UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Systems for System Implementation

por

Paulo Blauth Menezes and Félix Costa

RP 262 Maio/1996

UFRGS-II-CPGCC
Caixa Postal 15064 - CEP 91501-970
Porto Alegre RS BRASIL
Telefone: (051)316-6155
Fax: (051) 336-5576
Email: pgcc@inf.ufrgs

Systems for System Implementation *

P. Blauth Menezes† and J. Félix Costa††

† Departamento de Matemática, Instituto Superior Técnico, Portugal - blauth@raf.ist.utl.pt

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

Abstract. Inspired by Meseguer and Montanari's "Petri Nets are Monoids", we propose that a refinement of a Petri net is a special kind of net morphism were the target object is enriched with all conceivable sequential and concurrent computations. Then it is proven that while refinement of nets satisfies the vertical compositionality requirement (i.e., refinements compose), it lacks the horizontal compositionality requirement (i.e., refinement does not distribute over parallel composition). To achieve both requirements, a new categorial semantic domain based on labeled transition systems with full concurrency, called nonsequential automata, is constructed. Again, a class of morphisms stands for refinement and, in this framework, the diagonal compositionality requirement (i.e., both vertical and horizontal) is achieved. Adjunctions between both models are provided extending the approach of Winskel and Nielsen. The steps of abstraction involved in moving between models show that nonsequential automata are more concrete than Petri nets.

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], [12] 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 [13]) lack 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 composition, and the approach in [12] 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 [2] and [7]. The resulting category is bicomplete with products isomorphic to coproducts. The categorial product (or coproduct) stand for the parallel composition. In [11] we introduce (functorial) operations of synchronization and encapsulation for nonsequential automata, where the synchronization restricts a parallel composition according to some given interaction specification and the encapsulation extracts a view of an automaton through hiding of transitions introducing an internal nondeterminism.

A refinement morphism maps transitions into transactions reflecting an implementation of an automaton 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 is defined using Kleisli categories. Therefore, refinements compose, i.e., the vertical compositionality requirement is achieved. Moreover we find a general theory for refinement which also satisfies the horizontal compositionality requirement. i.e., for refinements φ: $A_1 \rightarrow tcB_1$, ψ: $A_2 \rightarrow tcB_2$, we have that $\varphi A_1 \times \psi A_2 = \varphi \times \psi (A_1 \times A_2)$ where $\varphi A_1 \times \psi A_2$ and $A_1 \times A_2$ are the parallel composition and the refinement $\varphi \times \psi$ is (uniquely) induced by φ and ψ. While the vertical compositionality is easily achieved in several models, they lack horizontal compositionality (in this paper, we show that Petri nets lacks horizontal compositionality).

Adjunctions between categories of Petri nets and nonsequential automata are provided, extending the approach in [15] where a scene for a formal classification of models for concurrency is set. From the steps of abstractions that are involved in moving between models, we can infer that nonsequential automata are more concrete then Petri nets. Moreover, categories of Petri nets are isomorphic the subcategories of nonsequential automata. For our knowledge, the proposed model of nonsequential automata is the least concrete model which satisfies both vertical and horizontal compositionality requirements with respect to implementation.

The categories of Petri nets presented in this paper are extended with labeling on transitions. Also, for simplicity, we do not deal with initial states (or markings). If the categories in this framework are extended with initial states as in [8], all result are preserved, including the bicompleteness of the categories of Petri nets and nonsequential automata. In what follows, the proofs omitted are in [10].

* This work was partially supported by: UFRGS and CNPq in Brazil; CEC under ESPRIT-III BRA WG 6071 IS-CORE, HCM Scientific Network MEDICIS and JNICT (PBIC/C/TIT/1227/92) in Portugal.

2 Labeled Petri Nets

We introduce Petri nets viewed as graphs extended with labeling on transitions. As proposed in [12], to define a Petri net as a graph, we consider the states as a commutative monoid. For simplicity and comparing with the corresponding categories in [12], we drop the requirement of monoids being free. In what follows, suppose that k is in {0, 1}, *CMon* is the category of commutative monoids which has products isomorphic to coproducts and u_{set}: *CMon* → *Set* is a functor which forget about the monoidal structure.

Definition 2.1 Labeled Petri Net. A (labeled) Petri net is $N = \langle V, T, \partial_0, \partial_1, L, lab\rangle$ such that V is a commutative monoid of states, T is a set of transitions, $\partial_0, \partial_1: T \to u_{set}V$ are total functions called source and target, respectively, L is a set of labels and $lab: T \to L$ is a total function called labeling. ❑

Definition 2.2 Labeled Petri Net morphism. A (labeled) Petri net morphism is a triple $h = \langle h_V, h_T, h_L\rangle: \langle V_1, T_1, \partial_{01}, \partial_{11}, L_1, lab_1\rangle \to \langle V_2, T_2, \partial_{02}, \partial_{12}, L_2, lab_2\rangle$ such that $h_V: V_1 \to V_2$ is a *CMon*-morphism, $h_T: T_1 \to T_2$ is a total function such that $u_{set}h_V \circ \partial_{k1} = \partial_{k2} \circ h_T$ and $h_L: L_1 \to L_2$ is a total function such that $h_L \circ lab_1 = lab_2 \circ h_T$. ❑

A transition t such that $\partial_0(t) = A$, $\partial_1(t) = B$ and labeled by a is denoted by $a[t]: A \to B$ or just by $a: A \to B$. Labeled Petri nets and its morphisms constitute a category denoted by *LPetri* which is bicomplete (see [10]). The coproduct and product constructions represent the asynchronous and synchronous composition of nets, respectively.

3 Nonsequential Automata

A nonsequential automaton is a reflexive graph (a graph with an identity arc 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 identity arcs interpreted as *idle* transitions. A structured transition specify a concurrency relation between component transitions. Comparing with asynchronous transition systems (first introduced in [2]), 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.

Definition 3.1 Nonsequential Automaton. A nonsequential automaton $A = \langle V, T, \partial_0, \partial_1, \iota, L, lab\rangle$ is such that $T = \langle T, \|, \tau\rangle$, $V = \langle V, \oplus, e\rangle$, $L = \langle L, \|, \tau\rangle$ are *CMon*-objects of transitions, states and labels respectively, $\partial_0, \partial_1: T \to V$ are *CMon*-morphisms called source and target respectively, $\iota: V \to T$ is a *CMon*-morphism called identity such that $\partial_k \circ \iota = id_V$ and $lab: T \to L$ is a *CMon*-morphism such that $lab(t) = \tau$ whenever there is v in V where $\iota(v) = t$. ❑

For a state A, $\iota_A: A \to A$ is also denoted by $A: A \to A$. Note that every identity transitions is labeled by τ. Since a state is an element of a monoid, it may be denoted as a formal sum, as in Petri nets. The denotation of a transition is analogous. We also refer to a (structured) transition as the *parallel composition* of component transitions. A transition $x \| \tau: X \oplus A \to Y \oplus A$ where $t: X \to Y$ and $A: A \to A$ 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 identity transitions. States and labeled transitions are graphically represented as circles and boxes, respectively.

Example 3.2 Let $\langle \{A, B, X, Y\}^{\oplus}, \{t_1, t_2, t_3, A, B, C, X, Y\}^{\otimes}, \partial_0, \partial_1, \iota, \{x, y\}^{\otimes}, lab\rangle$ be a nonsequential automaton with ∂_0, ∂_1 determined by the local arcs $t_1: 2A \to B$, $t_2: X \to Y$, $t_3: Y \to X$ and lab determined by $t_1 \mapsto x$, $t_2 \mapsto x$, $t_3 \mapsto y$. The distributed and infinite schema in Figure 1 (left) represents the automaton. Since in this framework we do not deal with initial states, the graphical representation makes explicit all possible states that can be reached by all possible independent combination of component transitions. For instance, if we consider the initial state A⊕2X, only the corresponding part of the schema of the automata in the figure has to be considered. In Figure 1 (right), we illustrate a labeled Petri net which simulates the behavior of the automaton. Comparing both schema, we realize that, while the concurrence and possible reachable markings are implicit in a net, they are explicit in an automaton. ❑

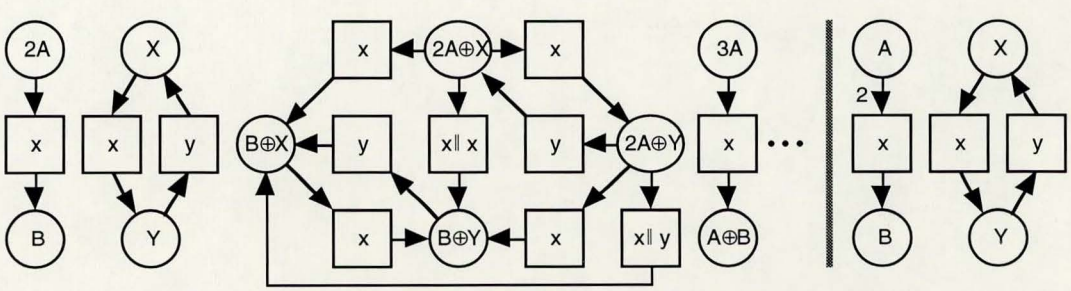

Figure 1 A nonsequential automaton (left) and the corresponding labeled Petri net (right)

Definition 3.3 Nonsequential Automaton Morphism. A nonsequential automaton morphism is a triple $h = \langle h_V, h_T, h_L\rangle$: $\langle V_1, T_1, \partial_{01}, \partial_{11}, \iota_1, L_1, lab_1\rangle \to \langle V_2, T_2, \partial_{02}, \partial_{12}, \iota_2, L_2, lab_2\rangle$ where h_V: $V_1 \to V_2$, h_T: $T_1 \to T_2$ and h_L: $L_1 \to L_2$ are *CMon*-morphisms such that $h_V \circ \partial_{k1} = \partial_{k2} \circ h_T$, $h_T \circ \iota_1 = \iota_2 \circ h_V$ and $h_L \circ lab_1 = lab_2 \circ h_T$. ❑

Nonsequential automata and its morphisms constitute a category denoted by $\mathcal{NAut}$ which is bicomplete with products isomorphic to coproducts (see [10]). The product construction stands for the parallel composition.

4 Adjunctions Between Petri Nets and Nonsequential Automata

The relationship between nonsequential automata and Petri nets is done through adjunctions as illustrated in Figure 2. To compare the expressiveness between nonsequential automata and Petri nets we introduce a subcategory of automata, denoted by $\mathcal{NAut}^s$, where the (non-identity) transitions of an automaton are elements of a free commutative monoid and the morphisms on transitions are induced by total functions. Since $\mathcal{NAut}^s$ is isomorphic to *LPetri*, $\mathcal{NAut}^s$ is bicomplete and *LPetri* is a subcategory of $\mathcal{NAut}$. The following notations are used: a set S is the set of generator of the free commutative monoid $S^\oplus$; for a commutative monoid $M = \langle M, \otimes, e\rangle$, M is the corresponding set.

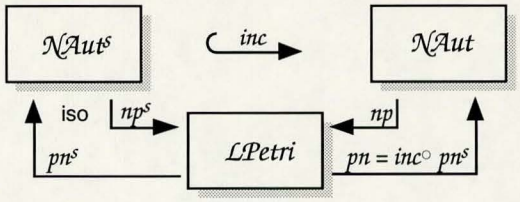

Figure 2 Adjunctions between the category of Petri nets and the categories of nonsequential automata

Definition 4.1 Nonsequential s-Automaton. The category of nonsequential s-automata, denoted by $\mathcal{NAut}^s$, is a subcategory of $\mathcal{NAut}$ where (consider the Figure 3): (a) a $\mathcal{NAut}^s$-object $A = \langle V, T^+, \partial_0^+, \partial_1^+, \iota, L^\oplus, lab^+\rangle$ is a $\mathcal{NAut}$-object such that $V = \langle V, \oplus, e\rangle$, T^+ and ι: $V \to T^+$ are given by the coproduct $T^\oplus + V$, ∂_k^+: $T^+ \to V$ is uniquely induced by the coproduct construction in *CMon* and lab^+: $T^+ \to L^\oplus$ is uniquely induced by a total function lab: $T \to L$ and by the product construction; (b) a $\mathcal{NAut}^s$-morphism $h = \langle h_V, h_{T+}, h_{L\oplus}\rangle$: $A_1 \to A_2$ is a $\mathcal{NAut}$-morphism such that h_{T+}: $T_1^+ \to T_2^+$ is uniquely induced by a total function h_T: $T_1 \to T_2$ and by the coproduct construction in *CMon* and $h_{L\oplus}$: $L_1^\oplus \to L_2^\oplus$ is induced by a total function h_L: $L_1 \to L_2$. ❑

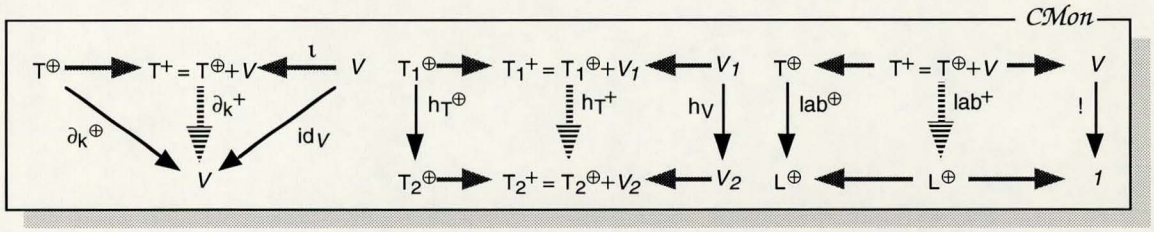

Figure 3 Diagrams for the category of nonsequential s-automaton (in *CMon*, products and coproducts are ismorphic)

Definition 4.2 Functors pn^s, np^s. (a) The functor pn^s: *LPetri* $\to \mathcal{NAut}^s$ is such that for each *LPetri*-object $N = \langle V, T, \partial_0, \partial_1, L, lab\rangle$, $pn^s N = \langle V, T^+, \partial_0^+, \partial_1^+, \iota, L^\oplus, lab^+\rangle$ and for each *LPetri*-morphism $h = \langle h_V, h_T, h_L\rangle$, $pn^s h = \langle h_V, h_{T+}, h_{L\oplus}\rangle$. (b) The functor np^s: $\mathcal{NAut}^s \to$ *LPetri* is such that for each $\mathcal{NAut}^s$-object $A = \langle V, T^+, \partial_0^+, \partial_1^+, \iota, L^\oplus, lab^+\rangle$, $np^s A = \langle V, T, \partial_0, \partial_1, L, lab\rangle$ where T, L are the sets of generators of the corresponding free monoids and for each $\mathcal{NAut}^s$-morphism $h = \langle h_V, h_{T+}, h_{L\oplus}\rangle$, $np^s h = \langle h_V, h_T, h_L\rangle$. ❑

Basically, the functor pn^s freely generates an automata from some given net and np^s forgets about the monoidal structure on transitions and erases the identity transitions and the transitions added by the generation of the free monoid. The Figure 1 illustrates both functors. The proof of the following proposition is straightforward.

Proposition 4.3 The categories *LPetri* and $\mathcal{NAut}^s$ are isomorphic. ❑

Definition 4.4 Functors pn, np. (a) Consider the functor pn^s: *LPetri* $\to \mathcal{NAut}^s$ defined above. Let *inc*: $\mathcal{NAut}^s \to \mathcal{NAut}$ be the inclusion functor which defines $\mathcal{NAut}^s$ as a subcategory of $\mathcal{NAut}$. Then, $pn = inc \circ pn^s$: *LPetri* $\to \mathcal{NAut}$. (b) The functor np: $\mathcal{NAut} \to$ *LPetri* is such that for each $\mathcal{NAut}$-object $A = \langle V, T, \partial_0, \partial_1, \iota, L, lab\rangle$ with $T = \langle T, \|, \tau\rangle$ and $L = \langle L, \|, \tau\rangle$, $npA = \langle V, T, \partial_0', \partial_1', L, lab'\rangle$ where lab', ∂_0', ∂_1' are canonically induced by lab, ∂_0, ∂_1 and for each $\mathcal{NAut}$-morphism $h = \langle h_V, h_T, h_L\rangle$, $pnh = \langle h_V, h_T', h_L'\rangle$ where h_T': $T_1 \to T_2$ and h_L': $L_1 \to L_2$ are canonically induced by h_T and h_L, respectively. ❑

Proposition 4.5 The functor pn: *LPetri* $\to \mathcal{NAut}$ is left adjoint to np: $\mathcal{NAut} \to$ *LPetri*.

5 Refinement

First, we introduce the refinement for automata. Then, using the adjunctions between automata and nets, the refinement for Petri nets is straightforward. Comparing with the implementation as in [12], the main differences are that nets and automata have labels on transitions and the transitive closure functor is defined over internal categories (see [3, 6]) instead of monoidal categories. With internal categories we may easily substitute the monoidal structure by any other structure (such as groups) provided that some properties about limits and colimits are preserved.

The category of categories internal to $\mathcal{CM}on$ is denoted by $Cat(\mathcal{CM}on)$. We introduce the category $\mathcal{L}Cat(\mathcal{CM}on)$ which can be viewed as a generalization of labeling on $Cat(\mathcal{CM}on)$. There is a forgetful functor from $\mathcal{L}Cat(\mathcal{CM}on)$ into $\mathcal{NA}ut$. This functor has a left adjoint which freely generates a nonsequential automaton into a labeled internal category. The composition of both functors from $\mathcal{NA}ut$ into $\mathcal{L}Cat(\mathcal{CM}on)$ leads to an endofunctor, called transitive closure. The composition of refinements of nonsequential automata is defined using Kleisli categories. In fact, the adjunction above induces a monad which defines a Kleisli category.

An important result is that, while the vertical compositionality is achieved by all related categories, the horizontal compositionality "starts" at $\mathcal{NA}ut$. This means that $\mathcal{NA}ut$ is the least concrete (or more abstract) level which satisfies the diagonal compositionality. In what follows, a $\mathcal{NA}ut$-object $A = \langle V, T, \partial_0, \partial_1, \iota, L, \mathsf{lab}\rangle$ is also denoted by $\langle G, L', \mathsf{lab}\rangle$ where $G = \langle V, T, \partial_0, \partial_1, \iota\rangle$ is a $\mathcal{RG}raph(\mathcal{CM}on)$-object, i.e., a reflexive graph internal to $\mathcal{CM}on$.

5.1 Vertical Compositionality

Definition 5.1 Category $\mathcal{L}Cat(\mathcal{CM}on)$. Consider the category $Cat(\mathcal{CM}on)$. The category $\mathcal{L}Cat(\mathcal{CM}on)$ is the comma category $id_{Cat(\mathcal{CM}on)}\downarrow id_{Cat(\mathcal{CM}on)}$ where $id_{Cat(\mathcal{CM}on)}$ is the identity functor in $Cat(\mathcal{CM}on)$. ❑

Therefore, a $\mathcal{L}Cat(\mathcal{CM}on)$-object is triple $\mathcal{N} = \langle \mathcal{G}, \mathcal{L}, \mathcal{lab}\rangle$ where $\mathcal{G}$, $\mathcal{L}$ are $Cat(\mathcal{CM}on)$-objects and $\mathcal{lab}$ is a $Cat(\mathcal{CM}on)$-morphism.

Proposition 5.2 The category $\mathcal{L}Cat(\mathcal{CM}on)$ has all (small) products and coproducts. Moreover, products and coproducts are isomorphic.

Definition 5.3 Functor cn. Let $\mathcal{N} = \langle \mathcal{G}, \mathcal{L}, \mathcal{lab}\rangle$ be a $\mathcal{L}Cat(\mathcal{CM}on)$-object and $\mathcal{h} = \langle \mathcal{h}_{\mathcal{G}}, \mathcal{h}_{\mathcal{L}}\rangle: \mathcal{N}_1 \to \mathcal{N}_2$ be a $\mathcal{L}Cat(\mathcal{CM}on)$-morphism. The functor $cn: \mathcal{L}Cat(\mathcal{CM}on) \to \mathcal{NA}ut$ is such that:

a) the $Cat(\mathcal{CM}on)$-object $\mathcal{G} = \langle V, T, \partial_0, \partial_1, \iota, ;\rangle$ is taken into the $\mathcal{RG}raph(\mathcal{CM}on)$-object $G = \langle V, T', \partial_0', \partial_1', \iota'\rangle$, where T' is T subject to the equational rule below and $\partial_0', \partial_1', \iota'$ are induced by $\partial_0, \partial_1, \iota$ considering the monoid T'; the $Cat(\mathcal{CM}on)$-object $\mathcal{L} = \langle V, L, \partial_0, \partial_1, \iota, ;\rangle$ is taken into the $\mathcal{CM}on$-object L', where L' is L subject to the same equational rule; the $\mathcal{L}Cat(\mathcal{CM}on)$-object $\mathcal{N} = \langle \mathcal{G}, \mathcal{L}, \mathcal{lab}\rangle$ is taken into the $\mathcal{NA}ut$-object $N = \langle G, L', \mathsf{lab}\rangle$ where lab is the $\mathcal{RG}raph(\mathcal{CM}on)$-morphism canonically induced by the $Cat(\mathcal{CM}on)$-morphism $\mathcal{lab}$;

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

b) the $\mathcal{L}Cat(\mathcal{CM}on)$-morphism $\mathcal{h} = \langle \mathcal{h}_{\mathcal{G}}, \mathcal{h}_{\mathcal{L}}\rangle: \mathcal{N}_1 \to \mathcal{N}_2$ with $\mathcal{h}_{\mathcal{G}} = \langle h_{NV}, h_{NT}\rangle$, $\mathcal{h}_{\mathcal{L}} = \langle h_{LV}, h_{LT}\rangle$ is taken into the $\mathcal{NA}ut$-morphism $h = \langle h_{NV}, h_{NT'}, h_{LT'}\rangle: N_1 \to N_2$ where $h_{NT'}$ and $h_{LT'}$ are the monoid morphisms induced by h_{NT} and h_{LT}, respectively. ❑

The functor cn has a requirement about concurrency which is $(t;u)\|(t';u') = (t\|t');(u\|u')$. That is, the computation determined by two independent composed transitions $t;u$ and $t';u'$ is equivalent to the computation whose steps are the independent transitions $t\|t'$ and $u\|u'$. For further details on this equation see [12].

Definition 5.4 Functor nc. Let $A = \langle G, L, \mathsf{lab}\rangle$ be a $\mathcal{NA}ut$-object and $h = \langle h_G, h_L\rangle: A_1 \to A_2$ be a $\mathcal{NA}ut$-morphism. The functor $nc: \mathcal{NA}ut \to \mathcal{L}Cat(\mathcal{CM}on)$ is such that:

a) the $\mathcal{RG}raph(\mathcal{CM}on)$-object $G = \langle V, T, \partial_0, \partial_1, \iota\rangle$ with $V = \langle V, \oplus, e\rangle$, $T = \langle T, \|, \tau\rangle$ is taken into the $Cat(\mathcal{CM}on)$-object $\mathcal{G} = \langle V, T^c, \partial_0^c, \partial_1^c, \iota, ;\rangle$ with $T^c = \langle T^c, \otimes, \tau\rangle$, $\partial_0^c, \partial_1^c, \_;\_: T^c \times T^c \to T^c$ inductively defined as follows:

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

subject to the following equational rules:

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

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

the *CMon*-object L is taken into the *Cat(CMon)*-object $\mathcal{L} = \langle 1, L^c, !, !, !, ; \rangle$ as above; the $\mathcal{NA}ut$-object $A = \langle G, L, \text{lab} \rangle$ is taken into the *LCat(CMon)*-object $\mathcal{A} = \langle \mathcal{G}, \mathcal{L}, lab \rangle$ where *lab* is the morphism induced by lab;

d) the $\mathcal{NA}ut$-morphism $h = \langle h_V, h_T, h_L \rangle: A_1 \to A_2$ is taken into the *Cat(CMon)*-morphism $\mathcal{h} = \langle h_{\mathcal{G}}, h_{\mathcal{L}} \rangle: \mathcal{A}_1 \to \mathcal{A}_2$ where $h_{\mathcal{G}} = \langle h_V, h_{Tc} \rangle$, $h_{\mathcal{L}} = \langle !, h_{Lc} \rangle$ and h_{Tc}, h_{Lc} are the monoid morphisms generated by the monoid morphisms h_T and h_{T_L}, respectively. ❑

Proposition 5.5 The functor $nc: \mathcal{NA}ut \to \mathcal{LC}at(CMon)$ is left adjoint to $cn: \mathcal{LC}at(CMon) \to \mathcal{NA}ut$.

Definition 5.6 Transitive Closure Functor, Refinement Morphism: The transitive closure functor is $tc = cn \circ nc: \mathcal{NA}ut \to \mathcal{NA}ut$. A refinement morphism φ from A into the computations of B is a $\mathcal{NA}ut$-morphism $\varphi: A \to tcB$. ❑

Example 5.7 Consider the nonsequential automaton with free monoids on states and transitions, determined by the transitions $a: A \to B$ and $b: B \to C$. Then, for instance, $a;2b: A \oplus B \to B \oplus C$ is a transition in the transitive closure. Note that, a;2b represents a class of transitions. In fact, from the equations we can infer that $a;2b = a;(b \| b) = (\tau[B] \| a);(b \| b) = (\tau[B];b) \| (a;b) = b \| (a;b) = (b;\tau[C]) \| (\tau[A];(a;b)) = (b \| \tau[A]);(\tau[C] \| (a;b)) = b;a;b = \ldots$ ❑

Let $\langle nc, cn, \eta, \varepsilon \rangle$ be the adjunction from $\mathcal{NA}ut$ into $\mathcal{LC}at(CMon)$ as above. Then, $T = \langle tc, \eta, \mu \rangle$ is a monad on $\mathcal{NA}ut$ such that $\mu = cn\varepsilon\, nc: tc^2 \to tc$ where $cn: cn \to cn$ and $nc: nc \to nc$ are the identity natural transformations and $cn\varepsilon\, nc$ is the horizontal composition of natural transformations. For some given automaton A, tcA is A enriched with its computations, $\eta_A: A \to tcA$ includes A into its computations and $\mu_A: tc^2A \to tcA$ flattens computations of computations of A into the computations of A. Each monad defines a Kleisli category which provides the right setting to describe the composition of refinement and thus, the *vertical compositionality is achieved.*

Definition 5.8 Category of Nonsequential Automata and Refinements. Consider the monad $T = \langle tc, \eta, \mu \rangle$ on $\mathcal{NA}ut$ induced by the adjunction $\langle nc, cn, \eta, \varepsilon \rangle: \mathcal{NA}ut \to \mathcal{LC}at(CMon)$. The category of nonsequential automata and refinement morphisms, denoted by $\mathcal{RefNA}ut$, is the Kleisli category determined by the monad T. ❑

Therefore, for the refinements $\varphi: A_1 \to tcA_2$, $\psi: A_2 \to tcA_3$ the composition $\psi \circ \varphi: A_1 \to tcA_3$ in $\mathcal{RefNA}ut$ is given by the commutative diagram illustrated in the Figure 5.

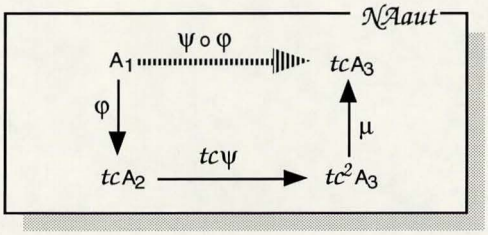

Figure 4 Composition of refinement morphisms of nonsequential automata

The definition of the category of Petri nets and refinements is analogous. The adjunction considered is obtained through the composition of adjunctions.

Definition 5.9 Category of Petri Nets and Refinements. Let $\langle pc = nc \circ pn,\ cp = np \circ cn \rangle: \mathcal{LP}etri \to \mathcal{LC}at(CMon)$ be the adjunction determined by the composition of the adjunctions $\langle nc, cn \rangle: \mathcal{NA}ut \to \mathcal{LC}at(CMon)$, $\langle pn, np \rangle: \mathcal{LP}etri \to \mathcal{NA}ut$. The category of labeled Petri nets and refinement morphisms, denoted by $\mathcal{RefP}etri$, is the Kleisli category determined by $\langle pc, cp \rangle$. ❑

5.2 Horizontal Compositionality

In the following proposition, we show that, for some given refinement morphisms of nonsequential automata, the morphism uniquely induced by the product construction is also a refinement morphism and thus, the *horizontal compositionality is achieved.*

Proposition 5.10 Let $\{\varphi_i: A_i \to B_i\}_{i \in I}$ be a an indexed family of $\mathcal{RefNA}ut$-morphisms. Then $\times_{i \in I} \varphi_i: \times_{i \in I} A_i \to \times_{i \in I} B_i$ is a $\mathcal{RefNA}ut$-morphism.

Proof: For simplicity, $\times_{i \in I}$ and $+_{i \in I}$ are abbreviated by $\times_i$ and $+_i$, respectively. Let $\times_i \varphi_i: \times_i A_i \to \times_i tcB_i$ be the morphism uniquely induced by the product construction in $\mathcal{NA}ut$. Now, we have just to prove that $\times_i \varphi_i$ is a $\mathcal{RefNA}ut$-morphism. Since $tc = cn \circ nc$ and cn preserves limits, then $\times_i \varphi_i: \times_i A_i \to cn(\times_i ncB_i)$. Since $\times_i ncB_i$ and $+_i ncB_i$ are isomorphic then, up to an isomorphism, $\times_i \varphi_i: \times_i A_i \to cn(+_i ncB_i)$. Since nc preserves colimits, then $\times_i \varphi_i: \times_i A_i \to cn \circ nc(+_i B_i)$. Since $\times_i B_i$ and $+_i B_i$ are isomorphic then, up to an isomorphism, $\times_i \varphi_i: \times_i A_i \to tc(\times_i B_i)$. Therefore, $\times_i \varphi_i$ is a $\mathcal{RefNA}ut$-morphism. ❑

However, the category of Petri nets lacks the horizontal compositionality requirement. We can summarize by just saying that tc is not a continuous functor at net level: it does not preserve limits.

Proposition 5.11 The endofunctor tc: $\mathcal{LP}etri^{\bullet} \rightarrow \mathcal{LP}etri^{\bullet}$ does not preserve products.
Proof: To prove that tc does not preserve products we have just to show an example. Consider the nets N given by the transition a: A → B, M given by the transition x: X → Y and the product N×M with the single transition a | x: A⊕X → B⊕Y. Then, for instance, 2a: 2A → 2B, 3x: 3X → 3Y and 2a‖3x: 2A⊕3X → 2B⊕3Y are transitions of tcN, tcM and tcN× tcM, respectively. However, 2a‖3x is not a transition of tc(N×M) and thus, tc does not preserve products. ❑

6 Concluding Remarks

In this paper we propose a categorial semantic domain for concurrent systems which satisfies the diagonal compositionality (or modularity) requirement, i.e., refinement compose (vertical compositionality) and distributes over parallel composition (horizontal compositionality). It is basically a labeled transition system with a monoidal structure on transitions making explicit which transitions are independent of which. The construction is inspired by Petri nets are monoids [12] and extends the notion of independence relation of asynchronous transition system [2].

A refinement or implementation morphism is defined as a special morphism where the target object is enriched with all conceivable sequential and concurrent computations that can be split into sequential and concurrent permutations of the original transitions respecting source and target states. Then we prove that, for nonsequential automata, the diagonal compositionality requirement is achieve.

The clarification of the relationship between nonsequential automata and Petri nets is done through translation functors extending the approach in [15] where a scene for a formal classification of models for concurrency is set. From the steps of abstractions that are involved, we can infer that nonsequential automata are more concrete then Petri nets. Moreover, the category of Petri nets are isomorphic the subcategory of nonsequential automata.

An important result is that, while the vertical compositionality is achieved by all related models, the horizontal compositionality "starts" at nonsequential automaton, meaning that nonsequential automaton is the least concrete (or more abstract) level among related models which satisfies the diagonal compositionality requirement.

With respect to further work, the next step is an extension toward a semantic domain for object-oriented concepts following the ideas in this paper and the functorial operations for encapsulation and interaction proposed in [11].

References

1 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.

2 M. A. Bednarczyk, *Categories of Asynchronous Systems*, Ph.D. thesis, technical report 1/88, University of Sussex, 1988.

3 A. Corradini, *An Algebraic Semantics for Transition Systems and Logic Programming*, Ph.D. thesis, technical report TD-8/90, Università di Pisa, 1990.

4 J. F. Costa, *Fundamentos Matemáticos da Concorrência*, Ph.D. thesis, UTL/IST/Departamento de Informática, Lisbon, 1991.

5 R. Gorrieri, *Refinement, Atomicity and Transactions for Process Description Language*, Ph.D. thesis, Università di Pisa, 1990.

6 S. Mac Lane, *Categories for the Working Mathematician*, Springer-Verlag, 1971.

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, *Marked Petri Nets*, IST, Lisbon, 1995. To appear in RITA - Revista de Informatica Teórica e Aplicada, UFRGS, Brazil.

9 P. B. Menezes, J. F. Costa, *Synchronization in Petri Nets*, preprint IST/DM/2-94, IST, Lisbon, 1993. Revised version to appear in Fundamenta Informaticae, IOS Press.

10 P. B. Menezes, J. F. Costa, *Compositional Refinement of Concurrent Systems*, preprint IST/DM/26-94, IST, Lisbon, 1994. Revised version to appear in the Journal of the Brazilian Computer Society - Special Issue on Parallel Computation.

11 P. B. Menezes, J. F. Costa, A. Sernadas, *Refinement Mapping for General (Discrete Event) Systems Theory*, to appear in the proceedings of EUROCAST 95, LNCS, Springer-Verlag.

12 J. Meseguer, U. Montanari, *Petri Nets are Monoids*, Information and Computation 88, pp. 105-155, Academic Press, 1990.

13 R. Milner, *Communication and Concurrency*, Prentice Hall, 1989.

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.