presentation, but for a cartesian (rather than monoidal) structure of configurations, was reported in [16]. CCS with the replication operator is modeled with tiles in [2]. A concurrent version of CCS with localities is in [14].

Syntax of CCS. Let Δ be the alphabet for basic actions (which is ranged over by α) and $\bar{\Delta}$ the alphabet of complementary actions ($\Delta = \bar{\bar{\Delta}}$ and $\Delta \cap \bar{\Delta} = \emptyset$); the set $\Lambda = \Delta \cup \bar{\Delta}$ will be ranged over by λ . Let $\tau \notin \Lambda$ be a distinguished action, and let $A \cup \{\tau\}$ (ranged over by μ) be the set of CCS actions.

The syntax of finite CCS agents is defined by the following grammar

$$P ::= nil \quad | \quad \mu.P \quad | \quad P \setminus \alpha \quad | \quad P + P \quad | \quad P \mid P$$

Operational Semantics of CCS. The dynamic behavior of CCS closed agents can be described by a transition system TS_{CCS} , where labels are actions, states are closed CCS agents and the transition relation is freely generated from the following set of inference rules

$$\begin{array}{c} \frac{}{\mu.P \xrightarrow{\mu} P} \qquad \frac{P \xrightarrow{\mu} Q \quad \mu \notin \{\alpha, \bar{\alpha}\}}{P \setminus \alpha \xrightarrow{\mu} Q \setminus \alpha} \\ \\ \frac{P \xrightarrow{\mu} Q}{P + R \xrightarrow{\mu} Q} \qquad \frac{P \xrightarrow{\mu} Q}{R + P \xrightarrow{\mu} Q} \\ \\ \frac{P \xrightarrow{\mu} Q}{P \mid R \xrightarrow{\mu} Q \mid R} \qquad \frac{P \xrightarrow{\lambda} Q, P' \xrightarrow{\bar{\lambda}} Q'}{P \mid P' \xrightarrow{\tau} Q \mid Q'} \qquad \frac{P \xrightarrow{\mu} Q}{R \mid P \xrightarrow{\mu} R \mid Q} \end{array}$$

Given the transition

$$((a.nil + b.nil) \mid \bar{a}.nil) \setminus a \xrightarrow{\tau} (nil \mid nil) \setminus a,$$

its proof is as follows:

$$\frac{\frac{\frac{a.nil \xrightarrow{a} nil}{a.nil + b.nil \xrightarrow{a} nil} \quad \frac{}{\bar{a}.nil \xrightarrow{\bar{a}} nil}}{(a.nil + b.nil) \mid \bar{a}.nil \xrightarrow{\tau} nil \mid nil}}{((a.nil + b.nil) \mid \bar{a}.nil) \setminus a \xrightarrow{\tau} (nil \mid nil) \setminus a}$$

Abstract Semantics of CCS. Ordinary bisimilarity between CCS agents, written $P \sim_{ord} Q$, is just the relation $P \sim_{TSCCS} Q$ according to the general Definition 9. We now present the tile rewrite system for CCS.

Signatures of the CCS tile rewrite system. There is only one sort $\underline{1}$. The free monoid generated by it is represented by underlined natural numbers with $\underline{n} \otimes \underline{m} = \underline{m+n}$ and $e = \underline{0}$. For the horizontal signature the operators are: $nil \in (\Sigma_h^{CCS})_{\underline{0}, \underline{1}}$; $\underline{\mu}. and $\backslash \alpha \in (\Sigma_h^{CCS})_{\underline{1}, \underline{1}}$; $- + -$ and $- | - \in (\Sigma_h^{CCS})_{\underline{2}, \underline{1}}$; and $!(_) \in (\Sigma_h^{CCS})_{\underline{1}, \underline{0}}$. The latter constructor, called *eraser*, is needed to discard the rejected alternative after a choice step. Except for the eraser, the operators of the horizontal signature directly correspond to CCS syntactical operators. Thus we will consider CCS agents (not necessarily closed) as suitable arrows of the monoidal category $\mathbf{M}(\Sigma_h^{CCS})$. For the vertical signature the operators are $\underline{\mu}. $\in (\Sigma_v^{CCS})_{\underline{1}, \underline{1}}$.$$

Rules of the CCS tile rewrite system.

$$\begin{array}{ll}
\text{Pref}_\mu : \underline{\mu} \xrightarrow{\underline{1}} \underline{1} & \text{Res}_\mu^\alpha : \backslash \alpha \xrightarrow{\underline{\mu}} \backslash \alpha \quad \text{for } \mu \notin \{\alpha, \bar{\alpha}\} \\
\text{Suml}_\mu : - + - \xrightarrow{\underline{\mu}} \underline{1} \otimes ! & \text{Sumr}_\mu : - + - \xrightarrow{\underline{1} \otimes \underline{\mu}} ! \otimes \underline{1} \\
\text{Compl}_\mu : - | - \xrightarrow{\underline{\mu}} - | - & \text{Compr}_\mu : - | - \xrightarrow{\underline{1} \otimes \underline{\mu}} - | - \quad \text{Synch}_\lambda : - | - \xrightarrow{\underline{\lambda} \otimes \bar{\lambda}} - | -
\end{array}$$

We call \mathcal{T}_{CCS} the CCS tile rewrite system. The rules of \mathcal{T}_{CCS} closely correspond to the SOS rules. For instance rule Pref_μ states, as its SOS counterpart, that constructor $\underline{\mu}$ can be deleted, i.e., it can be replaced by the identity $\underline{1}$. Furthermore, the trigger is also the identity, and thus the corresponding SOS rule is an axiom. Finally, the effect is $\underline{\mu}$ and this corresponds to the label of the transition in the SOS case. As another example, rule Suml_μ defines left choice. The initial configuration is the constructor $- + - : \underline{2} \rightarrow \underline{1}$, while the final configuration is $\underline{1} \otimes ! : \underline{2} \rightarrow \underline{1}$, which states that the first component is preserved and the second component is discarded. The trigger states that in the first component we must have an action $\underline{\mu}$ while no action (i.e., identity action) is required on the second component. Action $\underline{\mu}$ is then transferred to the effect.

The tile corresponding to the previous example is obtained as follows:

$$A = (((nil * \text{Pref}_a) \otimes (b.nil) * \text{Suml}_a) \otimes (nil * \text{Pref}_{\bar{a}})) * \text{Synch}_a * \text{Res}_\tau^a.$$

The next proposition states the equivalence of the SOS and tile definitions of CCS.

Proposition 10 (SOS/tile equivalence for CCS). *Given two CCS agents P and Q , we have $P \sim_{ord} Q$ if and only if $P \sim_{op} Q$ for the CCS tile rewrite system.*

In order to represent \mathcal{T}_{CCS} as a coalgebra, we have to verify the reflective horizontal decomposition property. Thanks to Proposition 4, it is enough to check by inspection that all rules of \mathcal{T}_{CCS} satisfy the basic source property.

Proposition 11 (coalgebraic presentation of CCS tile rewrite system).

The CCS tile rewrite system \mathcal{T}_{CCS} satisfies the basic source property and hence the reflective horizontal decomposition property. As a consequence, the tile transition system $F_T(\mathcal{T}_{CCS})$ generated by \mathcal{T}_{CCS} can be represented as a $\hat{P}_{\mathbf{M}(\Sigma_h^{CCS})}$ -coalgebra

$$\hat{C}lg(F_T(\mathcal{T}_{CCS})) = \langle \mathbf{M}(\Sigma_h^{CCS}), \sigma \rangle$$

such that $P \sim_{op} Q$ iff $\phi(P) = \phi(Q)$ for each two processes $P, Q \in \mathbf{M}(\Sigma_h^{CCS})$, where $\phi : \hat{C}lg(F_T(\mathcal{T}_{CCS})) \rightarrow 1_{\hat{P}_{\mathbf{M}(\Sigma_h^{CCS})}}$ is the unique arrow to the final $\hat{P}_{\mathbf{M}(\Sigma_h^{CCS})}$ -coalgebra.

Corollary 1 (\sim_{op} is a congruence). Tile bisimilarity \sim_{op} for CCS is a congruence w.r.t. the operations of the horizontal 1-category of states.

Due to Proposition 10, this implies the well-known fact that bisimilarity is a congruence for CCS. Notice however that the above statement is more general, since it does not only apply to closed process terms but also to processes which contain variables.

8 Conclusion

In this paper we have investigated the relation between two models of concurrent systems: tile rewrite systems and coalgebras. Tile rewrite systems consist of rewrite rules with side effects which are reminiscent of SOS rules [30]. For these rules, which can also be seen as elementary transitions, closure operations of parallel and sequential composition and synchronization are defined. The models of tile rewrite systems are monoidal double categories.

Coalgebras can be considered, in a suitable mathematical setting, as dual to algebras. They can be used as models of dynamical systems with hidden states in order to study concepts of observational equivalence and bisimilarity in a more general setting.

We have pointed out that, in order to retain in the coalgebraic presentation of tile models the operations of parallel composition and synchronization, coalgebras have to be endowed with an algebraic structure. This has led us to the concept of *structured coalgebras*, i.e., coalgebras for an endofunctor on a category of algebras. However, structured coalgebras are a more restricted notion than tile models, since they only allow to represent models where bisimilarity is a congruence. For tile models, this condition corresponds to functoriality of bisimilarity, which is ensured by the horizontal decomposition property.

The insights on the relation between tile rewrite systems and structured coalgebras can be obtained by applying to tile models (seen as double transition systems) the results of [8, 10] on the coalgebraic presentation of transition systems with algebraic structure specified by SOS rules.² In that paper, we have characterized those transition systems for which a coalgebraic presentation is possible

² In fact, what would be needed is a many-sorted version of the presentation in [10] which, for simplicity, is restricted to the one-sorted case.

and the classes of SOS specifications generating such “well-behaved” systems. It turned out that the conditions which guarantee a coalgebraic presentation are very similar to the ones which ensure that bisimilarity is a congruence. They require that the behavior of the system is compositional in the sense that all transitions from complex states can be derived using the transitions out of component states: this is essentially the statement of the horizontal decomposition property.

In the case without structural axioms, such condition is verified if each basic rule in the tile rewrite system has signature operators both as initial configuration and as effect; indeed this is the common point of many SOS formats (see e.g. [12, 17, 5]). Notice, however, that, with structural axioms, the situation is more complicated, since signature operators can be equivalent to complex terms, and complex states may be decomposed into component states in many different ways.

We think that the coalgebraic presentation of monoidal tile rewrite systems can be a starting point for transferring to the coalgebraic setting concrete applications of tile logic to formalisms like concurrent or located CCS, π -calculus, etc.

9 Acknowledgments

We would like to thank Roberto Bruni for his comments.

References

1. R. Bruni, F. Gadducci and U. Montanari, Normal Forms for Partitions and Relations, Proc. 13th Workshop on Algebraic Development Techniques, Lisbon, April 2-4, 1998, Springer LNCS, 1999, to appear.
2. R. Bruni, J. Meseguer and U. Montanari, Executable Tile Specifications for Process Calculi, in: Jean-Pierre Finance, Ed., FASE'99, Springer LNCS 1577, pp. 60-76.
3. R. Bruni, J. Meseguer and U. Montanari, Symmetric Monoidal and Cartesian Double Categories as a Semantic Framework for Tile Logic, to appear in MSCS.
4. R. Bruni and U. Montanari, Cartesian Closed Double Categories, their Lambda-Notation, and the Pi-Calculus, Proc. LICS'99, to appear.
5. B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation can't be traced. *Journal of the ACM*, 42(1):232–268, 1995.
6. A. Corradini and A. Asperti. A categorical model for logic programs: Indexed monoidal categories. In *Proceedings REX Workshop, Beekbergen, The Netherlands, June 1992*, volume 666 of *LNCS*. Springer Verlag, 1993.
7. A. Corradini and F. Gadducci. A 2-categorical presentation of term graph rewriting. In E. Moggi and G. Rosolini, editors, *Category Theory and Computer Science*, volume 1290 of *LNCS*, pages 87–105. Springer Verlag, 1997.
8. A. Corradini, M. Große-Rhode, and R. Heckel. Structured transition systems as lax coalgebras. In B. Jacobs, L. Moss, H. Reichel, and J. Rutten, editors, *Proc. of First Workshop on Coalgebraic Methods in Computer Science (CMCS'98), Lisbon, Portugal*, volume 11 of *Electronic Notes of TCS*. Elsevier Science, 1998. <http://www.elsevier.nl/locate/entcs>.

9. A. Corradini, M. Große-Rhode, and R. Heckel. An algebra of graph derivations using finite (co-) limit double theories. In J.L. Fiadeiro, editor, *Proc. 13th Workshop on Algebraic Development Techniques (WADT'98)*, volume 1589 of *LNCS*. Springer Verlag, 1999.
10. A. Corradini, R. Heckel, and U. Montanari. From SOS specifications to structured coalgebras: How to make bisimulation a congruence. In B. Jacobs and J. Rutten, editors, *Proc. of Second Workshop on Coalgebraic Methods in Computer Science (CMCS'99), Amsterdam*, volume 19 of *Electronic Notes of TCS*. Elsevier Science, 1999. <http://www.elsevier.nl/locate/entcs>.
11. A. Corradini and U. Montanari. An algebraic semantics for structured transition systems and its application to logic programs. *Theoret. Comput. Sci.*, 103:51–106, 1992.
12. R. De Simone. Higher level synchronizing devices in MEIJE-SCCS. *Theoret. Comput. Sci.*, 37:245–267, 1985.
13. C. Ehresmann, *Catégories Structurées*: I and II, Ann. Éc. Norm. Sup. 80, Paris (1963), 349-426; III, Topo. et Géo. diff. V, Paris (1963).
14. G. Ferrari and U. Montanari, Tile Formats for Located and Mobile Systems, Information and Computation, to appear.
15. F. Gadducci and R. Heckel. A inductive view of graph transformation. In F. Parisi Presicce, editor, *Recent Trends in Algebraic Development Techniques, LNCS 1376*, pages 223 – 237. Springer Verlag, 1998.
16. F. Gadducci and U. Montanari. The tile model. In G. Plotkin, C. Stirling, and M. Tofte, editors, *Proof, Language and Interaction: Essays in Honour of Robin Milner*. MIT Press, 1999. To appear. An early version appeared as Tech. Rep. TR-96/27, Dipartimento di Informatica, University of Pisa, 1996. Paper available from <http://www.di.unipi.it/~gadducci/papers/TR-96-27.ps.gz>.
17. J.F. Groote and F. Vandraager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100:202–260, 1992.
18. R. Heckel. *Open Graph Transformation Systems: A New Approach to the Compositional Modelling of Concurrent and Reactive Systems*. PhD thesis, TU Berlin, 1998.
19. G.M. Kelly and R.H. Street. Review of the elements of 2-categories. In G.M. Kelly, editor, *Sydney Category Seminar*, volume 420 of *Lecture Notes in Mathematics*, pages 75–103. Springer Verlag, 1974.
20. F.W. Lawvere. Some algebraic problems in the context of functorial semantics of algebraic theories. In *Proc. Midwest Category Seminar II*, number 61 in Springer Lecture Notes in Mathematics, pages 41–61, 1968.
21. K.G. Larsen, L. Xinxin, *Compositionality Through an Operational Semantics of Contexts*, in Proc. ICALP'90, LNCS 443, 1990, pp. 526-539.
22. J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *TCS*, 96:73–155, 1992.
23. J. Meseguer, *Rewriting Logic as a Semantic Framework for Concurrency: A Progress Report*, in: U. Montanari and V. Sassone, Eds., *CONCUR'96: Concurrency Theory*, Springer LNCS 1119, 1996, 331-372.
24. J. Meseguer. Membership algebra as logical framework for equational specification. In F. Parisi Presicce, editor, *Recent Trends in Algebraic Development Techniques, LNCS 1376*, pages 18–61. Springer Verlag, 1998.
25. J. Meseguer and U. Montanari. Petri nets are monoids. *Information and Computation*, 88(2):105–155, 1990.

26. J. Meseguer and U. Montanari. Mapping tile logic into rewriting logic. In Francesco Parisi-Presicce, editor, *Recent Trends in Algebraic Development Techniques*, number 1376 in Springer LNCS, pages 62–91, 1998.
27. R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
28. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes. *Information and Computation*, 100:1–77, 1992.
29. U. Montanari and F. Rossi, Graph Rewriting, Constraint Solving and Tiles for Coordinating Distributed Systems, to appear in *Applied Category Theory*.
30. G. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Aarhus University, Computer Science Department, 1981.
31. J.J.M.M. Rutten. Universal coalgebra: a theory of systems. Technical Report CS-R9652, CWI, 1996. To appear in *TCS*.
32. D. Turi and G. Plotkin. Towards a mathematical operational semantics. In *Proc. of LICS'97*, pages 280–305, 1997.