DEPARTAMENTO
DE MATEMÁTICA

INSTITUTO SUPERIOR TÉCNICO

DEPARTAMENTO DE MATEMÁTICA - INSTITUTO SUPERIOR TÉCNICO
AVENIDA ROVISCO PAIS - 1096 LISBOA CODEX
TEL. 351-1- 841 70 00 FAX. 351-1- 849 92 42

To Appear in the Proceedings of the
Fifth International Conference on Computer Aided System Technology,
EUROCAST 95, LNCS, Springer-Verlag

Refinement Mapping for General (Discrete Event) Systems Theory

P. Blauth Menezes†, J. Félix Costa†† and A. Sernadas†

† Departamento de Matemática, Instituto Superior Técnico
Av. Rovisco Pais, 1096 Lisboa Codex, Portugal - {blauth, acs}@raf.ist.utl.pt

†† Departamento de Informática, Faculdade de Ciências, Universidade de Lisboa
Campo Grande, 1700 Lisboa, Portugal - fgc@di.fc.ul.pt

Abstract. A categorial semantic domain for general (discrete event) systems based on labeled transition systems with full concurrency is constructed, where synchronization and hiding are functorial. Moreover, we claim that, within the proposed framework, a class of mappings stands for refinement. Then we prove that refinement satisfies the diagonal compositionality requirement, i.e., refinements compose (vertical) and distribute over system composition (horizontal).

1 Introduction

We construct a semantic domain for interacting systems which satisfies the diagonal compositionality requirement, i.e., refinements compose (vertically), reflecting the stepwise description of systems, involving several levels of abstraction, and distributes through combinators (horizontally), meaning that the refinement of a composite system is the composition of the refinement of its parts.

Taking into consideration the developments in Petri net theory (mainly with seminal papers like [17], [11] and [15]) it was clear that nets might be good candidates. However, most of net-based models such as Petri nets in the sense of [14] and labeled transition systems (see [12]) lack composition operations (modularity) and abstraction mechanisms in their original definitions. This motivate the use of the category theory: the approach in [17] provides the former, where categorical constructions such as product and coproduct stand for system composition, and the approach in [11] provides the later for Petri nets where a special kind of net morphism corresponds to the notion of implementation. Also, category theory provides powerful techniques to unify different categories of models (i.e., classes of models categorically structured) through adjunctions (usually reflections and coreflections) expressing the relation of their semantics as in [15].

We introduce the concept of (nonsequential) automaton as a kind of automaton structured on states and transitions. Structured states are "bags" of local states like tokens in Petri nets and structured transitions specify a concurrency relationship between component transitions in the sense of [3] and [7]. In [9] we show that nonsequential automata are more concrete then Petri nets (in fact, categories of Petri nets are isomorphic to subcategories of nonsequential automata) extending the approach in [15], where a formal framework for classification of models for concurrency is set.

The resulting category is bicomplete where the categorial product and coproduct stand for (system) composition. Synchronization and hiding are functorial operations. A synchronization restricts a (system) composition according to some given interaction specification. A view of a system is obtained through hiding of transitions introducing an internal nondeterminism. A hidden transition cannot be used for interaction.

A refinement mapping maps transitions into transactions reflecting an implementation of a system on top of another. It is defined as an automaton morphism where the target object is enriched with all conceivable sequential and nonsequential computations. Computations are induced by an endofunctor *tc* (transitive closure) and composition of refinements $\varphi: N_1 \to tcN_2$, $\psi: N_2 \to tcN_3$ is defined using Kleisli categories as illustrated in the Figure 1.

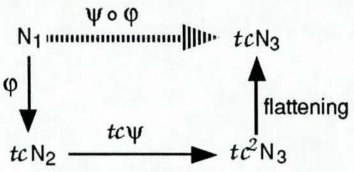

Fig. 1. Composition of refinements

Therefore, refinements compose, i.e., the vertical compositionality requirement is achieved. Moreover we find a general theory of refinement of (discrete) systems which also satisfies the horizontal compositionality requirement. i.e., for refinements $\varphi: N_1 \to tcM_1$, $\psi: N_2 \to tc\ M_2$, we have that:

$$\varphi N_1 \times \psi N_2 = \varphi \times \psi (N_1 \times N_2)$$

where $\varphi N_1 \times \psi N_2$ and $N_1 \times N_2$ are composed systems and the refinement $\varphi \times \psi$ is (uniquely) induced by φ and ψ.

Note that, while the vertical compositionality is easily achieved in several models, they lack horizontal compositionality (see [9] for Petri nets and [10] for transition systems).

2 Nonsequential Automata

A nonsequential automaton is a reflexive graph (a graph with an endoarc for every node) labeled on arcs such that nodes, arcs and labels are elements of commutative monoids. A reflexive graph represents the *shape* of an automaton where nodes and arcs stand for states and transitions, respectively, with endoarcs interpreted as *idle* transitions. The labeling procedure allows the occurrence of more then one transition with the same label. A structured transition specify a concurrency relation between component transitions. Comparing with asynchronous transition systems (first introduced in [3]), the independence relation of a nonsequential automaton is explicit in the graphical representation. A structured state can be viewed as a "bag" of local states where each local state can be viewed as a resource to be consumed or produced, like a token in Petri nets.

Nonsequential automata and its morphisms constitute a category which is complete and cocomplete with products isomorphic to coproducts. A product (or coproduct) can be viewed as (system) composition. In what follows $\mathcal{CMon}$ denotes the category of commutative monoids and suppose that k is in $\{0, 1\}$.

Definition 2.1 Nonsequential Automaton. A nonsequential automaton $N = \langle V, T, \partial_0, \partial_1, \iota, L, \mathsf{lab}\rangle$ is such that $T = \langle T, \|, \tau\rangle$, $V = \langle V, \oplus, e\rangle$, $L = \langle L, \|, \tau\rangle$ are $\mathcal{CMon}$-objects of transitions, states and labels respectively, $\partial_0, \partial_1: T \to V$ are $\mathcal{CMon}$-morphisms called source and target respectively, ι: $V \to T$ is a *CMon*-morphism such that $\partial_k \circ \iota = id_V$ and lab: $T \to L$ is a *CMon*-morphism such that lab(t) = τ whenever there is v in V where $\iota(v)$ = t. ❑

We may refer to a nonsequential automaton N = $\langle V, T, \partial_0, \partial_1, \iota, L, lab\rangle$ by N = $\langle G, L, lab\rangle$ where G = $\langle V, T, \partial_0, \partial_1, \iota\rangle$ is a reflexive graph internal to *CMon* (i.e., V, T are *CMon*-objects and ∂_0, ∂_1, ι are *CMon*-morphisms).

In an automaton, a transition labeled by τ represents a hidden transition (as we will see later, a hidden transition is encapsulated and therefore, can not be triggered from the outside). Note that, all idle transitions are hidden. The definition above is not extensional in the sense that two distinct transitions with the same label may have the same source and target states. In this paper we are not concerned with initial states.

A transition t such that $\partial_0(t)$ = X, $\partial_1(t)$ = Y is denoted by t: X $\to$ Y. Since a state is an element of a monoid, it may be denoted as a formal sum $n_1A_1 \oplus ... \oplus n_mA_m$, with the order of the terms being immaterial, where A_i is in V and n_i indicate the multiplicity of the corresponding (local) state, for i = 1...m. The denotation of a transition is analogous. We also refer to a structured transition as the *parallel composition* of component transitions. When no confusion is possible, a structured transition $x \| \tau$: $X \oplus A \to Y \oplus A$ where t: X $\to$ Y and ι_A: A $\to$ A are labeled by x and τ, respectively, is denoted by x: $X \oplus A \to Y \oplus A$. For simplicity, in graphical representation, we omit the endotransitions. A state $n_1A_1 \oplus ... \oplus n_mA_m$ and a labeled transition $n_1t_1 \| ... \| n_mt_m$ are graphically represented as in the Figure 2.

Fig. 2. Graphical representation of structured states and transitions

Example 2.2 The graphical representation of an automaton N = $\langle \{X, Y\}^{\oplus}, \{a, b, \iota_X, \iota_Y\}^{\|}, \partial_0, \partial_1, \iota, \{x, y\}^{\|}, lab\rangle$ with free monoids determined by the local transitions a: 2X $\to$ Y, b: 2X $\to$ Y and with labeling given by a $\mapsto$ x, b $\mapsto$ y is illustrated in the Figure 3. ❑

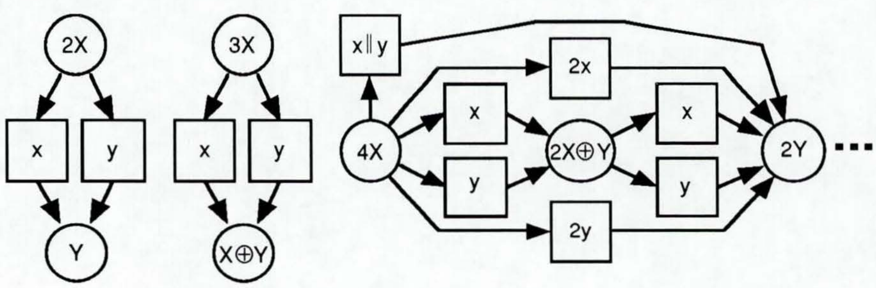

Fig. 3. Graphical representation of a nonsequential automaton

Considering the monoidal structure of nonsequential automata and since in this paper we are not concerned with initial states, the schema above has an infinite number of distributed diagrams. If an initial state is considered, only the corresponding diagram may be drawn. For instance, in the example above, if the initial state is 4X then the schema could be reduced to the rightmost diagram in the Figure 3.

Comparing the graphical representation with the one for Petri nets (see, e.g., [14]), in a nonsequential automaton all possible states are explicit while in Petri nets the reachable markings are implicit. Also, the concurrency relation between transitions in Petri nets is implicit. Both models, categories of Petri nets and categories of nonsequential automata can be unified through adjunctions. For details, see [9].

Remark 2.3 Non-Reflexive Automata. If we define the category of non-reflexive automata (with source, target and labeling preserving morphisms) the product construction reflects a composition operation with (total) synchronization in the sense that each transition of the first automaton is synchronized with all transitions of the second. This construction has very few practical applications. ❑

Remark 2.4 Structured Transition × Independence Square. Consider the Figure 4. Let $a: A \to B$, $x: X \to Y$ be two transitions of some automaton. Then, $a \| x: A \oplus X \to B \oplus Y$, $a: A \oplus X \to B \oplus X$, $a: A \oplus Y \to B \oplus Y$, $x: A \oplus X \to A \oplus Y$, $x: B \oplus X \to B \oplus Y$ are also labeled transitions of the same automaton. This leads to the "independence square" associated to the structured transition $a \| x$, i.e.:

a) if two transitions can fire independently from the same source state, then they should be able to fire concurrently and doing so, reach the same target state;
b) if two independent transitions can fire, one immediately after the other, then they should be able to fire with interchanged order. ❑

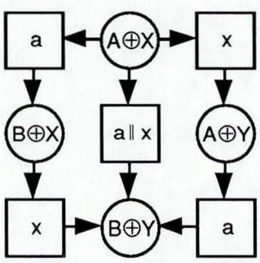

Fig. 4. Independence square

Definition 2.5 Nonsequential Automaton Morphism. A nonsequential automaton morphism $h: N_1 \to N_2$ where $N_1 = \langle V_1, T_1, \partial_{01}, \partial_{11}, \iota_1, L_1, lab_1 \rangle$ and $N_2 = \langle V_2, T_2, \partial_{02}, \partial_{12}, \iota_2, L_2, lab_2 \rangle$ is a triple $h = \langle h_V, h_T, h_L \rangle$ such that $h_V: V_1 \to V_2$, $h_T: T_1 \to T_2$, $h_L: L_1 \to L_2$ are $\mathcal{CMon}$-morphisms, $h_V \circ \partial_{k1} = \partial_{k2} \circ h_T$, $h_T \circ \iota_1 = \iota_2 \circ h_V$ and $h_L \circ lab_1 = lab_2 \circ h_T$. ❑

Nonsequential automata and their morphisms constitute the category $\mathcal{NAut}$.

Proposition 2.6 The category $\mathcal{NAut}$ is complete and cocomplete. Moreover products and coproducts are isomorphic.
Proof: See [9]. ❑

A categorical product (or coproduct) of two automata $N_1 = \langle V_1, T_1, \partial_{01}, \partial_{11}, \iota_1, L_1, lab_1 \rangle$, $N_2 = \langle V_2, T_2, \partial_{02}, \partial_{12}, \iota_2, L_2, lab_2 \rangle$ is as follows:

$$N_1 \times_{\mathcal{NAut}} N_2 = \langle V_1 \times_{\mathcal{CMon}} V_2, T_1 \times_{\mathcal{CMon}} T_2, \partial_{01} \times \partial_{02}, \partial_{11} \times \partial_{12}, \iota_1 \times \iota_2, L_1 \times_{\mathcal{CMon}} L_2, lab_1 \times lab_2 \rangle$$

where $\partial_{k1} \times \partial_{k2}$, $\iota_1 \times \iota_2$ and $lab_1 \times lab_2$ are uniquely induced by the product construction. Intuitively, the product in $\mathcal{NAut}$ is viewed as a composition of component automata.

Example 2.7 Consider the nonsequential automata $N_1 = \langle\{A, B, C\}^{\oplus}, \{a, b, \iota_A, \iota_B, \iota_C\}^{\parallel}, \partial_{0_1}, \partial_{1_1}, \iota_1, \{u\}^{\parallel}, lab_1\rangle$ and $N_2 = \langle\{X, Y\}^{\oplus}, \{x, \iota_X, \iota_Y\}^{\parallel}, \partial_{0_2}, \partial_{1_2}, \iota_2, \{v\}^{\parallel}, lab_2\rangle$ (free monoids) where source and target morphisms are determined by the local transitions $a: A \to B$, $b: B \to C$, $x: 2X \to Y$ and with labeling given by $a \mapsto u$, $b \mapsto u$, $x \mapsto v$. Then, $N_1 \times N_2 = \langle\{A, B, C, X, Y\}^{\oplus}, \{a, b, x, \iota_A, \iota_B, \iota_C, \iota_X, \iota_Y\}^{\parallel}, \partial_0, \partial_1, \iota, \{u, v\}^{\parallel}, lab\rangle$ with ∂_0, ∂_1, ι, lab uniquely induced by the product construction is represented in the Figure 5. ❑

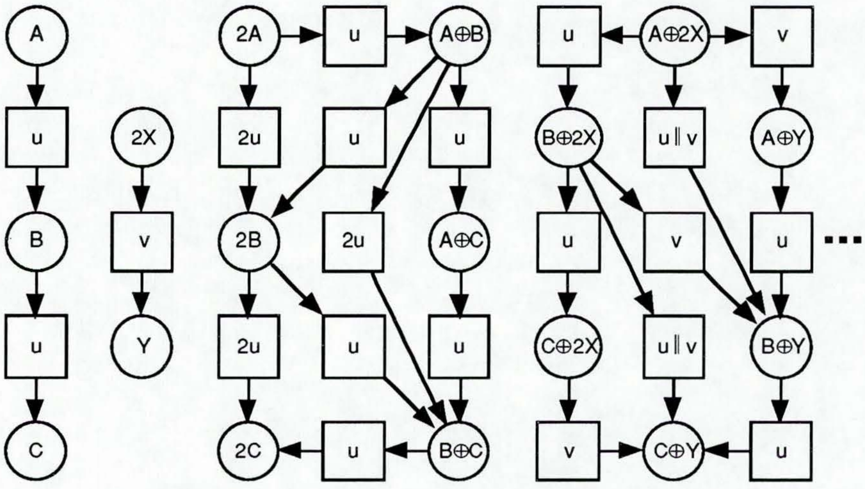

Fig. 5. Resulting nonsequential automaton of a product

3 Synchronization and Hiding

Synchronization and hiding of transitions are functorial operations defined using fibration and cofibration techniques. Both functors are induced by morphisms at the label level.

The synchronization operation erases from the product all those transitions which do not reflect some given table of synchronizations. The approach for synchronization is inspired by [8] and is as follows (see the Figure 6):

a) let N_1, N_2 be nonsequential automata with L_1, L_2 as the corresponding commutative monoids of labels;
b) let $\text{Table}(L_1, L_2)$ be a table of synchronizations determined by the pairs of labels to be synchronized and $\text{sync}: \text{Table}(L_1, L_2) \to L_1 \times L_2$ be the synchronization morphism which maps the table into the labels of a given automaton;
c) let $u: \mathcal{NAut} \to \mathcal{CMon}$ be the obvious forgetful functor taking each automaton into its commutative monoid of labels. The functor u is a fibration and the fibers $u^{-1}\text{Table}(L_1, L_2)$, $u^{-1}L_1 \times L_2$ are subcategories of $\mathcal{NAut}$;
d) the fibration u and the morphism sync induce a functor $sync: u^{-1}L_1 \times L_2 \to u^{-1}\text{Table}(L_1, L_2)$. The functor $sync$ applied to $N_1 \times N_2$ provides the automaton reflecting the desired synchronizations.

Traditionally, in concurrency theory, the concealment of transitions is achieved by resorting to labeling and using the special label τ (cf. [17]). Such hidden transitions cannot

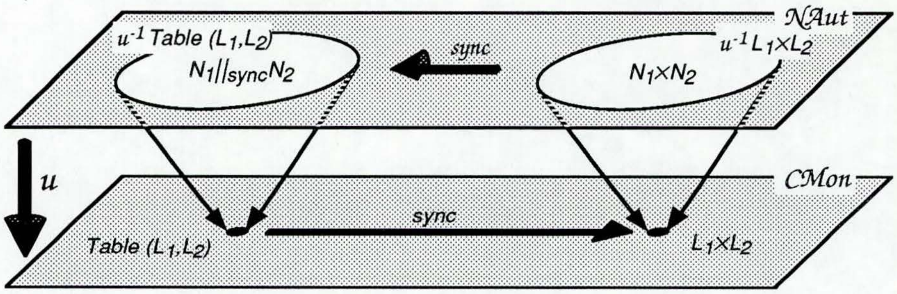

Fig. 6. Induced synchronization functor

be used for synchronization since they are *encapsulated*. The steps for hiding are the following:

a) let N be a nonsequential automaton with L_1 as the commutative monoid of labels;
b) let hide: $L_1 \to L_2$ be a morphism taking the transitions to be hidden into τ;
c) let u: $\mathcal{NAut} \to \mathcal{CMon}$ be the same forgetful functor used for synchronization purpose. The functor u is a cofibration (and therefore, a bifibration) and the fibers $u^{-1}L_1$, $u^{-1}L_2$ are subcategories of $\mathcal{NAut}$;
d) the cofibration u and the morphism hide induce a functor *hide*: $u^{-1}L_1 \to u^{-1}L_2$. The functor *hide* applied to N provides the automaton reflecting the desired encapsulation.

3.1 Synchronization

In what follows, we show a categorial way to construct tables of synchronizations for event calling and event sharing and the corresponding synchronization morphism.

Table of Synchronizations. The table of synchronizations for interaction is given by a colimit of a "twin peaks" or "M" diagram (i.e., a diagram with the shape $\bullet\leftarrow\bullet\rightarrow\bullet\leftarrow\bullet\rightarrow\bullet$). We say that a shares x if and only if a calls x and x calls a. In what follows, we denote by a | x a pair of synchronized transitions.

Definition 3.1 Table of Synchronizations. Let N_1, N_2 be nonsequential automata with L_1, L_2 as the corresponding commutative monoids of labels and let i be in {1, 2}:

a) let Channel(L_1, L_2) be the least commutative monoid determined by all pairs of transitions to be synchronized;
b) let L_i' be the least commutative submonoid of L_i containing all transitions of N_i which call a transition of the other automaton;
c) the morphisms $call_i$: $L_i' \to$ Channel(L_1, L_2) are such that, for a in L_i', if a calls x then $call_i(a)$ = a | x.

Let M(L_1, L_2) be the twin peaks diagram represented in the Figure 7 where inc_i: $L_i' \to L_i$ are the canonical inclusion morphisms. The table of synchronizations Table(L_1, L_2) is given by the colimit of M(L_1, L_2). ❑

From the definition above, we can infer that: (from c) $call_i$ are monomorphisms.

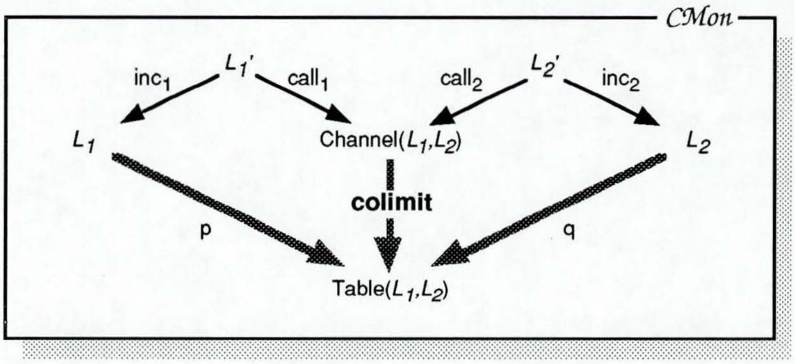

Fig. 7. Table of synchronizations

Example 3.2 Consider the free commutative monoids of labels $L_1 = \{a, b, c\}^{\parallel}$, $L_2 = \{x, y\}^{\parallel}$. Suppose that a calls x, b calls y and y calls b (i.e., b shares y). Then, Channel(L_1, L_2) = $\{a|x, b|y\}^{\parallel}$, $L_1' = \{a, b\}^{\parallel}$, $L_2' = \{y\}^{\parallel}$ and Table(L_1, L_2) = $\{c, x, a|x, b|y\}^{\parallel}$. ❑

Let M(L_1, L_2) be a twin peaks diagram whose colimit determines Table(L_1, L_2) and p: $L_1 \to$ Table(L_1, L_2), q: $L_2 \to$ Table(L_1, L_2). Then there are retractions for p and q denoted by p^R and q^R respectively as follows:

for every b in Table(L_1, L_2),

if there is a in L_1 such that p(a) = b then p^R(b) = a else p^R(b) = ✔;

if there is a in L_2 such that q(a) = b then q^R(b) = a else q^R(b) = ✔.

Definition 3.3 Synchronization Morphism. The synchronization morphism sync: Table(L_1, L_2) $\to L_1 \times L_2$ is uniquely induced by the product construction as illustrated in the Figure 8. ❑

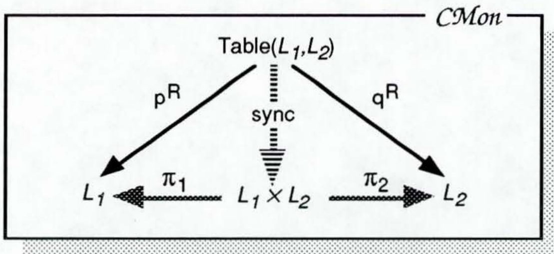

Fig. 8. Synchronization morphism

Synchronization Functor. First we show that the forgetful functor which takes each nonsequential automaton into its commutative monoids of labels is a fibration and then we introduce the synchronization functor.

Proposition 3.4 The forgetful functor *u: $\mathcal{NAut} \to \mathcal{CMon}$* that takes each nonsequential automaton onto its underlying commutative monoid of labels is a fibration. ❑

Proof: Let $\mathcal{RGr}(\mathcal{CMon})$ be the category of reflexive graphs internal to $\mathcal{CMon}$ and let id: $\mathcal{RGr}(\mathcal{CMon}) \to \mathcal{RGr}(\mathcal{CMon})$, emb: $\mathcal{CMon} \to \mathcal{RGr}(\mathcal{CMon})$ be functors. Then, $\mathcal{NAut}$ can be defined as the comma category $id \downarrow emb$. Let $f: L_1 \to L_2$ be a $\mathcal{CMon}$-morphism and $N_2 = \langle G_2, L_2, lab_2 \rangle$ be a nonsequential automaton where $G_2 = \langle V_2, T_2, \partial_{02}, \partial_{12}, \iota_2 \rangle$ is a $\mathcal{RGr}(\mathcal{CMon})$-object. Consider the $\mathcal{RGr}(\mathcal{CMon})$-pullback represented in the Figure 9. Define $N_1 = \langle G_1, L_1, lab_1 \rangle$ which is an automaton by construction. Then $u = \langle u_G, f \rangle$: $N_1 \to N_2$ is cartesian with respect to f and N_2. ❑

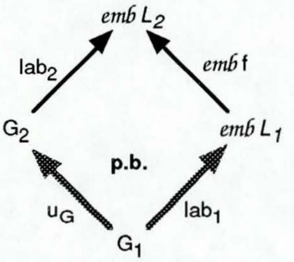

Fig. 9. Pullback

Definition 3.5 Functor sync. Consider the fibration u: $\mathcal{NAut} \to \mathcal{CMon}$, the nonsequential automata $N_1 = \langle V_1, T_1, \partial_{01}, \partial_{11}, \iota_1, L_1, lab_1 \rangle$, $N_2 = \langle V_2, T_2, \partial_{02}, \partial_{12}, \iota_2, L_2, lab_2 \rangle$ and the synchronization morphism sync: $\text{Table}(L_1, L_2) \to L_1 \times L_2$. The synchronization of N_1, N_2 represented by $N_1 \|_{sync} N_2$ is given by the functor $sync$: $u^{-1}(L_1 \times L_2) \to u^{-1}(\text{Table}(L_1, L_2))$ induced by u and sync applied to $N_1 \times N_2$, i.e.:

$$N_1 \|_{sync} N_2 \text{ is } sync(N_1 \times N_2).$$ ❑

Example 3.6 Consider the nonsequential automata Consumer and Producer (with free monoids) determined by the following labeled transitions:

Producer: prod: A → B, send: B → A
Consumer: rec: X → Y, cons: Y → X

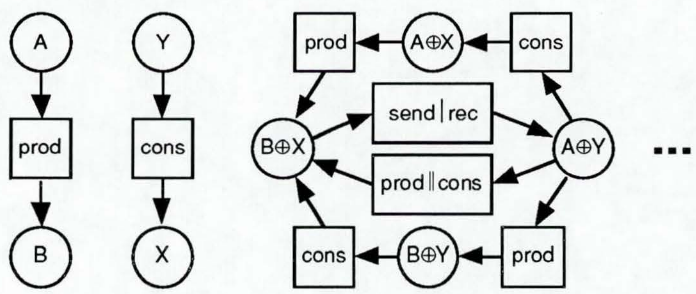

Fig. 10. Synchronized automaton

Suppose that we want a joint behavior sharing the transitions send and rec (a communication without buffer such as in CSP [6] or CCS [12]). Then, $\text{Channel}(L_1, L_2) = \{\text{send} \mid \text{rec}\}^{\|}$ and $\text{Table}(L_1, L_2) = \{\text{prod, cons, send} \mid \text{rec}\}^{\|}$. The resulting automaton is illustrated in the Figure 10. Note that the transitions send, rec are erased and send | rec is included. ❑

3.2 Hiding

For encapsulation purposes, we work with *hiding morphisms*. A hiding morphism is in fact an injective morphism except for those labels we want to hide (i.e., to relabel by τ). In what follows, remember that a monoid with only one element, denoted by e, is a zero object.

Definition 3.7 Hiding Morphism. Let L_1 be the commutative monoid of labels of the automata to be encapsulated, L be least commutative submonoid of L_1 containing all labels to be hidden and inc: $L \to L_1$ be the inclusion morphism. The hiding morphism hide: $L_1 \to L_2$ is determined by the pushout illustrated in the Figure 11 where the morphism ! is unique. ❑

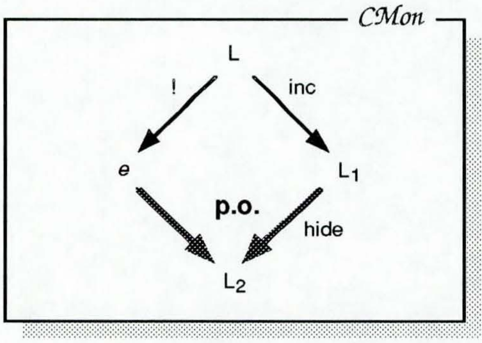

Fig. 11. Hiding morphism

Proposition 3.8 The forgetful functor u*:* $\mathcal{NAut} \to \mathcal{CMon}$ that maps each automaton onto its underlying commutative monoid of labels is a cofibration.
Proof: Let f: $L_1 \to L_2$ be a $\mathcal{CMon}$-morphism and $N_1 = \langle V_1, T_1, \partial_{0_1}, \partial_{1_1}, \iota_1, L_1, lab_1 \rangle$ be an automaton. Define $N_2 = \langle V_1, T_1, \partial_{0_1}, \partial_{1_1}, \iota_1, L_2, f \circ lab_1 \rangle$. Then $u = \langle id_{V_1}, id_{T_1}, f \rangle$: $N_1 \to N_2$ is cocartesian with respect to f and N_1. ❑

Definition 3.9 Functor hide. Consider the fibration u*:* $\mathcal{NAut} \to \mathcal{CMon}$, the nonsequential automata $N = \langle V, T, \partial_0, \partial_1, \iota, L_1, lab \rangle$ and the hiding morphism hide: $L_1 \to L_2$. The hiding of N satisfying hide denoted by N\hide is given by the functor *hide:* $u^{-1} L_1 \to u^{-1} L_2$ induced by u and hide applied to N, i.e.,

$$N\backslash hide = \mathit{hide}\, N$$ ❑

Example 3.10 Consider the resulting automata of the Example 3.6. Suppose that we want to hide the synchronized transition send | rec. Then, the hiding morphism is induced by send | rec $\mapsto \tau$ and the encapsulated automaton is as illustrated in the Figure 12. ❑

4 Refinement

A refinement mapping is defined as a special automaton morphism where the target object is closed under computations, i.e., the target (more concrete) automaton is enriched with all the conceivable sequential and nonsequential computations that can be split into permutations of original transitions, respecting source and target states. This transitive closure is easily performed in Category Theory:

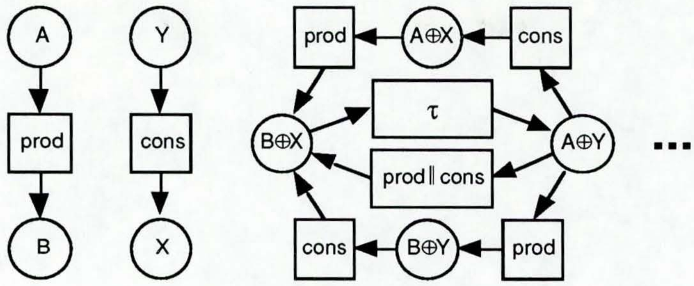

Fig.12. Encapsulated automaton

a) a reflexive graph plus a composition operation on transitions determines a category;
b) there exists a (obvious) functor forgetting the composition operation;
c) this functor has a left adjoint: a functor that freely generates a category from a reflexive graph;
d) the composition of both functors determines an endofunctor taking each reflexive graph onto its transitive closure;
e) the generalization of the above approach for nonsequential automata leads to the envisaged transitive closure.

Therefore, a refinement of an automaton N on top of an automaton M is a morphism φ: N $\to$ *tc*M, where *tc* is the transitive closure functor. Automata and refinement morphisms constitute a category (defined as a Kleisli category - see [2]) and thus, refinements compose. Then we show that refinement distributes over (system) composition and therefore, the resulting category of automata and refinements *satisfies the diagonal compositionality*.

In what follows, let $\mathcal{CMonCat}$ be the category of small strictly symmetric strict monoidal categories which is complete and cocomplete with products isomorphic to coproducts. Consider the functor $id_{\mathcal{CMonCat}}$: $\mathcal{CMonCat} \to \mathcal{CMonCat}$ and the comma category $id_{\mathcal{CMonCat}} \downarrow id_{\mathcal{CMonCat}}$ denoted by $\mathcal{CMC} \downarrow \mathcal{CMC}$. Note that the objects of $\mathcal{CMC} \downarrow \mathcal{CMC}$ are functors.

Definition 4.1 Functor u: $\mathcal{CMC} \downarrow \mathcal{CMC} \to \mathcal{NAut}$. The functor *u*: $\mathcal{CMC} \downarrow \mathcal{CMC} \to \mathcal{NAut}$ is such that for each $\mathcal{CMC} \downarrow \mathcal{CMC}$-object ℓ: $\mathcal{M} \to \mathcal{L}$ we have that:

a) for $\mathcal{M} = \langle\langle V, T, \partial_0, \partial_1, \iota, ;\rangle, \otimes, e\rangle$, $u\mathcal{M}$ is the $\mathcal{RGr}(\mathcal{CMon})$-object M = $\langle\langle V, \otimes, e\rangle, \langle T^a, \otimes^a, \iota_e\rangle, \partial_0^a, \partial_1^a, \iota\rangle$ where T^a is T subjected to the equational rule below and ∂_0^a, ∂_1^a, $\otimes^a$ are ∂_0, ∂_1, $\otimes$ restricted to T^a;

$$\frac{t: A \to B \in T^a \quad t': A' \to B' \in T^a \quad u: B \to C \in T^a \quad u': B' \to C' \in T^a}{(t;u) \otimes (t';u') = (t \otimes t');(u \otimes u')}$$

b) for $\mathcal{L} = \langle\langle V, T, \partial_0, \partial_1, \iota, ;\rangle, \otimes, e\rangle$, $u\mathcal{L}$ is the $\mathcal{CMon}$-object $L = \langle L, \otimes^a, \iota_e\rangle$ where L = T^a - { t | there is v in V such that $\iota(v)$ = t} and T^a, $\otimes^a$ are as defined above;
c) lab: M $\to$ L is the labeling morphism canonically induced by ℓ: $\mathcal{M} \to \mathcal{L}$. ❑

Besides forgetting about the composition operation, the functor *u*: $\mathcal{CMC} \downarrow \mathcal{CMC} \to \mathcal{NAut}$ has an additional requirement about concurrency:

$$(t;u) \parallel (t';u') = (t \parallel t');(u \parallel u')$$

That is, the parallel composition of two computations t;u and t';u' has the same effect as the computation whose steps are the parallel compositions t‖t' and u‖u'. As an illustration, let t: A → B and u: C → D be two computations. Then, for $t \| u$: A⊕C → B⊕D, we have that (in the following, we do not identify an endotransition by its label τ):

$$t \| u = (\iota_A;t) \| (u;\iota_D) = (\iota_A \| u);(t \| \iota_D) = u;t$$
$$t \| u = u \| t = (\iota_C;u) \| (t;\iota_B) = (\iota_C \| t);(u \| \iota_B) = t;u$$

Therefore, the concurrent execution of two transitions is equivalent to their execution in any order. As a consequence, any computation $t = t_1 \| t_2 \| ... \| t_n$ can be split as the sequential composition of its local transitions, i.e. (suppose t_i: $A_i \to B_i$):

$$t = t_1 \| t_2 \| ... \| t_n = (t_1 \| \iota_{A_1});(t_2 \| \iota_{A_2});...;(t_n \| \iota_{A_n}) = t_1;t_2;...;t_n$$

Definition 4.2 Functor f: 𝒩𝒜ut → 𝒞ℳ𝒞↓𝒞ℳ𝒞. The functor *f*: 𝒩𝒜*ut* → 𝒞ℳ𝒞↓𝒞ℳ𝒞 is such that:

a) for each 𝒩𝒜*ut*-object N = ⟨M, *L*, lab⟩ where M = ⟨*V*, *T*, ∂_0, ∂_1, ι⟩, *V* = ⟨V, ⊕, e⟩, *T* = ⟨T, ‖, τ⟩, *L* = ⟨L, ‖, τ⟩ we have that:

 a.1) *f*M is the *CMonCat*-object ℳ = ⟨⟨V, T^c, ∂_0^c, ∂_1^c, ι, ;⟩, ⟨⊕, ‖⟩, e⟩ where the composition is a partial operation and T^c, ∂_0^c, ∂_1^c are defined by the following rules of inference:

$$\frac{t: A \to B \in T}{t: A \to B \in T^c} \qquad \frac{t: A \to B \in T^c \quad u: B \to C \in T^c}{t;u: A \to C \in T^c}$$

$$\frac{t: A \to B \in T^c \quad u: C \to D \in T^c}{t \| u: A \oplus C \to B \oplus D \in T^c}$$

 subject to the following equational rules:

$$\frac{t: A \to B \in T^c}{\iota_A;t = t \text{ and } t;\iota_B = t}$$

$$\frac{t: A \to B \in T^c \quad u: B \to C \in T^c \quad v: C \to D \in T^c}{t;(u;v) = (t;u);v}$$

$$\frac{t \in T^c}{t \| \tau = t} \qquad \frac{t \in T^c \quad u \in T^c}{t \| u = u \| t}$$

$$\frac{}{\iota_A \| \iota_B = \iota_{A \oplus B}} \qquad \frac{t \in T^c \quad u \in T^c \quad v \in T^c}{t \| (u \| v) = (t \| u) \| v}$$

 a.2) *f*L is the *CMonCat*-object ⟨⟨{e}, L^c, !, !, ι, ;⟩, ‖, e⟩ where L^c is defined as above, ! is unique and ι is such that ι(e) = τ;

 a.3) the functor freely generated by N = ⟨M, *L*, lab⟩ is *f*lab: *f*M → *fL*;

b) for each 𝒩𝒜*ut*-morphism h = ⟨h_V, h_T, h_L⟩ where ⟨h_V, h_T⟩ is a ℛ𝒢*r*(𝒞ℳ*on*)-morphism and h_L is a *CMon*-morphism we have that:

 b.1) *f*⟨h_V, h_T⟩ is the *CMonCat*-morphism ⟨h_V, h_T^c⟩: $fM_1 \to fM_2$ where h_T^c is inductively defined as follows (suppose A, B in V and t , u in T):

$$h_T^c(t) = h_T(t) \qquad h_T^c(\iota_A) = \iota_{h_V(A)}$$
$$h_T^c(t \| u) = h_T^c(t) \| h_T^c(u) \qquad h_T^c(t;u) = h_T^c(t) ; h_T^c(u)$$

 b.2) fh_L is the *CMonCat*-morphism ⟨!, h_L^c⟩: $fL_1 \to fL_2$ where h_L^c is defined as above. ❑

Proposition 4.3 The functor f is left adjoint to u.

Proof: Consider η: $id_{\mathcal{NA}ut} \to u \circ f$ a natural transformation which is an embedding on transitions (and corresponding labels). Thus, for each $\mathcal{NA}ut$-object N = ⟨M, L, lab⟩, for each $\mathcal{CMC}{\downarrow}\mathcal{CMC}$-object $\mathcal{N} = \langle \mathcal{M}, \mathcal{L}, \ell \rangle$, for each $\mathcal{NA}ut$-morphism f: N $\to u\mathcal{N}$, there is only one $\mathcal{CMC}{\downarrow}\mathcal{CMC}$-morphism g: fN $\to \mathcal{N}$ such that f = ug $\circ\, \eta$N. In fact g is just like ff except that its target is $\mathcal{N}$ instead of $f \circ u\mathcal{N}$. By duality, ε: $f \circ u \to id_{\mathcal{CMC}\downarrow\mathcal{CMC}}$ is a natural transformation which takes each freely composed transition (label) ⟨t⟩;⟨u⟩ and ⟨t⟩‖⟨u⟩ onto the transition (label) ⟨t;u⟩ and ⟨t‖u⟩, respectively. Thus, ⟨f, u, η, ε⟩: $\mathcal{NA}ut \to \mathcal{CMC}{\downarrow}\mathcal{CMC}$ is an adjunction. ❑

Let ⟨f, u, η, ε⟩: $\mathcal{NA}ut \to \mathcal{CMC}{\downarrow}\mathcal{CMC}$ be the adjunction defined in the proposition above. Then, T = ⟨tc, η, μ⟩ is a monad on $\mathcal{NA}ut$, where $tc = u \circ f$: $\mathcal{NA}ut \to \mathcal{NA}ut$ is an endofunctor and $\mu = u\varepsilon f$: $tc^2 \to tc$ is a natural transformation where u: $u \to u$, f: $f \to f$ denote the identity natural transformations and $u\varepsilon f$ is the horizontal composition of natural transformations. A monad is useful to understand the computations of an automaton: for an automaton N, tcN reflects the computations of N, i.e., the transitive closure of N, η_N: N $\to$ tcN maps N into its computations and μ_N: tc^2N $\to$ tcN flattens computations of computations into computations.

Example 4.4 Consider the nonsequential automaton N_1 with free monoids on states, transitions and labels determined by the labeled transitions a: A $\to$ B and b: B $\to$ C. Its transitive closure is represented in the Figure 13 (the transactions added by the transitive closure are dashed). Note that transactions with "‖" are in fact classes of transactions. For instance, for a;2b: A⊕B $\to$ 2C we have that a;2b = (ι_B‖a);(b‖b) = (ι_B;b)‖(a;b) = b‖(a;b) = (b;ι_C)‖(ι_A;(a;b)) = (b‖ι_A);(ι_C‖(a;b)) = b;a;b = ... ❑

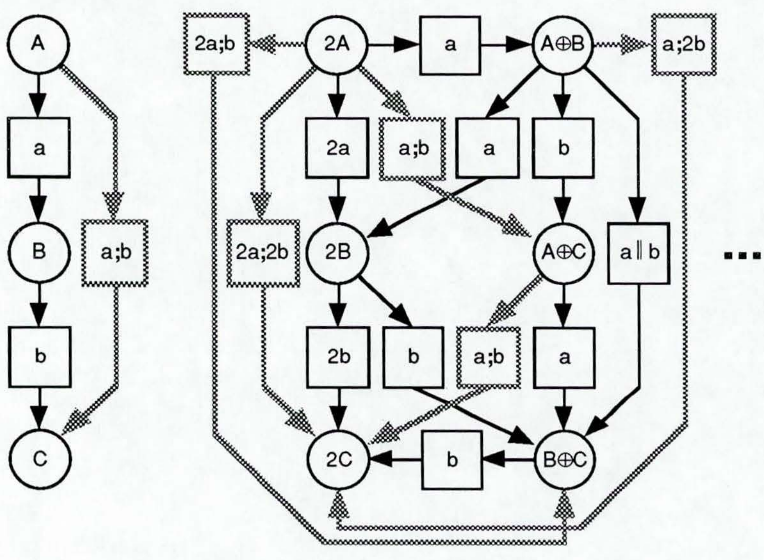

Fig. 13. Transitive closure of a nonsequential automaton

Definition 4.5 Category $\mathcal{R}ef\mathcal{NA}ut$. Let T = ⟨$tc$, η, μ⟩ be a monad on $\mathcal{NA}ut$ induced by the adjunction ⟨f, u, η, ε⟩: $\mathcal{NA}ut \to \mathcal{CMC}{\downarrow}\mathcal{CMC}$. The category of nonsequential automata and refinement morphisms is the Kleisli category determined by T, denoted by $\mathcal{R}ef\mathcal{NA}ut$.❑

Therefore, a refinement between two nonsequential automata N_1 and N_2, denoted by $\varphi: N_1 \Rightarrow N_2$, is a $\mathcal{NAut}$-morphism $\varphi: A_1 \to tc\,A_2$ and the composition of given refinement morphisms is the composition in $\mathcal{RefNAut}$.

Example 4.6 Consider the nonsequential automaton N_1 (previous example) and the automaton N_2 with free monoids on states, transitions and labels determined by the local labeled transitions $x: X \to Y$ and $y: Y \to X$. The refinement morphism $\varphi: N_1 \Rightarrow N_2$ is given by $A \mapsto 2X$, $B \mapsto 2Y$, $C \mapsto 2Y$, $a \mapsto x \| x$ and $b \mapsto 2y;x;y;2x$. ❑

In the next proposition, we prove that this construction also satisfies the horizontal compositionality: refinement of systems distributes through system composition.

Proposition 4.7 Let $\{\varphi_i: N_i \Rightarrow M_i\}_{i\in I}$ be a family of refinement, with I a set. Then $\times_{i\in I}\varphi_i: \times_{i\in I} N_i \Rightarrow \times_{i\in I} M_i$.
Proof: For simplicity, we abbreviate $\times_{i\in I}$ and $+_{i\in I}$ by $\times_i$ and $+_i$, respectively Consider the morphism $\times_i \varphi_i: \times_i N_i \to \times_i\, tcM_i$ uniquely induced by the product construction as illustrated in the Figure 14. Now, we have only to prove that $\times_i \varphi_i: \times_i N_i \to \times_i\, tcM_i$ is a $\mathcal{RefNAut}$-morphism. Since $tc = u \circ f$ and u is right adjoint we have that $\times_i \varphi_i: \times_i N_i \to u(\times_i fM_i)$. Moreover $\times_i fN_i'$ is isomorphic to $+_i fN_i'$. Thus, up to an isomorphism, $\times_i \varphi_i: \times_i N_i \to u(+_i fM_i)$. Since f is left adjoint (and so, preserves colimits) we have that $\times_i \varphi_i: \times_i N_i \to u \circ f(+_i M_i)$. Since $\times_i M_i$ is isomorphic to $+_i M_i$, then $\times_i \varphi_i: \times_i N_i \to tc(\times_i M_i)$ and thus, is a $\mathcal{RefNAut}$-morphism. ❑

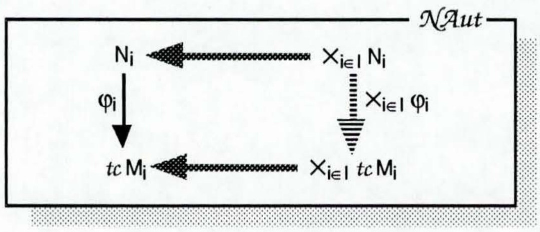

Fig. 14. Refinement morphism uniquely induced

5 Concluding Remarks

We introduced a new semantic domain for (discrete event) system based on structured labeled transition systems. Concepts and constructions like interaction, refinement and hiding, not (fully) explained in other semantic domains, have now a precise mathematical semantics.

Interaction of processes is categorically explained, by fibration techniques. Tables for interaction are categorically defined. The hiding of events is also dealt with, by cofibration techniques, introducing the essential ingredient of internal non-determinism. Refinement is explained through Kleisli categories ensuring the envisaged levels of diagonal (vertical and horizontal) compositionality.

With respect to further work, it should be clear that this may be the starting point of a rather fruitful line of research on the semantics of discrete event systems around transition systems and graph based models.

Acknowledgments

This work was partially supported by: UFRGS - Universidade Federal do Rio Grande do Sul and CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico in Brazil; CEC under ESPRIT-III BRA WG 6071 IS-CORE, HCM Scientific Network MEDICIS, JNICT (PBIC/C/TIT/1227/92) in Portugal.

References

1. M. A. Arbib, E. G. Manes, *Arrows, Structures and Functors - The Categorial Imperative*, Academic Press, 1975.
2. A. Asperti, G. Longo, *Categories, Types and Structures - An Introduction to the Working Computer Science*, Foundations of Computing (M. Garey, A. Meyer Eds.), MIT Press, 1991.
3. M. A. Bednarczyk, *Categories of Asynchronous Systems*, Ph.D. thesis, technical report 1/88, University of Sussex, 1988.
4. H. D. Ehrich, A. Sernadas, *Algebraic Implementation of Objects over Objects*, Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness (J. de Bakker, W. -P. de Roever, G. Rozenberg Eds.), pp. 239-266, Springer-Verlag, 1990.
5. R. Gorrieri, *Refinement, Atomicity and Transactions for Process Description Language*, Ph.D. thesis, Università di Pisa, 1990.
6. C. A. R. Hoare, *Communicating Sequential Processes*, Prentice Hall, 1985.
7. A. Mazurkiewicz, *Basic Notion of Trace Theory*, REX 88: Linear Time, Branching Time and Partial Orders in Logic and Models for Concurrency (J. W. de Bakker, W. -P. de Roever, G. Rozenberg, Eds.), pp. 285-363, LNCS 354, Springer-Verlag, 1988.
8. P. B. Menezes, J. F. Costa, *Synchronization in Petri Nets*, preprint IST/DM/2-94, IST, Lisbon, 1993. Revised version accepted for publication in Fundamenta Informaticae.
9. P. B. Menezes, J. F. Costa, *Compositional Refinement of Concurrent Systems*, preprint IST/DM/26-94, IST, Lisbon, 1994. Revised version accepted for publication in the Journal of the Brazilian Computer Society - Special Issue on Parallel Computation.
10. P. B. Menezes, J. F. Costa, *Object Refinement*, preprint IST/DM/24-94, IST, Lisbon, 1994.
11. J. Meseguer, U. Montanari, *Petri Nets are Monoids*, Information and Computation 88, pp. 105-155, Academic Press, 1990.
12. R. Milner, *Communication and Concurrency*, Prentice Hall, 1989.
13. C. Rattray, *The Shape of Complex Systems*, EUROCAST 93: Computer Aided Systems Theory (F. Pichler, R. M. Díaz, Eds.), pp. 72-82, LNCS 763, Springer-Verlag, 1994.
14. W. Reisig, *Petri Nets: An Introduction*, EATCS Monographs on Theoretical Computer Science 4, Springer-Verlag, 1985.
15. V. Sassone, M. Nielsen, G. Winskel, *A Classification of Models for Concurrency*, CONCUR 93: 4th International Conference of Concurrency (E. Best, Ed.), pp. 82-96, LNCS 715, Springer-Verlag, 1993.
16. M. E. Szabo, *Algebra of Proofs*, Studies in Logic and the Foundations of Mathematics, vol. 88, North-Holland, 1978.
17. G. Winskel, *Petri Nets, Algebras, Morphisms and Compositionality*, Information and Computation 72, pp. 197-238, Academic Press, 1987.