URL: http://www.elsevier.nl/locate/entcs/volume62.html 20 pages

Comparing cospan-spans and tiles via a Hoare-style process calculus ⋆

Fabio Gadducci [a], Piergiulio Katis [b], Ugo Montanari [a], Nicoletta Sabadini [b], Robert F.C. Walters [b]

[a] *Dipartimento di Informatica, Università di Pisa*
Corso Italia 40, 56125 Pisa, Italy.
gadducci,ugo@di.unipi.it

[b] *Facoltà di Science MM. FF. NN., Università dell'Insubria*
via Valleggio 11, I-22100 Como, Italy.
Piergiulio.Katis,Nicoletta.Sabadini,Robert.Walters@uninsubria.it

Abstract

The large diffusion of concurrent and distributed systems has spawned in recent years a variety of new formalisms, equipped with features for supporting an easy specification of such systems. The aim of our note is to compare two recent formalisms for the design of concurrent and distributed systems, namely the *tile logic* and the *cospan-span model*. We first present a simple, yet rather expressive Hoare-style process algebra; then, after presenting the basics of both approaches, we compare them via their modeling of the calculus.

Key words: cospan-span approach; tile logic; process algebras; concurrent, distributed and interactive systems.

1 Introduction

The advances in the growing field of languages for the specification of concurrent and distributed systems, and the difficulties in relating different calculi and special-purpose formalisms, clearly ask for the development of meta-frameworks, where common aspects between different approaches can be factored out and exploited. But even at this level the task of reducing the complexity of these approaches into a unique meta-model is daunting, so that we should aim at providing suitable comparison results between meta-frameworks.

⋆ Research partly supported by the Italian MIUR Project *Teoria della Concorrenza, Linguaggi di Ordine Superiore e Strutture di Tipi* (TOSCA).

The cospan-span model of distributed systems was introduced in [14] (under the name of CP automata) as an algebra of transition systems/labelled graphs. The choice of operations was influenced by category theory, and the model is closely related to the automata of Arnold and Nivat [1]. The basic idea of the model is to represent a system by a graph of states and transitions, with extra structure required for capturing two fundamental operations on systems: *parallel composition with communication* requires *interfaces*, and *sequential composition* requires *conditions*.

The full expressiveness of the cospan-span model has yet to be understood. However the model includes the span model described in [13] which concerns finite automata constructed compositionally with communication-parallel operations. It also includes the circuit and sequential algorithm models of [12], which incorporate data types, and are Turing complete. The ability to express systems with changing geometry, and hence mobility, was described in [14]. Recursive equations in cospan-span were discussed in [15], together with further examples of mobility.

Tile logic [10] is a framework for the specification of rule-based systems, whose behaviour relies on the notions of *synchronization* and *side-effects*. The key idea is to enrich rewrite rules with an *observation*, carrying information on the possible behaviour of its arguments, that is, imposing a behavioural constraint to the terms to which it can be applied. The resulting formalism extends similar approaches such as rewriting logic [16], allowing for representing generic *open configurations* of *interactive systems with coordination*.

Space limitations prevent us to discuss the range of applications of the framework, so that we may just cite a few references. The characterisation of tile logic as an extension of the classical sos specification methodology for process algebras was already exploited in the original paper [10], while its application to calculi with name mobility and localities appears in [9]. An overview of its use with formalisms for concurrency, as well as a comparison with other rule-based frameworks (much in the spirit of the present article) can be found in [11]. The application to logic programming is reported in [7]. As for the properties of tile logic as a meta-framework, investigations have been carried out on ensuring that modularity properties are preserved by the specifications (i.e., that suitable behavioural equivalences are congruences if certain requirements on the format of the rules [3,8] are satisfied). The executability of tile specifications is guaranteed by their embedding into rewriting logic [5]. Finally, an higher-order extension of the framework has been proposed in [6].

A direct mapping between the cospan-span model and the tile logic would be rather cumbersome, since, on the one hand, the former focuses on the algebraic structure over *states*, with a denotational semantics flavour; whilst, on the other hand, the latter is a rule-based formalism focusing on a powerful inference mechanism for defining rewrites, hence *transitions*, with an operational semantics flavour.

We compare instead the two approaches by analysing their respective encodings of a Hoare-style process algebra, the SPA (for *span process algebra*) calculus, that we are going to introduce. In SPA, each process has a fixed set of channels; actions are allowed to occur simultaneously on all the channels of a process; asynchrony is modeled by the use of silent actions; communication is anonymous: the communication between two processes P and Q is described by operations which connect some of the ports of P to some of the ports of Q, possibly hiding them, and a process can only communicate with other processes via its channels; broadcasting, or synchronization of many processes, is presented as a derived operation, using special "diagonal" components.

The paper has the following structure. Section 2 introduces SPA. Section 3 presents the basics of cospan-span model, and the denotational encoding of SPA. Section 4 presents the basics of tile logic, and the operational encoding of SPA. Finally, Section 5 compares the two approaches, illustrating a correspondence result between the respective encodings of SPA.

2 A Hoare-style process calculus

We present a simple process calculus, using a Hoare-style semantics, that we denote as SPA. As we will see, SPA expressions will be built out of (among others) a summation operation, a non-communicating parallel operation, a (derived) family of communicating parallel operations, and recursion.

2.1 The construction of well-formed SPA *expressions*

We will deal with *well-formed expressions*, considering also the *channels* occurring in a process as part of the specification.

Definition 2.1 (well-formed expressions) *Given a countable set* $U = \{A, B, \ldots\}$ *of channels, a* well-formed (process) expression *is a pair* $P : \sigma$*, for* P *a process (i.e, a term out of a signature, to be defined later on) and* σ *a type, consisting of a finite set of channels (i.e.,* $\sigma \subset U$*).*

Now we present the inference system defining the set $\mathcal{P}$ of well-formed, closed SPA expressions. We begin by assuming the following data is given:

- for each channel A an infinite set L_A of *local actions* which includes a specified element τ, called the *silent* action;
- for each pair A, B of channels an isomorphism $\phi_{A,B} : L_A \to L_B$ preserving the silent action (and such that $\phi_{B,A} = \phi_{A,B}^{-1}$ and $\phi_{B,C} \circ \phi_{A,B} = \phi_{A,C}$).

For each type σ we denote by Act_σ its set of *(global) actions*: each action is a subset of $\uplus_{A \in \sigma} L_A$ containing exactly one element (called *component action*) for each channel $A \in \sigma$. The set Act of all possible actions, ranged over by μ, is defined as $\uplus_{\sigma \subset U} Act_\sigma$. Now, the definition below jointly presents the set of processes P and of well-formed expressions $P : \sigma$.

Definition 2.2 (well-formed expressions of SPA) *Let $X = \uplus_{\sigma \subset U} X_\sigma$ be a set of variables. The set of processes $\mathcal{P}roc_X$ and the set of well-formed* SPA *expressions $\mathcal{P}_X$ are jointly generated by the following set of axioms and rules*

$$\frac{\sigma \subset U}{\mathsf{0} : \sigma} \qquad \frac{x \in X_\sigma}{x : \sigma} \qquad \frac{P_i : \sigma,\ \ \mu_i \in Act_\sigma \text{ for } i \in I}{\sum_{i \in I} \mu_i.P_i : \sigma}$$

$$\frac{P : \sigma}{P[\Phi] : \Phi(\sigma)} \text{ for } \Phi \text{ relabeling} \qquad \frac{P : \sigma \cup \{A\}}{(\nu A)P : \sigma \backslash \{A\}}$$

$$\frac{P : \sigma,\ \ Q : \gamma}{P \| Q : \sigma \cup \gamma} \text{ for } \sigma \cap \gamma = \emptyset$$

$$\frac{P : \sigma \cup \{A, B\}}{[B \Rightarrow A]P : (\sigma \backslash \{B\}) \cup \{A\}} \qquad \frac{P : \sigma,\ \ x \in X_\sigma}{rec\, x.P : \sigma}$$

For each type σ, the set $\mathcal{P}_{\sigma,X}$ denotes those well-formed expressions of the kind $P : \sigma$. Then, $\mathcal{P}_\sigma$ and $\mathcal{P}$ denote the sub-sets of respectively $\mathcal{P}_{\sigma,X}$ and $\mathcal{P}_X$, containing just closed *expressions, i.e. those expressions $P : \sigma$ such that the process P contains no free variables.*

A few comments on the intended meaning of the operators are in order.

- Activity occurs simultaneously on each channel of a process: a well-formed expression $P : \sigma$ defines a transition system whose edges are labeled by actions $\mu \in Act_\sigma$. Processes communicate via common channels.
- Asynchrony is modeled by silent actions; for instance, the expression $\{a_A, b_B, \tau_C\}.P : \{A, B, C\}$ represents a system that can perform an action $\{a_A, b_B, \tau_C\}$: (the component action) a occurs on channel A, b occurs on channel B and nothing occurs on channel C.
- Given a bijection Φ from U to itself, $P[\Phi]$ is obtained by substituting each component action a_A with $\phi_{A,\Phi(A)}(a_A)$.
- Summation will have the usual interpretation. Notice that the expression $\sum_{i \in I} \mu_i.P_i$ is well-formed if all the P_i's have the same type σ, and all the actions μ_i's belong to Act_σ.
- We interpret hiding as masking component actions on a given channel (equivalently, as deleting that channel), but otherwise, differently from e.g. restriction operators, not preventing any transition.
- We interpret merging as requesting the fusion of one channel into another.
- The interpretation of $P \| Q$ is that processes P and Q are operating in parallel and independently; in particular, they may execute actions simultaneously. They cannot synchronize, though, since we are not connecting any channel.
- Recursion is handled by an unfolding rule, replacing x in P with an instance of P itself.

2.2 Inference Rules for SPA

It is clear that for each non-deadlocked process P there is at most one-well well-formed expression $P : \sigma$. Thus, we will sometimes drop the type from an expression, whenever this is not going to cause any confusion.

Definition 2.3 (operational semantics of SPA) *The* SPA transition system *is the relation $\mathcal{T}_{\text{spa}} \subseteq \mathcal{P} \times Act \times \mathcal{P}$ inductively generated by the following set of axioms and inference rules*

$$\frac{}{(\sum_{i\in I} \mu_i.P_i) \xrightarrow{\mu_i} P_i} \qquad \frac{P \xrightarrow{\mu} Q}{P[\Phi] \xrightarrow{\Phi(\mu)} Q[\Phi]} \text{ for } \Phi \text{ relabeling}$$

$$\frac{P \xrightarrow{\mu\cup\{a\}} P'}{(\nu A)P \xrightarrow{\mu\setminus\{a\}} (\nu A)P'} \text{ for } a \in L_A \qquad \frac{Q \xrightarrow{\mu_1} Q', \quad R \xrightarrow{\mu_2} R'}{(Q\|R) \xrightarrow{\mu_1\cup\mu_2} (Q'\|R')}$$

$$\frac{P \xrightarrow{\mu\cup\{\phi_{B,A}(b),b\}} P'}{[B \Rightarrow A]P \xrightarrow{(\mu\setminus\{b\})\cup\{\phi_{B,A}(b)\}} [B \Rightarrow A]P'} \text{ for } b \in L_B$$

$$\frac{P[{}^{x}/_{rec\,x.P}] \xrightarrow{\mu} P'}{P \xrightarrow{\mu} P'} \text{ for } \_[{}^{x}/_{rec\,x.P}] \text{ capture avoiding substitution}$$

where $P \xrightarrow{\mu} Q$ means that $\langle P, \mu, Q\rangle \in \mathcal{T}_{\text{spa}}$.

We say that a process P may execute the action μ and become Q if the *transition* $\langle P, \mu, Q\rangle \in \mathcal{T}_{\text{spa}}$. No confusion may arise, since $\mathcal{T}_{\text{spa}}$ is obtained as the disjoint union of transition systems $\mathcal{T}_\sigma$'s: for each of them, the set of states is $\mathcal{P}_\sigma$, and transitions are labeled by actions $\mu \in Act_\sigma$.

2.3 Examples and derived operators

2.3.1 Communicating parallel

The intuitive interpretation of $P \star Q$ is that the two processes P and Q are operating in parallel, but the common channels among P and Q have been connected. That is, P can execute an action μ_1 at the same time as Q can execute an action μ_2 – but, for each channel $A \in (\sigma \cap \tau)$, the component actions of μ_1 and μ_2 must agree. Moreover, the operation has the effect of *hiding* the common channels.

Let $P : \sigma$ and $Q : \gamma$ be closed, well-formed expressions. If Φ is a relabeling mapping each channel $A \in \sigma \cap \gamma$ into a channel $\Phi(A) \notin \sigma \cup \gamma$, the communicating parallel composition $P \star Q : (\sigma \cup \gamma)\setminus(\sigma \cap \gamma)$ is defined as

$$P \star Q = (\nu\sigma \cap \gamma)[\Phi(\sigma \cap \gamma) \Rightarrow \sigma \cap \gamma](P\|(Q[\Phi]))$$

where $[\Phi(\sigma\cap\gamma) \Rightarrow \sigma\cap\gamma]$ is just the application of $[\Phi(A) \Rightarrow A][\Phi(B) \Rightarrow B]\ldots$, and $(\nu\sigma \cap \gamma)$ of $(\nu A)(\nu B)\ldots$ for all $A, B \in \sigma \cap \gamma$.

2.3.2 Joining three processes

We define a few processes that are used later on.

The process $\Delta_A^{B,C}$, of type $\{A, B, C\}$, splits a channel, and it is defined as

$$\Delta_A^{B,C} = rec\, x.(\sum_{a \in A} \{a, \phi_{A,B}(a), \phi_{A,C}(a)\}.x)$$

Instead, the process $\eta^{B,C}$, of type $\{B, C\}$, just creates two component actions out of a silent action, and it is defined as $\eta^{B,C} = (\nu A)\Delta_A^{B,C}$.

Let $P : \sigma_1 \cup \{A\}$, $Q : \sigma_2 \cup \{B\}$ and $R : \sigma_3 \cup \{C\}$ be well-formed expressions, such that the σ_i's and $\{A\}$, $\{B\}$, $\{C\}$ are mutually disjoint. Then, the well-formed expression

$$((P \star \Delta_A^{B,C}) \star (Q \| R)) : \sigma_1 \cup \sigma_2 \cup \sigma_3$$

is to be thought of as a system formed by splitting the channel A into B and C, and then connecting it with the channels of the non-communicating parallel composition of Q and R. The result is that the channel A of the process P has been joined to the channels B and C of the processes Q and R, respectively. It is clear that to give a transition out of $((P \star \Delta_A^{B,C}) \star (Q \| R))$ is to give three transitions $P \overset{\mu_1 \cup \{a\}}{\longrightarrow} P'$, $Q \overset{\mu_2 \cup \{\phi_{A,B}(a)\}}{\longrightarrow} Q'$ and $R \overset{\mu_3 \cup \{\phi_{A,C}(a)\}}{\longrightarrow} R'$.

2.3.3 The Dining Philosophers

We give a well-formed expression intended to model an asynchronous variant of the dining philosophers system: other versions (where e.g. philosophers must pick up the left fork first) may be easily captured in the formalism. For the sake of simplicity, we assume that for each channel mentioned later on, its set of local actions contains $\{\tau, \mathsf{l}, \mathsf{u}\}$, and that those symbols are preserved by the ϕ's. The symbol l denotes the action lock and the symbol u the action unlock.

First, we describe the action of the left hand of a philosopher, as

$$P_L = rec\, x.(\tau.x + \mathsf{l}.(rec\, y.(\tau.y + \mathsf{u}.x)))$$

so that the associated well-formed expression is $P_L : \{L\}$; and, assuming that Φ_L^R just swaps the channels L and R, then $P = P_L \| P_R$, for $P_R = P_L[\Phi_L^R]$, with $P : \{L, R\}$. The philosopher may act concurrently, capturing (or releasing) two forks at once. Instead, each fork may be captured only by one philosopher at most, either on the right or on the left. So, if we consider the sub-processes

$$F_L = \{\mathsf{l}_I, \tau_J\}.(rec\, y.(\{\tau_I, \tau_J\}.y + \{\mathsf{u}_I, \tau_J\}.x))$$

and, for the relabeling Φ_I^J just swapping I and J, the process is

$$F = rec\, x.(\{\tau_I, \tau_J\}.x + F_L + F_L[\Phi_I^J])$$

so that the associated well-formed expression is $F : \{I, J\}$.

So, the behaviour of a single philosopher is described by the process

$$DP_1 = \eta^{L,I} \star (P \star F[\Phi_J^R])$$

and the associated well-formed expression is $DP_1 : \emptyset$. Similarly, the description of the behaviour of two philosophers is given by the process

$$DP_2 = \eta^{L,\Phi(I)} \star ((P \star F[\Phi_J^R]) \star ((P \star F[\Phi_J^R])[\Phi]))$$

for the relabeling satisfying $\Phi(L) = I$ and $\Phi(I)$ a new channel (i.e., $\Phi(I) \neq L$), and the associated well-formed expression is $DP_2 : \emptyset$. The effect of the η is to couple the left hand of the first philosopher to the rightmost fork of the other.

3 The cospan-span formalism

As mentioned in the Introduction, in the cospan-span model a system is represented by a graph of states and transitions, with extra structure required for capturing two operations on systems: *parallel composition with communication* requires *interfaces*, and *sequential composition* requires *conditions*.

We distinguish between *left* (often *input*) and *right* interfaces (often *output*). An interface is represented by a labeling of the graph arcs (the transitions of the transition system) on a suitable alphabet: when an arc appears in the graph (i.e., a transition occurs in the transition system), the corresponding label occurs on the interface. Interfaces allow to describe both communicating and non-communicating parallel composition (*restricted product* denoted $\cdot$, and *free product* denoted $\times$) of transition systems; in both cases, a state is a pair of states, and a transition is a pair of transitions, one for each transition system, which agree (synchronize) on the common interface.

Very often the state space of a transition system decomposes into a disjoint sum of *cases*, usually relevant to activating (creating) or disactivating (destroying) sub-systems. Any such *condition* is a subset of the set of states and it represents states in which the configuration may change in a particular way. However it is *crucial* not to think of conditions just as either initial or terminal states. It may sound reasonable in sequential programming, but when there are several active processes, one of those may die in a particular terminal state while the others are in general active – that is, the global state of the system in which a change of configuration occurs is a terminal state in only one component. To permit a change of configuration in only one component of a system it is crucial to allow for the whole state space among both *in-* and *out-conditions*. With the structure of in- and out-conditions, the *restricted sum* (denoted $+$) of transition systems can be defined, which expresses the deactivation of the first transition system in one of its out-conditions, followed by the activation of the second in one of its in-conditions.

3.1 Graphs

A *graph* **G** is a set G_0 of vertices and a set G_1 of (directed) arcs, together with two functions $d_0, d_1 : G_1 \to G_0$ which specify the source and target, respectively, of each arc. A *morphism* from **G** to **H** consists of a function from vertices to vertices, and a function from arcs to arcs which respects source and target; an isomorphism is a morphism for which both functions are bijections. [1]

3.2 Cospans of spans of graphs

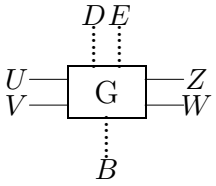
Definition 3.1 *A* cospan of spans of graphs $\mathcal{G}$ *(called for simplicity a* transition system*) consists of a graph* **G**, *four sets* X, Y, A, B *and four functions*

$$\partial_0 : G_1 \to X,\ \partial_1 : G_1 \to Y,$$
$$\gamma_0 : A \to G_0,\ \gamma_1 : B \to G_0.$$

Both ∂_0, ∂_1 may be thought of as *labeling* the arcs of **G** in the alphabets X, Y, respectively, and in fact they may be considered to be graph morphisms from **G** to one vertex graphs. These labelings are used in the restricted product of two transition systems, the operation which expresses communicating parallel processes. Alternatively, one may think of the vertices and arcs of **G** as the *states* and *transitions* of the system, whereas the elements of X, Y are the transitions of the interfaces. We call X the *left interface*, and Y the *right interface* – transition systems communicate through these interfaces.

The set A represents a *condition* on the states in which the transition system may come into existence, and the set B a condition in which it may cease to exist. We call A the *in-condition* of the transition system, and B the *out-condition*. The functions γ_0, γ_1 of a transition system will be used in the restricted sum of transition systems - an operation which expresses change of configuration of processes.

There is a useful diagrammatic representation of transition systems (see also [13]). For example, we represent a transition system with left interface $U \times V$, right interface $Z \times W$, in-condition $D + E$, and out-condition B, by



[1] We shall use the following terminology: given two sets X and Y, their cartesian product is denoted $X \times Y$, and the projections $pr_X : X \times Y \to X$, $pr_Y : X \times Y \to Y$; similarly, their sum (disjoint union) is denoted $X + Y$, and the injections $in_X : X \to X + Y$, $in_Y : Y \to X + Y$.

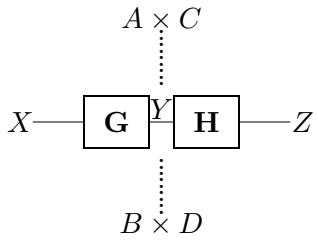
For the sake of simplicity we often use the same names ∂_0, ∂_1, γ_0, γ_1 for the four functions of any transition system when there is no risk of confusion, or we drop them altogether, introducing instead further suffixes when clarification is needed. We use symbols $X, Y, Z, U, V, W...$ for the (left and right) interfaces, and symbols $A, B, C, D, E, F, I, ...$ for the (in- and out-) conditions.

3.2.1 Parallel composition

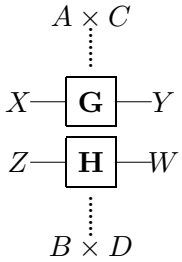
Given two transition systems $\mathcal{G} = (\mathbf{G}, X, Y, A, B)$ and $\mathcal{H} = (\mathbf{H}, Y, Z, C, D)$, the *restricted product* (communicating parallel composition) of $\mathcal{G}$ and $\mathcal{H}$, denoted $\mathcal{G} \cdot \mathcal{H}$, is the transition system whose set of vertices is $G_0 \times H_0$ and whose set of arcs is that subset of $G_1 \times H_1$ consisting of pairs of arcs (g, h) such that $\partial_1(g) = \partial_0(h)$. The interfaces and conditions of $\mathcal{G} \cdot \mathcal{H}$ are $X, Z, A \times C, B \times D$; the four functions are

$$\partial_{0,\mathcal{G}\cdot\mathcal{H}}(g,h) = \partial_{0,\mathcal{G}}(g),\ \partial_{1,\mathcal{G}\cdot\mathcal{H}}(g,h) = \partial_{1,\mathcal{H}}(h),$$
$$\gamma_{0,\mathcal{G}\cdot\mathcal{H}} = \gamma_{0,\mathcal{G}} \times \gamma_{0,\mathcal{H}}\ ,\ \gamma_{1,\mathcal{G}\cdot\mathcal{H}} = \gamma_{1,\mathcal{G}} \times \gamma_{1,\mathcal{H}}\ .$$

Diagrammatically we represent the restricted product as follows



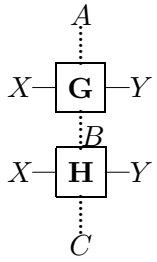
Closely related is the free product of transition systems. Given two transition systems $\mathcal{G} = (\mathbf{G}, X, Y, A, B)$ and $\mathcal{H} = (\mathbf{H}, Z, W, C, D)$, the *free product* (parallel composition with no communication) of $\mathcal{G}$ and $\mathcal{H}$, denoted $\mathcal{G} \times \mathcal{H}$, is the transition system diagrammatically represented as follows



Ignoring functions γ_0, γ_1, the restricted product of transition systems is the *span composition* of [13] and the free product is the *tensor product* of the corresponding spans of graphs. See there for some examples of how these operations may be used to model concurrent systems.

3.2.2 Sums

Given two transition systems $\mathcal{G} = (\mathbf{G}, X, Y, A, B)$ and $\mathcal{H} = (\mathbf{H}, X, Y, B, C)$, the *restricted sum* (change of configuration) of $\mathcal{G}$ and $\mathcal{H}$, denoted $\mathcal{G} + \mathcal{H}$, is the transition system diagrammatically represented as follows



The intended interpretation is that initially only the process $\mathcal{G}$ exists; when a state in B is reached the process $\mathcal{G}$ may die and the process $\mathcal{H}$ be created.

Similarly, given two transition systems $\mathcal{G} = (\mathbf{G}, X, Y, A, B)$ and $\mathcal{H} = (\mathbf{H}, X, Y, C, D)$, their *unrestricted sum* $\mathcal{G} \oplus \mathcal{H} = (\mathbf{G} \oplus \mathbf{H}, X, Y, A + C, B + D)$ is obviously defined.

3.2.3 Feedbacks

Given a transition system $\mathcal{G} = (\mathbf{G}, X \times Y, Z \times Y, A, B)$, the *product feedback* of $\mathcal{G}$ with respect to Y, denoted $\mathsf{Pfb}_Y(\mathcal{G})$, is the transition system whose set of vertices is G_0 and whose set of arcs is that subset of G_1 consisting of arcs g such that $(pr_Y \circ \partial_1)(g) = (pr_Y \circ \partial_0)(g)$. The interfaces and conditions of $\mathsf{Pfb}_Y(\mathcal{G})$ are X, Z, A, B, with the four functions defined as follows:

$$\partial_{0,\mathsf{Pfb}_Y(\mathcal{G})} = pr_X \circ \partial_{0,\mathcal{G}},\ \partial_{1,\mathsf{Pfb}_Y(\mathcal{G})} = pr_Z \circ \partial_{1,\mathcal{G}},$$
$$\gamma_{0,\mathsf{Pfb}_Y(\mathcal{G})} = \gamma_{0,\mathcal{G}},\quad \gamma_{1,\mathsf{Pfb}_Y(\mathcal{G})} = \gamma_{1,\mathcal{G}}.$$

Given a transition system $\mathcal{G} = (\mathbf{G}, X, Y, A + B, C + B)$, the *sum feedback* of $\mathcal{G}$ with respect to $\mathcal{B}$, denoted $\mathsf{Sfb}_B(\mathcal{G})$, is the transition system whose set of arcs is G_1 and whose set of vertices is $G_0/\sim$, i.e, G_0 quotiented by the relation $(\gamma_1 \circ in_B)(b) \sim (\gamma_0 \circ in_B)(b)$ (for all $b \in B$). The interfaces and conditions of $\mathsf{Sfb}_B(\mathcal{G})$ are X, Y, A and C, and the four functions are defined as follows:

$$\partial_{0,\mathsf{Sfb}_B(\mathcal{G})} = \partial_{0,\mathcal{G}},\ \partial_{1,\mathsf{Sfb}_B(\mathcal{G})} = \partial_{1,\mathcal{G}},$$
$$\gamma_{0,\mathsf{Sfb}_B(\mathcal{G})} = \gamma_{0,\mathcal{G}} \circ in_A,\quad \gamma_{1,\mathsf{Sfb}_B(\mathcal{G})} = \gamma_{1,\mathcal{G}} \circ in_C.$$

The diagrammatic representation of $\mathsf{Pfb}_Y(\mathcal{G})$ involves joining the right interface Y to the left interface Y; and, similarly, the diagrammatic representation of $\mathsf{Sfb}_B(\mathcal{G})$ involves joining the out-condition B to the in-condition B.

Remark 3.2 We described here only the principal operations, though there are a variety of useful special constants described in [13,14], including a diagonal component which allows synchronization between multiple components.

Notice also that clearly any *finite* graph labeled in $X \times Y$ may be obtained as an expression using only the *sum* operations of the cospan-span model in constant graphs which have at most three states and at most one transition. Further any expression in the cospan-span model involving finite constants is again a finite transition system. To describe infinite transition systems *recursive equations* over cospan-span expressions must be added, see e.g. [15].

3.3 Denotational encoding of SPA

We present a denotational semantics of well-formed expressions, by means of (labeled, directed) graphs. We actually encode processes into a sub-class of graphs, often denoted in the literature as graphs with no 'horizontal sharing' (i.e., such that between two nodes there is at most one cycle-free path).

We first need an assumption: since SPA does not make an a priori choice of left and right channels (which results in the main operation being, in engineering terms, 'soldering' rather than 'series composition') we arbitrarily choose of taking *all* channels to be on the *left*.

We denote the semantics of a well-formed expression $P : \sigma$ by $\mathsf{Sem}(P)$: it represents a (possibly cyclic) graph, whose arcs are labeled on the left by tuples of actions as indicated by the inference rules and on the right by τ. We view $\mathsf{Sem}(P)$ as a transition system labeled by $(\prod_{A\in\sigma} L_A) \times \{\tau\}$, with in-condition the root vertex, and out-condition the set of free variables in P.

In order to define the mapping, we need to choose some constants. First, for sets: S_σ is the set $\prod_{A\in\sigma} L_A$, for each σ type. Then, for transition systems: T_μ is the transition system with two states and a transition labelled (μ, τ) among them (and $\{\mathsf{0}\}$ as the obvious in- and out-condition); $proj_A$ is the transition system with one state and $\mid L_A \times (\prod_{B\in\sigma} L_B) \mid$ transitions, with transition $(a, b_1, \ldots, b_n)$ labeled on the left by $(b_1, \ldots, b_n)$ and on the right by $(a, b_1, \ldots, b_n)$;[2] the transition system η_A has one state and $\mid L_A \mid$ transitions, with a labeled on the left by τ, and on the right by (a, a).

We may now describe the operators of SPA in terms of the operations of the cospan-span model. It is clear that only a restricted version of recursion is codeable in the basic operations of the cospan-span formalism: The denotational encoding $\mathsf{Sem} : \mathcal{SP}_X \to \mathcal{G}$ maps each well-formed expression *with sequential recursion* (i.e., an expression where a free variable may occur only in the scope of either a summation or a recursion operator) into a transition system with $\{\mathsf{0}\}$ as in-condition, $\{\tau\}$ as right interface and S_σ as left-interface.

[2] That is, $proj_A$ has fewer channels on the left than on the right, and will be used to cut channels: we have just represented the projection function $A \times (\prod_{B\in\sigma} B) \to (\prod_{B\in\sigma} B)$ as a transition system. In a similar way any finite relation, including the relabeling isomorphisms, and the opposite relation Δ^o_A of the diagonal function, may be represented as a transition system with only one state – we will use the same name for the relation and the corresponding transition system.

Definition 3.3 (model mapping) *The encoding* $\mathsf{Sem} : \mathcal{SP}_X \to \mathcal{G}$ *is inductively defined by the following set of of inference rules.*

(i) Let σ *be a set of channels. Then*

$$\mathsf{Sem}(0 : \sigma) = (\{0\}, S_\sigma, \{\tau\}, \{0\}, \emptyset) \qquad \mathsf{Sem}(x : \sigma) = (\{x\}, S_\sigma, \{\tau\}, \{0\}, \{x\}).$$

(ii) Let $P_i : \sigma$ *for* $i \in I$ *be well-formed expressions, and* μ_i *for* $i \in I$ *be actions for the channels in* σ*. Then*

$$\mathsf{Sem}(\sum_{i \in I} \mu_i P_i) = D_I + \oplus_{i \in I}(T_{\mu_i} + \mathsf{Sem}(P_i)),$$

where D_I *is the transition system with one state,* $\{0\}$ *as in-condition and* $\sum_{i \in I}\{0\}$ *as out-conditions, and* $\oplus_{i \in I}(T_{\mu_i} + \mathsf{Sem}(P_i))$ *is the transition system obtained by the unrestricted sum of the* $T_{\mu_i} + \mathsf{Sem}(P_i)$*'s, with the free variables of the processes as out-conditions.*[3]

(iii) Let $P : \sigma \cup \{A\}$ *be a well-formed, closed expression. Then*

$$\mathsf{Sem}((\nu A)P) = proj_A \cdot \mathsf{Sem}(P).$$

(iv) Let $P : \sigma \cup \{A, B\}$ *be a well-formed, closed expression. Then*

$$\mathsf{Sem}([B \Rightarrow A]P) = \mathsf{Pfb}_A((\Delta_A^o \times \eta_B) \cdot (\mathsf{Sem}(P) \times \phi_{B,A})).$$

(v) Let $P : \sigma$ *and* $Q : \rho$ *be well-formed, closed expressions. Then*

$$\mathsf{Sem}(P \| Q) = \mathsf{Sem}(P) \times \mathsf{Sem}(Q).$$

(vi) Let $P : \sigma$ *be a well-formed expression. Then*

$$\mathsf{Sem}(rec\, x.P) = \mathsf{Sfb}_{\{x\}}((\{0\}, S_\sigma, \{\tau\}, \{0\} + \{x\}, \{0\}) + \mathsf{Sem}(P)).$$

It is clear that the graph associated to a well-formed expression $P : \sigma$ is not isomorphic to the fragment of the transition system $\mathcal{T}_\sigma$ reachable from P. Nevertheless, a tight correspondence holds among the two models.

Proposition 3.4 (model correspondence) *Let* P *be a closed process with sequential recursion, and* $P : \sigma$ *a well-formed expression. Then* $\mathsf{Sem}(P)$ *and the fragment of* $\mathcal{T}_\sigma$ *reachable from* P *are trace equivalent.*

The proof is given by induction, defining a mapping from $\mathsf{Sem}(P)$ to $\mathcal{T}_\sigma$ that coalesces nodes corresponding to different instances of the same process.

[3] Of course, note that the collapsing of some nodes is required: we leave to the reader the task of providing the additional constants needed for performing such a collapsing.

4 The tile logic

4.1 Building States

We introduce an inductive presentation for *Conway theories* (see e.g. in [2] for their definition). With respect to the better-known algebraic theories, they are equipped with a parametric operator, †, allowing to capture also recursive definitions. We spell out only the definition for one-sorted signatures, with (underlined) natural numbers as sorts, the general instance of many-sorted being a straightforward extension.

Our presentation of Conway theories is original, even if it relies on similar descriptions for algebraic and weaker theories [4]. Some of them are mentioned later on, and used for our encoding of SPA at the end of the section.

Definition 4.1 (Conway theories) *Let Σ be a one-sorted signature. The set of arrows of the* Conway theory $\mathbf{C}(\Sigma)$ *is generated by the inference rules*

$$\text{(generators)}\ \frac{f \in \Sigma_{n,m}}{f : \underline{n} \to \underline{m}} \qquad \text{(pairing)}\ \frac{s : \underline{n} \to \underline{m}, t : \underline{n'} \to \underline{m'}}{s \otimes t : \underline{n+n'} \to \underline{m+m'}}$$

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

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

$$\text{(permutations)}\ \frac{n,m \in \mathbb{N}}{\rho_{\underline{n},\underline{m}} : \underline{n+m} \to \underline{m+n}} \qquad \text{(dagger)}\ \frac{s : \underline{n+m} \to \underline{m}}{s^\dagger : \underline{n} \to \underline{m}}$$

Moreover, the composition operator is associative, the pairing operator is associative with $id_{\underline{0}}$ the neutral element of the resulting monoid of arrows, and the monoidality axiom $id_{\underline{n+m}} = id_{\underline{n}} \otimes id_{\underline{m}}$ holds for all $n, m \in \mathbb{N}$. In addition, the monoid of arrows satisfies the functoriality *axiom $(s \otimes t);(s' \otimes t') = (s;s') \otimes (t;t')$ (whenever both sides are defined) and the* identity *axiom $id_{\underline{n}};s = s = s;id_{\underline{m}}$ for all $s : \underline{n} \to \underline{m}$; the* monoidality *axioms*

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

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

for all $n, m, l \in \mathbb{N}$; 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 $n, m \in \mathbb{N}$; the naturality *axioms*

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

for all $s : \underline{n} \to \underline{m}, t : \underline{l} \to \underline{k}$*; and finally, the* Conway *axioms*

$$((s \otimes id_{\underline{m}}); t)^\dagger = s; t^\dagger \qquad (u^\dagger)^\dagger = ((id_{\underline{n}} \otimes \nabla_{\underline{m}}); u)^\dagger$$

$$(\nabla_{\underline{l+n}}; (id_{\underline{l}} \otimes !_{\underline{n}} \otimes v); w)^\dagger = (\nabla_{\underline{l}}; (id_{\underline{l}} \otimes (\nabla_{\underline{l+m}}; (id_{\underline{l}} \otimes !_{\underline{m}} \otimes w); v)^\dagger); w)$$

for all $s : \underline{n} \to \underline{l}$, $t : \underline{l+m} \to \underline{m}$, $u : \underline{n+m+m} \to \underline{m}$, $v : \underline{l+n} \to \underline{m}$, $w : \underline{l+m} \to \underline{n}$.

We refer to *graph* theories $G(\Sigma)$ for the arrows obtained by using only the generators, pairing and identities rules; to *monoidal* theories $\mathbf{M}(\Sigma)$ if the composition rule is added; and to *algebraic* theories $\mathbf{A}(\Sigma)$ if all rules, except for dagger, are considered.

A Conway theory $\mathbf{C}(\Sigma)$ is just an instance of a monoidal category. It can be considered as categorical folklore the fact that the cartesian product canonically induces a monoidal product, together with a family of natural transformations, often denoted as *diagonals* and *projections*. Then, our definitions of Conway and algebraic theories are equivalent to the standard ones: a classical result relates algebraic theories and the usual term algebra construction.

Proposition 4.2 (algebraic theories and term algebras) *Given a one-sorted signature* Σ*, for all* $n, m \in \mathbb{N}$ *there exists a one-to-one correspondence between the set of arrows with source* $\underline{n}$ *and target* $\underline{m}$ *of* $\mathbf{A}(\Sigma)$ *and the* m*-tuples of elements of the term algebra –over a set of* n *variables– associated to* Σ*.*

4.2 Describing Systems

We recall now the basic definitions for tile logic [10], a general framework for the specification of rule-based systems, in the vein of both the *rewriting logic* formalism [16] and the sos approach [17]. Intuitively, it extends the standard definition for *rewriting systems*: a rule is seen as a *module* (kind of a basic component of a system) carrying information (equivalently, expressing conditions) on the possible behaviour of its sub-components (that is, of the terms to which it can be applied).

Definition 4.3 (algebraic rewriting systems) *An* algebraic rewriting system (ARS) $\mathcal{R}$ *is a four-tuple* $\langle \Sigma_\sigma, \Sigma_\tau, N, R \rangle$*, where* $\Sigma_\sigma, \Sigma_\tau$ *are signatures,* N *is a set of (rule) names and* R *is a function* $R : N \to \mathbf{A}(\Sigma_\sigma) \times G(\Sigma_\tau) \times G(\Sigma_\tau) \times \mathbf{A}(\Sigma_\sigma)$ *such that for all* $d \in N$*, with* $R(d) = \langle l, a, b, r \rangle$*, we have* $l : \underline{n} \to \underline{m}, r : \underline{p} \to \underline{q}$ *iff* $a : \underline{n} \to \underline{p}, b : \underline{m} \to \underline{q}$*.*

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

$$\begin{array}{ccc} n & \xrightarrow{\ l\ } & m \\ {\scriptstyle a}\downarrow & d & \downarrow{\scriptstyle b} \\ p & \xrightarrow[\ r\ ]{} & q \end{array}$$

making explicit the source and the target of the operators.

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

Definition 4.4 (the tile logic) *Let* $\mathcal{R} = \langle \Sigma_\sigma, \Sigma_\tau, N, R \rangle$ *be an* ARS. *We say that* $\mathcal{R}$ entails *the set* $T(\mathcal{R})$ *of* Conway sequents *obtained by a finite number of applications of the following set of* inference rules*: The* basic rules

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

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

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

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

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

and finally, the auxiliary rules

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

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

The basic rules provide the generators of the sequents, together with suitable identity arrows, whose intuitive meaning is that an element of $\mathbf{C}(\Sigma_\sigma)$ or $\mathbf{M}(\Sigma_\tau)$ stays idle during a rewrite. The composition rules express the way in which sequents can be combined, either sequentially (*vert*), or executing them in parallel (*par*), or nesting one inside the other (*hor*). The auxiliary rules are the counterpart of the auxiliary operators in Definition 4.1: they provide a way of permutating (*perm*) two sequents, and either duplicating (*dupl*) or discharging (*dis*) a sequent. No additional rule corresponds to the dagger operators of Conway theories: we do not want to close sequents with respect to such operators, but just to be able to unfold a recursive definition, and to this end the (*h-refl*) rule is all that is needed.

4.3 An observational semantics

An abstract semantics could be recovered by providing a set of axioms for proof terms, so that an equivalence class would correspond to a normalized proof: see e.g. [11], where a comparison is traced with *concurrent computations* in rewriting logic. Nevertheless, the two spatial dimensions of a sequent –horizontal for source and target, vertical for effect and trigger– hardly play the same role. When introducing tiles, we referred to source and target as *states* of our system, and to trigger and effect as *conditions* to be verified, before applying a rule. It seems then perspicuous to discuss a semantics over states, which is only observational, identifying states that show the same behaviour on the input (trigger) and output (effect) components. To this end, we simplify the structure of sequents, dropping the proof term, thus recovering a generalized notion of transition system.

Definition 4.5 (tile transition system) *Let $\mathcal{R} = \langle \Sigma_\sigma, \Sigma_\tau, N, R \rangle$ be an* ARS. *The associated* (Conway) tile transition system *is the relation $Tr(\mathcal{R}) \subseteq \mathbf{C}(\Sigma_\sigma) \times \mathbf{M}(\Sigma_\tau) \times \mathbf{M}(\Sigma_\tau) \times \mathbf{C}(\Sigma_\sigma)$ obtained dropping the first component from the relation $T(\mathcal{R})$.*

Abusing notation, we refer to a four-tuple $\langle s, a, b, t \rangle$ as a sequent entailed by an ARS. Restricting our attention to transition systems allows us to define a suitable notion of behavioural equivalence by means of *trace equivalence*.

Definition 4.6 (tile trace equivalences) *Let $\mathcal{R} = \langle \Sigma_\sigma, \Sigma_\tau, N, R \rangle$ be an* ARS. *Two terms $s, t \in \mathbf{C}(\Sigma_\sigma)$ are* tile trace equivalent *in $\mathcal{R}$ (denoted $s \equiv t$) if for any sequent $s \xrightarrow[b]{a} s'$ entailed by $\mathcal{R}$ a corresponding sequent $t \xrightarrow[b]{a} t'$ is entailed, and vice versa.*

4.4 Operational encoding of SPA

For representing the semantics of SPA by a tile system, we need a signature Σ_{spa} such that each process may be embedded into a term in $\mathbf{C}(\Sigma_{\text{spa}})$.

Definition 4.7 (many-sorted signature for SPA) *The many-sorted signature Σ_{spa} associated to* SPA *has 2^U as set of sorts, and it contains the constants $nil_\sigma : \epsilon \to \sigma$, denoting the* inactive *states; the unary operators* prefix $\{\mu_\sigma : \sigma \to \sigma\}$, restriction $\{(\nu A)_\sigma : \sigma \cup \{A\} \to \sigma \backslash \{A\}\}$ *and* merging $\{[B \Rightarrow A]_\sigma : \sigma \cup \{A, B\} \to (\sigma \backslash \{B\}) \cup \{A\}\}$*; and finally, the binary operators* non-deterministic choice $\{+_\sigma : \sigma \otimes \sigma \to \sigma\}$ *and* parallel composition $\{\|_{\sigma,\gamma} : \sigma \otimes \gamma \to \sigma \cup \gamma\}$ *for $\sigma \cap \gamma = \emptyset$.*

We often omit the subscript, whenever clear from the context. The reader may immediately recover for a well-formed, closed expression $P : \sigma$ the embedding $\{\!|P|\!\} : \epsilon \to \sigma$, obtaining an arrow of the Conway theory associated to Σ_{spa}. Note that recursion is simulated by the $\dagger$ operator. For example, let us consider the processes F_L, $F_L[\Phi_I^J]$ and F, as defined in Section 2.3.3: the associated terms are respectively $\{\!|F_L|\!\} = ((\{\mathsf{u}_I, \tau_J\} \otimes \{\tau_I, \tau_J\}); +)^\dagger; \{\mathsf{l}_I, \tau_J\}$,

$\{\!|F_L[\Phi_I^J]|\!\} = ((\{\mathsf{u}_J, \tau_I\} \otimes \{\tau_J, \tau_I\}); +)^\dagger; \{\mathsf{l}_J, \tau_I\}$, and

$\{\!|F|\!\} = (\nabla_{\{I,J\}}; (\nabla_{\{I,J\}} \otimes id_{\{I,J\}}); (\{\tau_I, \tau_J\} \otimes t_{F_L} \otimes t_{F_L[\Phi_I^J]}); (+ \otimes id_{\{I,J\}}); +)^\dagger$.

The operational semantics of SPA is given by the transition system $\mathcal{T}_{\mathrm{spa}}$, presented in Section 2 by a set of rules, in a sos style. A similar description may also be obtained for tiles, by means of their inference mechanism.

Definition 4.8 (tile system for SPA) *The* ARS $\mathcal{R}_{\mathrm{spa}}$ *for* SPA *is the 4-tuple* $\langle \Sigma_{\mathrm{spa}}, \Sigma_{\mathrm{spa}}, L_{\mathrm{spa}}, R_{\mathrm{spa}} \rangle$*, where* R_{spa} *is the set of labeled rules given below*

$$act : \mu \xrightarrow[\mu]{id_\sigma} id_\sigma \qquad res_A : (\nu A) \xrightarrow[\mu]{\mu \cup \{a\}} (\nu A) \quad \text{for } a \in L_A$$

$$+_l : + \xrightarrow[\mu]{\mu \otimes id_\sigma} id_\sigma \otimes !_\sigma \qquad +_r : + \xrightarrow[\mu]{id_\sigma \otimes \mu} !_\sigma \otimes id_\sigma \qquad par : || \xrightarrow[\mu \cup \gamma]{\mu \otimes \gamma} ||$$

$$mer_{B,A} : [B \Rightarrow A] \xrightarrow[(\mu \setminus \{b\}) \cup \{\phi_{B,A}(b)\}]{\mu \cup \{\phi_{B,A}(b), b\}} [B \Rightarrow A] \quad \text{for } b \in L_B$$

Except for recursion, there is one tile for each inference rule of SPA: they are parametric with respect to types, since the corresponding rules are so. The effect μ indicates that a process is 'running', outputting a label μ.

Proposition 4.9 (interleaving correspondence) *Let P and Q be closed* SPA *processes, and $P : \sigma$ a well-formed expression.* (1) *If a transition $P \xrightarrow{\mu} Q$ is entailed by the* SPA *transition system $\mathcal{T}_{\mathrm{spa}}$, then a sequent $\{\!|P|\!\} \xrightarrow[\mu]{id_\epsilon} \{\!|Q|\!\}$ is entailed by the tile system $\mathcal{R}_{\mathrm{spa}}$. Vice versa,* (2) *if a sequent $\{\!|P|\!\} \xrightarrow[\mu]{id_\epsilon} t$ is entailed by the tile system $\mathcal{R}_{\mathrm{spa}}$, then there exists a process Q such that a transition $P \xrightarrow{\mu} Q$ is entailed by the* SPA *transition system $\mathcal{T}_{\mathrm{spa}}$ and $t = \{\!|Q|\!\}$.*

5 Sketching the correspondence

How to compare the two encodings? On the one hand, the cospan-span model associates to each well-formed expression $P : \sigma$ (for P a closed process with sequential recursion) a transition system $\mathsf{Sem}(P)$, with no out-conditions and a single in-condition, whose abstract semantics is represented by the set of paths of the underlying graph. On the other hand, in the tile logic approach a set of tiles is associated to the set of inference rules of SPA, and transitions are mimicked by rewrites. Thus, a first attempt is to assume that these two sets coincide: the result stated below is an immediate consequence of the properties of the two encodings (namely, see Proposition 3.4 and Proposition 4.9).

Proposition 5.1 (on paths and rewrites) *Let P be a closed process with sequential recursion, and $P : \sigma$ a well-formed expression. Then,* (1) *for each path on $\mathsf{Sem}(P)$, mapped into a sequence of labels $\mu_1 \dots \mu_n$, there exists a closed process Q such that a tile $\{\!|P|\!\} \xrightarrow[\mu_1;\dots;\mu_n]{id_\epsilon} \{\!|Q|\!\}$ is entailed by $\mathcal{R}_{\mathrm{spa}}$; and vice versa* (2)*, for each tile $\{\!|P|\!\} \xrightarrow[\mu_1;\dots;\mu_n]{id_\epsilon} t$ entailed by $\mathcal{R}_{\mathrm{spa}}$, there exists a path on $\mathsf{Sem}(P)$, mapped into a sequence of labels $\mu_1 \dots \mu_n$.*

There are more paths than tiles: for example, given the process $P = \mu.0 + \mu.0$, $\mathsf{Sem}(P)$ has two paths labeled μ, corresponding to the tile $\{\!|P|\!\} \xrightarrow[\mu]{id_\epsilon} 0$. Nevertheless, the previous result may be enriched with a denotational flavour.

Theorem 5.2 (on trace semantics) *Let P and Q be closed processes with sequential recursion, and $P : \sigma$ and $Q : \sigma$ well-formed expressions. Then, $\mathsf{Sem}(P)$ and $\mathsf{Sem}(Q)$ are trace equivalent if and only if $\{\!|P|\!\} \equiv \{\!|Q|\!\}$.*

6 Conclusions and further work

In the paper we introduced a simple process algebra, and we encoded it into both the cospan-span model and the tile logic formalism. Then, we proved a correspondence among the two encodings, by means of trace equivalence.

The algebra we considered is quite powerful, since it offers an effective calculus for describing finite state systems. It is yet rather simplistic, with respect to the possibilities offered by the tools of both the cospan-span model and the tile logic formalism. In fact, its name reveals that we consider the operators of the calculus quite close to those of the span(Graph) model: the operators available on cospan-span(Graph) are much richer, and we plan to investigate a calculus which supports the same operations of that model.

Our choice of the SPA algebra, and our restriction to a behavioural semantics such as trace equivalence, instead of e.g. the more expressive bisimulation, is on purpose. A direct translation of cospan-spans into tiles would be by no means straightforward, since the composition operations of the first model only bear a similarity with those of double-categories, which are the semantic domain of tiles. Besides the technical points, intuitively the two dimensions of cospan-span expressions are spatial (the horizontal dimension being the underlying geometry of the system, the vertical dimension being the space of states), while instead the vertical dimension in tiles represents the flow of time. Thus, even if our case study does not allow for properly testing the expressiveness of the two formalisms, restricting our attention to an algebra which is easily encoded in both meta-frameworks focuses the paper on the comparison among them, highlighting the different tools they offer for system specification (in a similar fashion to what has been already done for tiles and other rewrite-based formalisms in [11]).

We then feel confident that we were able to cast some light on the correspondence between the two approaches. Roughly, each expression in the cospan-span model is a system, and each operation is a system constructor; on the contrary, a tile is a (open) state transformer, and each tile expression represents (the encoding of) a possible evolution from one state into another. The dichotomy denotational vs. operational that we mentioned in the Introduction is thus justified, even if, of course, also denotational models for tiles are at hand, as long as we analyze formalisms such as the calculus SPA, where processes univocally denote also states.

Summing up, on the one hand, the cospan-span model offers a rich algebra for the description of systems, and its operations thus allow for a powerful denotational formalism, on which the orthogonal aspects of parallel and sequential composition of systems are of pivotal importance. On the other hand, tile logic offers a flexible tool for the specification of rule-based formalisms: the emphasis is less on the structure of a system, which can actually be considered as parametric, and much more on the the inference mechanism for building single transitions out of an initial set, and sequentially compose them.

Acknowledgement

We wish to thank Roberto Bruni for the interesting discussions and the careful reading of the paper, as well as the referees for their valuable suggestions.

References

[1] A. Arnold. *Finite transition systems*. Prentice Hall, 1994.

[2] S. Bloom and Z. Ésik. *Iteration Theories*. EATCS Monographs on Theoretical Computer Science. Springer, 1993.

[3] R. Bruni, D. de Frutos-Escrig, N. Martí-Oliet, and U. Montanari. Bisimilarity congruences for open terms and term graphs via tile logic. In C. Palamidessi, editor, *Concurrency Theory*, volume 1877 of *Lect. Notes in Comp. Science*, pages 259–274. Springer, 2000.

[4] R. Bruni, F. Gadducci, and U. Montanari. Normal forms for algebras of connections. *Theoret. Comput. Sci.*, 2002. To appear. Available at `http://www.di.unipi.it/~ugo/tiles.html`.

[5] R. Bruni, J. Meseguer, and U. Montanari. Executable tile specifications for process calculi. In J.-P. Finance, editor, *Fundamental Approaches to Software Engineering*, volume 1577 of *Lect. Notes in Comp. Science*, pages 60–76. Springer, 1999.

[6] R. Bruni and U. Montanari. Cartesian closed double categories, their lambda-notation, and the pi-calculus. In *Logic in Computer Science*, pages 246–265. IEEE Computer Society Press, 1999.

[7] R. Bruni, U. Montanari, and F. Rossi. An interactive semantics of logic programming. *Theory and Practice of Logic Programming*, 2001.

[8] R. Bruni, U. Montanari, and V. Sassone. Open ended systems, dynamic bisimulation and tile logic. In J. van Leeuwen, O. Watanabe, M. Hagiya, P. D. Mosses, and T. Ito, editors, *IFIP Theoretical Computer Science*, volume 1872 of *Lect. Notes in Comp. Science*, pages 440–456. Springer, 2000.

[9] G. Ferrari and U. Montanari. Tile formats for located and mobile systems. *Information and Computation*, 156:173–235, 2000.

[10] 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, 2000.

[11] F. Gadducci and U. Montanari. Comparing logics for rewriting: Rewriting logic, action calculi and tile logic. *Theoret. Comput. Sci.*, 2002. To appear. Available at `http://www.di.unipi.it/~ugo/tiles.html`.

[12] P. Katis, N. Sabadini, and R.F.C. Walters. Bicategories of processes. *Journal of Pure and Applied Algebra*, 115:141–178, 1997.

[13] P. Katis, N. Sabadini, and R.F.C. Walters. SPAN(Graph): A categorical algebra of transition systems. In M. Johnson, editor, *Algebraic Methodology and Software Technology*, volume 1349 of *Lect. Notes in Comp. Science*, pages 307–321. Springer, 1997.

[14] P. Katis, N. Sabadini, and R.F.C. Walters. A formalization of the IWIM model. In A. Porto and G.-C. Roman, editors, *Coordination*, volume 1906 of *Lect. Notes in Comp. Science*, pages 267–283. Springer, 2000.

[15] P. Katis, N. Sabadini, and R.F.C. Walters. Recursion and concurrency. In A. Labella, editor, *Fixed Points in Computer Science*, 2001.

[16] J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoret. Comput. Sci.*, 96:73–155, 1992.

[17] G. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.