Artigo Técnico

Revista de
Informática
Teórica e Aplicada

Marked Petri Nets within a Categorial Framework

P.Blauth MENEZES
Departamento de Matemática, Instituto Superior Técnico
Av. Rovisco Pais, 1096 Lisboa Codex, Portugal
blauth@raf.ist.utl.pt

Abstract

Well know categories of Petri nets lack coproducts and some restrictions on nets, morphisms or initial markings are required in order to guarantee the existence of colimits. Categories of Petri nets equipped with a set of initial markings (instead of a single initial marking) are introduced. It is shown that the proposed categories of nets are complete and cocomplete. Moreover, interpretations of limits and colimits are adequate for expressing semantics of concurrent systems. Examples of structuring and modeling of behavior of nets using categorial constructions based on limits and colimits are provided.

Keywords. Petri nets, net-based semantics, concurrency, structuring of nets, refinement, initial markings, token game, category theory.

1. Introduction

Petri nets are one of the first models for concurrency developed and are widely used in many applications. Recently, categorial frameworks based on Petri nets have been proposed for expressing the semantics of concurrent systems in the so called true concurrency approach as in [Meseguer & Montanari 90] and [Sassone *et al* 93]. An important justification (among others) for the use of category theory is that of structuring, in the sense that Petri nets

in its original definition are not equipped with compositional operations. A step toward structuring is provided in [Winskel 84] and [Winskel 87] where the categorial constructions of product and coproduct stand for parallel and nondeterministic composition operation of nets, respectively. However, if an initial marking is added to the net structure, the categories in [Winskel 87] do not have coproducts. Since Petri nets with an initial marking are used for defining the operational semantics for concurrent languages (see, for instance, [Degano *et al* 88], [Degano & Montanari 87], [Winskel 84], [Olderog 87] and [Glabbeek & Vaandrager 87]), in order to guarantee the existence of coproducts Winskel restricted his categories to safe nets and morphisms. In [Meseguer & Montanari 90] a less restrictive solution is proposed: an initial marking may have at most one token in each place. The resulting categories have coproducts. However, as illustrated in the Figure 1, the coproduct construction reflects a kind of "total choice" composition with restricted applications for defining operational semantics. A different categorial approach is proposed in [Menezes & Costa 93] where a functorial operation for synchronization of nets is constructed. In this framework, the notion of a coproduct construction is simulated as a special case of synchronization. However, it may be the case that colimits (or coproducts in special) are needed for marked nets for any reason. An interesting example is the use of graph transformation using the so called double pushout approach [Ehrig 79] extended for Petri nets viewed as graphs. In this case, graph transformations extended for Petri nets may have several interpretation such as modeling the token game, dynamic specification of systems or systems refinement. For related work about net refinement using graph transformation, see [Menezes 94].

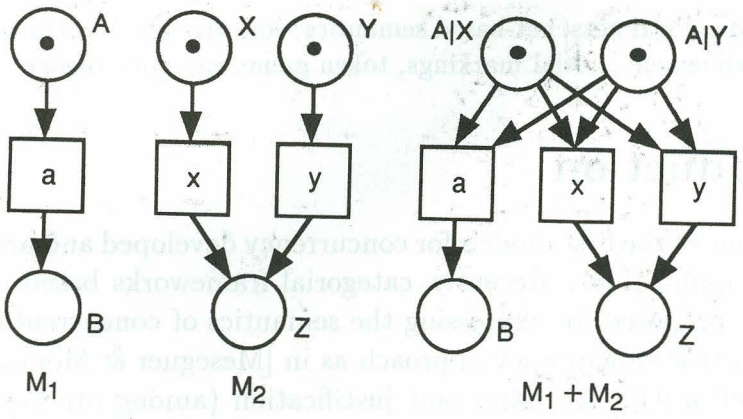

Figure 1. Coproduct of nets as in Meseguer and Montanari

In this paper, we define categories of marked Petri nets with a set of initial markings (instead of a single initial marking) which is also used, for instance, in [Jonsson 90] but in a different framework. In this case, the choice of which initial marking is considered at run time is an external nondeterminism. The only restriction is that initial markings must be preserved by morphisms. The resulting categories of Petri nets are complete and cocomplete. Moreover, the product and coproduct constructions reflect the parallel and asynchronous compositions, respectively. An object determined by a coproduct is the result of putting together "side by side" the component nets. For instance, for the nets M1 and M2 above, the graphical representation of a coproduct between M1 and M2 is illustrated in the Figure 2 (note that it is a distributed diagram).

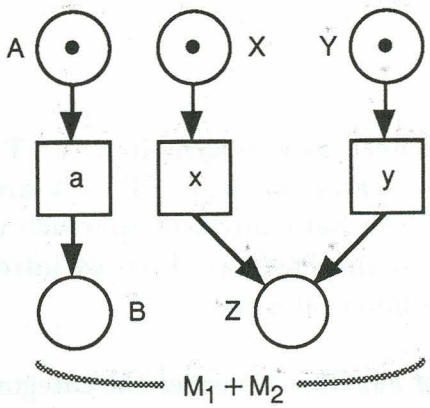

Figure 2. Coproduct of nets with sets of initial markings.

Also, we introduce a generalization of the proposed approach for Petri nets with "colored" markings. The categories of Petri nets for which initial markings are added are taken from [Meseguer & Montanari 90]. In what follows, states of Petri nets are structured as commutative monoids. If free commutative monoids should be considered, the properties about limits and colimits of nets are restricted to the corresponding properties of the category of free commutative monoids.

2. Graphs

A graph can be defined as a set of nodes, a set of arcs, and two functions called source and target which associate for each arc the corresponding source and target nodes. As stated in [Corradini 90] (see also [Asperti & Longo 91]), a graph can be viewed as a diagram in Set, the category of sets and total functions. It means that Set plays the role of "universe of discourse" of the category of graphs, i.e., graphs are defined internally to Set. This suggests a generalization of graphs as diagrams in an arbitrary universe (base) category. This approach is known as internalization and can be extended for reflexive graphs and categories. However, nodes and arcs may be objects of different categories. This leads to the notion of structured (internal) graphs, provided that there are functors from the categories of nodes and arcs to the base category.

2.1 Graph

Traditionally, a graph is defined as a quadruple $< \mathbf{V}, \mathbf{T}, \delta_0, \delta_1 >$ where $\mathbf{V}$ is a set of nodes, $\mathbf{T}$ is a set of arcs and δ_0, δ_1: $\mathbf{T} \rightarrow \mathbf{V}$ are source and target functions. We prefer a different but equivalent approach which is to consider a graph as an element of a comma category. First we introduce the definition of diagonal functors and comma category.

Definition 2.1 Diagonal Functor. Consider the category C. Let C^2 be the category where objects and morphism are pairs of objects and morphisms of C. The diagonal functor Δ_C: $C \rightarrow C^2$ takes each C-object $\mathbf{A}$ into $< \mathbf{A},\mathbf{A} >$ and each C-morphism $\mathbf{f}$: $\mathbf{A} \rightarrow \mathbf{B}$ into $<\mathbf{f},\mathbf{f}>$: $<\mathbf{A},\mathbf{A}> \rightarrow <\mathbf{B},\mathbf{B}>$.

If the category C has binary products (coproducts), then Δ_C has right (left) adjoint which is the functor induced by the product (coproduct) construction. Thus, Δ_C preserves colimits (limits) (see, for instance, [Mac Lane 71]).

Definition 2.2 Comma Category. Let f: $A \rightarrow C$, g: $B \rightarrow C$ be functors. The comma category $f \downarrow g$ is such that:

a) an object is a triple $<\mathbf{A}, \mathbf{f}, \mathbf{B} >$ where $\mathbf{A}$ is an A-object, B is a B-

object and $\mathbf{f}$: $f\mathbf{A} \rightarrow g\mathbf{B}$ is a C-morphism;

b) a morphism is a pair $\mathbf{h} = \langle \mathbf{h}_A, \mathbf{h}_B \rangle$: $\langle \mathbf{A}_1, \mathbf{f}_1, \mathbf{B}_1 \rangle \rightarrow \langle \mathbf{A}_2, \mathbf{f}_2, \mathbf{B}_2 \rangle$ where $\mathbf{h}_A$: $\mathbf{A}_1 \rightarrow \mathbf{A}_2$ is an A-morphism and $\mathbf{h}_B$: $\mathbf{B}_1 \rightarrow \mathbf{B}_2$ is a B-morphism such that $\mathbf{f}_2 \circ f\mathbf{h}_A = g\mathbf{h}_B \circ \mathbf{f}_1$.

c) for an object $\mathbf{S} = \langle \mathbf{A}, \mathbf{f}, \mathbf{B} \rangle$ the identity morphism on $\mathbf{S}$ is $\iota\mathbf{S} = \langle \iota\mathbf{A}: \mathbf{A} \rightarrow \mathbf{A}, \iota\mathbf{B}: \mathbf{B} \rightarrow \mathbf{B} \rangle$;

d) the composition of two morphisms $\mathbf{f} = \langle \mathbf{f}_A, \mathbf{f}_B \rangle$: $\mathbf{S}_1 \rightarrow \mathbf{S}_2$ and $\mathbf{g} = \langle \mathbf{g}_A, \mathbf{g}_B \rangle$: $\mathbf{S}_2 \rightarrow \mathbf{S}_3$ is $\mathbf{g} \circ \mathbf{f} = \langle \mathbf{g}_A \circ \mathbf{f}_A, \mathbf{g}_B \circ \mathbf{f}_B \rangle$: $\mathbf{S}_1 \rightarrow \mathbf{S}_3$.

Proposition 2.3. Consider the categories A, B and the functors f: $A \rightarrow C$, g: $B \rightarrow C$. Then:

a) if A, B are complete and g preserves limits, then $f \downarrow g$ is complete;
b) if A, B are cocomplete and f preserves colimits, then $f \downarrow g$ is cocomplete.
Proof: See, for instance, [Casley 91].

Definition 2.4 Graph. Consider the diagonal functor Δ_{Set}: $Set \rightarrow Set^2$. The category of (small) graphs is the comma category $\Delta_{Set} \downarrow \Delta_{Set}$ denoted by *Graph*.

Therefore, a graph is a triple $G = \langle T, \delta, V \rangle$ where $\delta = \langle \delta 0: T \rightarrow V, \delta 1: T \rightarrow V \rangle$. We denote a graph in the traditional way, i.e., $G = \langle V, T, \delta 0, \delta 1 \rangle$. As expected, a morphism in Graph preserves source and target nodes of transitions. It is usual to write $t: X \rightarrow Y$ to denote $\delta 0(t) = X$ and $\delta 1(t) = Y$ for any t in T. Since Graph is the comma category DSet+ DSet, the proof that Graph is bicomplete is straightforward.

2.2 Internal Graph

In what follows, we internalize the notion of graphs to an arbitrary base category.

Definition 2.5 Internal Graph. Let C be a (base) category. Consider the diagonal functor Δ_C: $C \rightarrow C^2$. The category of internal graphs over C, de-

noted by *Graph(C)*, is the comma category $\Delta_C \downarrow \Delta_C$. □

Therefore, an internal graph is a quadruple $\mathbf{G} = <\mathbf{V}, \mathbf{T}, \delta\mathbf{0}, \delta\mathbf{1}>$ where *V, T* are a *C*-objects and $\delta 0$, $\delta 1$ are *C*-morphisms and a *Graph(C)*-morphism $\mathbf{h}: \mathbf{G}_1 \rightarrow \mathbf{G}_2$ is a pair of *C*-morphisms $\mathbf{h} = <\mathbf{hV}: \mathbf{V}_1 \rightarrow \mathbf{V}_2, \mathbf{hT}: \mathbf{T}_1 \rightarrow \mathbf{T}_2>$ such that source and target nodes of transitions are preserved. The following result is not used in this paper. However, it is interesting by itself, since the existence of limits and colimits in *Graph(C)* is inherited from *C*:

Proposition 2.6. Consider the category C. Then:

a) if *C* is complete, then *Graph(C)* is complete;

b) if *C* is cocomplete, then *Graph(C)* is cocomplete.

Proof: Since *C* is complete (cocomplete) then Δ_C preserves limits (colimits). Since *Graph(C)* is the comma category $\Delta_C \downarrow \Delta_C$, *Graph(C)* is complete (cocomplete).

2.3 Structured Graph

Structured graphs allow the definition of a special kind of graphs where nodes and arcs are objects of different categories. They are defined over internal graphs provided that there are functors from the categories of nodes and arcs to the base category. The source and target morphisms are taken from the base category.

Definition 2.7 Structured Graph. Let *C* be a (base) category and $v: V \rightarrow C$, $t: T \rightarrow C$ be functors. Consider the diagonal functor $\Delta_C: C \rightarrow C^2$. The category of structured graphs over the base category *C* with respect to the functors *v* and *t*, denoted by *Graph(v,t)*, is the comma category $\Delta_C \circ t \downarrow \Delta_C \circ v$. □

Therefore, a structured graph is a quadruple $\mathbf{G} = <\mathbf{V}, \mathbf{T}, \delta\mathbf{0}, \delta\mathbf{1}>$ where **V** is a *V*-object, **T** is a *T*-object and δ_0, δ_1: $t\mathbf{T} \rightarrow v\mathbf{V}$ are *C*-morphisms. A *Graph(v,t)*-morphism h: **G1** $\rightarrow$ **G2** is a pair $\mathbf{h} = < \mathbf{h}_V: \mathbf{V}_1 \rightarrow \mathbf{V}_2, \mathbf{h}_T: \mathbf{T}_1 \rightarrow \mathbf{T}_2 >$ where $\mathbf{h}_V$ is a *V*-morphism and $\mathbf{h}_T$ is a *T*-morphism such that

source and target nodes of transitions are preserved.

Proposition 2.8. Consider the category C and the functors v: $V \rightarrow C$, t: $T \rightarrow C$. Then:

a) if V, T are complete and v preserves limits, then *Graph(v,t)* is complete;

b) if V, $T$are cocomplete and t preserves colimits, then *Graph(v,t)* is cocomplete.

Proof: Since *Graph(v,t)* is the category $\Delta_C \circ t \downarrow \Delta_C \circ v$, the proof is a direct corollary. □

3. Petri Nets

A Petri net, in this paper, means the general case of a place/transition net.

3.1 Petri Net

We introduce the standard definition of a place/transition net as in [Reisig 85] and then Petri nets as graphs. For further details see [Meseguer & Montanari 90].

Definition 3.1 Place/Transition Net. A place/transition net is a triple <**S, T, F**> where S is a set of places, T is a set of transitions and **F: (S x T) + (T x S)** $\rightarrow \mathbb{N}$ is the causal dependency relation (**F** is a multiset and $\mathbb{N}$ is the set of natural numbers). □

The casual dependency relation specifies how many tokens are consumed or produced in each place when a transition fires. For instance, (A, a) $\mapsto$ 3 and (a, B) $\mapsto$ 5 represented in the Figure 3, specify that when the transition a fires 3 tokens are consumed at A and 5 tokens are produced at B. For simplicity, in graphical representation, an arc labeled by 1 has its value omitted.

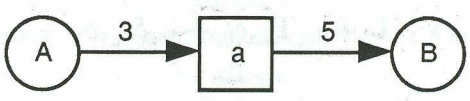

Figure 3. Graphical representation of (A, a) $\mapsto$ 3 and (a, B) $\mapsto$ 5

To define a Petri net as a graph, we consider that nodes are elements of a commutative monoid. In this case, nodes and arcs stand for states and transitions of a net, respectively, where for each transition, n tokens consumed or produced in a place **A** is represented by n_A and n_i tokens consumed or produced simultaneously in a place Ai with i ranging over 1,..., p is represented by $\mathbf{n_1A_1} \oplus \mathbf{n_2A_2} \oplus \cdots \oplus \mathbf{n_pA_p}$ where $\oplus$ is the monoidal operation.

In what follows, *CMon* denotes the category of commutative monoids. Remember that products and coproducts are isomorphic in *CMon*. Also, suppose that k is in $\{\mathbf{0}, \mathbf{1}\}$.

Definition 3.2 Petri Net. The category of Petri nets, denoted by *Petri*, is the category of structured graphs *Graph(cs, id_{Set})*, where *cs: CMon* $\rightarrow$ *Set* is the functor which forgets about the monoidal structure and id_{Set}*: Set* $\rightarrow$ *Set* is the identity functor in *Set.*

Therefore, a Petri net can be viewed as a quadruple $\mathbf{N} = \langle \mathbf{V}, \mathbf{T}, \delta\mathbf{0}, \delta\mathbf{1} \rangle$ where $\mathbf{V} = \langle \mathbf{V}, \oplus, \mathbf{e} \rangle$ is a commutative monoid of states, **T** is a set of transitions and δk: $\mathbf{T} \rightarrow cs\mathbf{V}$ are total functions. A *Petri*-morphism **h**: $\mathbf{N_1} \rightarrow \mathbf{N_2}$ is a pair $\mathbf{h} = \langle \mathbf{h}_V\mathbf{:}\ \mathbf{V_1} \rightarrow \mathbf{V_2},\ \mathbf{hT:}\ \mathbf{T_1} \rightarrow \mathbf{T_2} \rangle$ where $\mathbf{h}_V$ is a *CMon*-morphism and $\mathbf{h}_T$ is a total function such that source and target states of transitions are preserved.

Proposition 3.3 The category Petri is complete and cocomplete.
Proof: Since Petri is the comma category $\Delta_{Set} \circ id_{Set} \downarrow \Delta_{Set} \circ cs$ we have just to show that *cs: CMon* $\rightarrow$ *Set* preserves limits. In fact, the functor *sc: Set* $\rightarrow$ *CMon* that takes each set into its corresponding free commutative monoid is left adjoint to *cs.* □

In Petri, the coproduct and product constructions represent the asynchronous and synchronous composition of nets, respectively. The resulting objects of the product and coproduct of nets $\mathbf{N_1} = \langle \mathbf{V_1}, \mathbf{T_1}, \delta_{0_1}, \delta_{1_1} \rangle$, $\mathbf{N_2} = \langle \mathbf{V_2}, \mathbf{T_2}, \delta_{0_2}, \delta_{1_2} \rangle$ are as follows:

$$\mathbf{N_1} + \mathbf{N_2} = \langle \mathbf{V_1} +_{CMon} \mathbf{V_2}, \mathbf{T_1} +_{Set} \mathbf{T_2}, \delta_{0_1} +_{Set} \delta_{0_2}, \delta_{1_1} +_{Set} \delta_{1_2} \rangle$$

where the morphisms δ_{k_1} x *Set* δ_{k_2}, δ_{K_1} + *Set* δ_{k_2} are uniquely induced by the product and coproduct in Set, respectively. A pair of transitions $(\mathbf{t_1}, \mathbf{t_2})$

is denoted by $t_1 \mid t_2$ meaning that they are synchronized.

Example 3.4 Consider the Figure 4. Then, $N_1 + N_2$ and $N_1 \times N_2$ represent the resulting objects of a coproduct and product between N_1 and N_2 in *Petri*, respectively. Note that a coproduct of nets is the result of putting together "side by side" the component nets. □

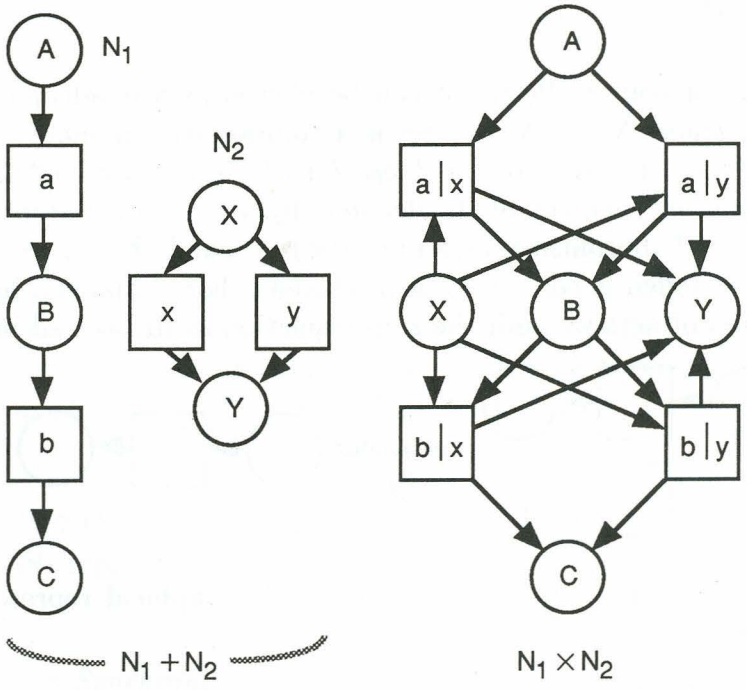

Figure 4. Coproduct and product in Petri

3.2 Pointed Petri Net

The category of pointed Petri nets is such that the set of transitions has a distinguished element. In a net morphism, the mapping of a transition into the distinguished transition is equivalent to forget or erase that transition. The resulting category of pointed Petri nets is complete and cocomplete, where the coproduct construction also reflects the asynchronous composition. However, a product expresses all possible combination between component transitions.

In what follows, $Set^{\bullet}$ denotes the category of pointed sets.

Definition 3.5 Pointed Petri Net. The category of pointed Petri nets, denoted by *Petri✓*, is the category of structured graphs *Graph(csp,$id_{Set^\bullet}$)*, where cs_p: *CMon* $\rightarrow$ *Set*$^\bullet$ is the functor which forgets about the monoidal structure such that the unity element of the monoid is taken into the distinguished element of the pointed set and $id_{Set^\bullet}$: *Set*$^\bullet$ $\rightarrow$ *Set*$^\bullet$ is the identity functor in Set$^\bullet$. □

Therefore, a pointed Petri net can be viewed as a quadruple $\mathbf{N} = <\mathbf{V}, \mathbf{T}, \delta_0, \delta_1 >$ where $\mathbf{V} = <\mathbf{V}, \oplus, \mathbf{e}>$ is a commutative monoid of states, T is a pointed set of transitions and δ_0, δ_1: $T \rightarrow csp$ are *Set*$^\bullet$-morphisms. The distinguished element of **T**, denoted by ✓, is called skip transition. Since δ_k are *Set*$^\bullet$-morphisms, $\delta_0(✓) = \delta 1(✓) = \mathrm{e}$ and thus, ✓ is an isolated transition (no token is consumed or produced). For simplicity, in graphical representation of nets we omit the skip transition as illustrated in Figure 5.

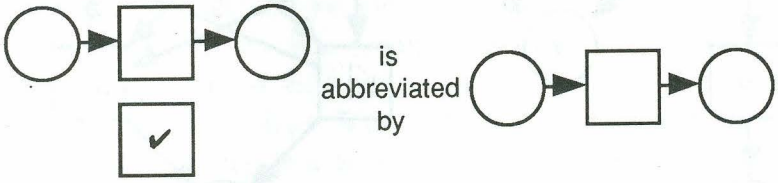

Figure 5. The transition skip is omitted in graphical representation of pointed nets

A Petri✓-morphism $\mathbf{h}:\mathbf{N}_1 \rightarrow \mathbf{N}_2$ is a pair $\mathbf{h} = < \mathbf{h}_V: \mathbf{V}_1 \rightarrow \mathbf{V}_2, \mathbf{h}_T: \mathbf{T}_1 \rightarrow \mathbf{T}_2 >$ where $\mathbf{h}_V$ is a *CMon*-morphism and $\mathbf{h}_T$ is a *Set*$^\bullet$-morphism such that source and target states of transitions are preserved. Therefore, any transition may be forgotten, provided that source and target states are taken into the unity of the monoid.

Proposition 3.6 The category Petri✓ is complete and cocomplete.
Proof: Since Petri✓ is the comma category $\Delta_{Set^\bullet} \circ id_{Set^\bullet} \downarrow \Delta_{Set^\bullet} \circ cs_p$ we have just to show that cs_p: *CMon* $\rightarrow$ *Set*$^\bullet$ preserves limits. In fact, the functor sc_p: $_{Set^\bullet} \rightarrow$ *CMon* that takes each pointed set into the corresponding free commutative monoid where the distinguished element is taken into the unity of the monoid, is left adjoint to cs_p. □

In Petri✓, the coproduct and product constructions represent the asyn-

chronous and parallel composition of nets, respectively. The resulting objects of the product and coproduct of nets $\mathbf{N}_1 =< \mathbf{V}_1, \mathbf{T}_1, \delta_{0_1}, \delta_{1_1} >$, $\mathbf{N}_2 =<\mathbf{V}_2, \mathbf{T}_2, \delta_{0_2}\ \delta_{12} >$ are as follows:

$$\mathrm{N_1xN_2} = < \mathrm{V_1x}_{CMon}\mathrm{V_2}, \mathrm{T_1x}_{Set^\bullet}\mathrm{T_2},\ \delta_{0_1}\mathrm{x}_{Set^\bullet}\delta_{0_2}, \delta_{1_1}\mathrm{x}_{Set^\bullet}\mathrm{x}\delta_{1_2} >$$

$$\mathrm{N_1+N_2} = = < \mathrm{V_1+}_{CMon}\mathrm{V_2},\ \mathrm{T_1+}_{Set^\bullet}\mathrm{T_2},\ \delta_{0_1}+_{Set^\bullet}\ \delta_{02},\ \delta_{11}+_{Set^\bullet}+\delta_{12} >$$

where the morphisms $\delta_{k_1}\mathrm{x}Set^\bullet+\delta_{k_2}$, $\delta_{k_1}+Set^\bullet+\delta_{k_2}$ are uniquely induced by the product and coproduct in $Set^\bullet$, respectively. A pair of transitions $(\tau, \mathbf{t})$ or $(\tau, \mathbf{t})$ is denoted just by $\mathbf{t}$ meaning that the transition $\mathbf{t}$ is not synchronized.
Since coproducts in $Set^\bullet$ are isomorphic to coproducts in *Set*, a coproduct of two nets is analogous to the one in *Petri*, i.e., it is the result of putting "side by side" the component nets. However, a product is quite different. To see the difference, remember that, for sets $\mathbf{A}$ and $\mathbf{B}$, $\mathbf{A}\checkmark\mathrm{x}_{Set^\bullet}B\checkmark$ is isomorphic to $\mathbf{A}+_{Set}(A\mathrm{x}_{Set}B)+_{Set}B$, where $\mathbf{A}\checkmark$, $\mathbf{B}\checkmark$ are $\mathbf{A}$, $\mathbf{B}$ canonically extended as pointed sets.

Example 3.7 Consider the nets $\mathbf{N}_1$ and $\mathbf{N}_2$ illustrated in the Figure 6. Then, $\mathbf{N}_1$ x $\mathbf{N}_2$ represents the resulting object of a product of $\mathbf{N}_1$ and $\mathbf{N}_2$ in *Petri*$\checkmark$. □

Remark 3.8 Synchronization of Petri Nets. In [Menezes & Costa 93], we construct a functorial operation for synchronization of nets, defined for transition calling and transition sharing. It is defined using the fibration technique. The synchronization operation erases from the parallel composition (categorial product) of given pointed nets all those transition which do not reflect the given synchronization specification. For instance, in the Example 3.7, if a shares x (i.e., the happening of a leads to the synchronous happening of x and vice-versa) the functorial operation erases from $\mathbf{N_1xN_2}$ all transitions related to a or x except $\mathbf{a|x}$ (i.e., erases $\mathbf{a}$, $\mathbf{x}$, and $\mathbf{b|x}$). □

Remark 3.9 Adjunction Between Petri and Petri$\checkmark$. As stated in [Meseguer & Montanari 90], there is an obvious forgetful functor *u: Petri*$\checkmark$ $\rightarrow$ *Petri* which forgets about the pointed structure of the transitions. This functor has a left adjoint *f: Petri* $\rightarrow$ *Petri*$\checkmark$ which takes each Petri net into a pointed

Petri net where the set of transitions is canonically extended as a pointed set and the source and target functions are extended such that the distinguished transition is taken into the unity of the monoid. □

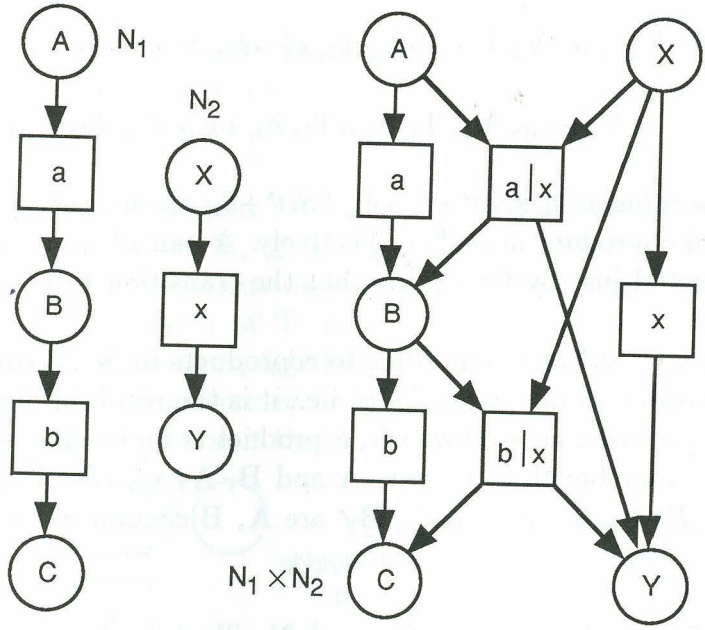

Figure 6. Product in *Petri*✓

4. Marked Petri Nets

A marked (pointed) Petri net is a (pointed) Petri net endowed with a set of initial markings. The resulting categories of marked Petri nets and marked pointed Petri nets are complete and cocomplete.

Definition 4.1 Marked Petri Net, Marked Pointed Petri Net. Consider the categories *Petri*, *Petri*✓ and the identity functors id_{Set}, $id_{Set^\bullet}$. Let ps: *Petri* $\rightarrow$ *Set*, ps_p: *Petri*✓ $\rightarrow$ *Set*$^\bullet$ be forgetful functors such that each net is taken into its corresponding set of transitions. Then:

a) the category of Petri nets with initial markings or initial states is the comma category $id_{Set} \downarrow ps$ denoted by *MPetri*;

b) the category of pointed Petri nets with initial markings or initial states is the comma category $id_{Set^\bullet} \downarrow ps_p$ denoted by *MPetri*✓. □

Therefore, a (pointed) Petri net with initial markings is a triple $\mathbf{M} = <\mathbf{N}, \mathbf{init}, \mathbf{I}>$ where $\mathbf{N} = <\mathbf{V}, \mathbf{T}, \delta_0, \delta_1>$ is a (pointed) Petri net, I is a (pointed) set of *initial markings* and init: $I \rightarrow V$ is a (pointed) total function which instantiates the initial states into the states of **N**. Thus, a net **M** may also be viewed as $\mathbf{M} = <\mathbf{V}, \mathbf{T}, \delta_0, \delta_1, \mathbf{I}, \mathbf{init}>$. For simplicity, if init is an inclusion, it is omitted, i.e., $<\mathbf{V}, \mathbf{T}, \delta_0, \delta_1, \mathbf{I}, \mathbf{init}>$ is abbreviated by $<\mathbf{V}, \mathbf{T}, \delta_0, \delta_1, \mathbf{I}>$.
A marked (pointed) Petri net morphism $\mathbf{h}: \mathbf{M}_1 \rightarrow \mathbf{M}_2$ is a pair $\mathbf{h} = < \mathbf{h}_N: \mathbf{N}_1 \rightarrow \mathbf{N}_2 , \mathbf{h}_I: \mathbf{I}_1 \rightarrow \mathbf{I}_2 >$ such that initial markings are preserved. Since $\mathbf{h}_N$ is a pair $\mathbf{h}_N = <\mathbf{hV}, \mathbf{h}_T>$, a morphism **h** is also represented as a triple $\mathbf{h} = <\mathbf{h}_V, \mathbf{h}_T, \mathbf{h}_I>$.
In graphical representation of marked nets, each initial marking is associated to a different symbol such as circle, star, square, etc. (see example below).

Remark 4.2 Colored Petri Nets. The approach proposed for marked Petri nets can easily be extended for several sets of distinguished markings. For instance:

a) initial and final markings: consider the functors $\gg: Set^2 \rightarrow Set$, $\gg p: (Set^\bullet)^2 \rightarrow Set^\bullet$ induced by the coproduct construction in *Set* and $Set^\bullet$, respectively. Then, $\gg\downarrow ps$ and $\gg p\downarrow ps_p$ are the categories of Petri nets and pointed Petri nets with initial and final markings. In these cases, a net **M** can be represented as $\mathbf{M} = <\mathbf{V}, \mathbf{T}, \delta_0, \delta_1, \mathbf{I}, \mathbf{F}, \mathbf{init}, \mathbf{final}>$ where **I** is a (pointed) set of initial states, **F** is a (pointed) set of final states and **init**, **final** are the corresponding instantiations morphisms;

b) colored markings: colored markings generalizes the definition above. Consider the categories *Hue*, $Color = Set^{Hue}$, $Color^\bullet = (Set^\bullet)^{Hue}$ and the functors $paint: Color \rightarrow Set$, $paint_p: Color^\bullet \rightarrow Set^\bullet$ induced by the coproduct constructions reflecting the combination of colors. Then, $paint\downarrow ps$ and $paint_p \downarrow ps_p$ are the categories of colored and pointed colored Petri nets. □

Proposition 4.3 The categories *MPetri* and *MPetri*✓ are complete and cocomplete.

Proof: Since *MPetri* and *MPetri*$\checkmark$ are the comma category $id_{Set} \downarrow ps$ and $id_{Set^\bullet} \downarrow ps_p$, respectively, we have only to prove that *ps: Petri* $\rightarrow$ *Set* and ps_p: *Petri*$\checkmark$ $\rightarrow$ *Set*$^\bullet$ preserve limits. Then:

a) consider the initial *Set*-object $\{\ \}$ and the functor *sp: Set* $\rightarrow$ *Petri* such that for all set $\mathbf{V}$, $sp\mathbf{V} =< \mathbf{V}^{\oplus}, \{\ \}, !, !>$ where $\mathbf{V}^{\oplus}$ is the monoid freely generated from $\mathbf{V}$ and $!: \{\ \} \rightarrow cs\mathbf{V}^{\oplus}$ is unique. The functor *sp* is left adjoint to *ps*;

b) consider the initial *Set*$^\bullet$-object $\{\checkmark\}$ and the functor sp_p: *Set*$^\bullet$ $\rightarrow$ *Petri*$\checkmark$ such that for all pointed set $\mathbf{V}$, $sp_p\mathbf{V} =< \mathbf{V}^{\oplus}, \{\checkmark\}, !, !>$ where $\mathbf{V}^{\oplus}$ is the monoid freely generated from $\mathbf{V}$ such that the distinguished element is taken into the unity of the monoid and $!: \{\checkmark\} \rightarrow cs_p\mathbf{V}^{\oplus}$ is unique. The functor sp_p is left adjoint to ps_p. □

Note that the above result may be easily extended for Petri nets with colored markings, depending on the properties about limits and colimits of the category of colors and the paint functor considered. The product and coproduct constructions in *MPetri* and *MPetri*$\checkmark$ have the same interpretation as in *Petri* and *Petri*$\checkmark$, respectively. The resulting objects of the product and coproduct of nets $M1 =< V_1, T_1, \delta_{0_1}, \delta_{1_1}, I_1, init_1 >$, $M2 =< V_2, T_2, \delta_{0_2}, \delta_{1_2}, I_2, init_2 >$ are as follows (depending on the category considered, $\mathbf{M}_1$ and $\mathbf{M}_2$ stand for *MPetri*-objects or *MPetri*$\checkmark$-objects):

$M_1 x_{MPetri} M_2 =< V_1 x_{CMon} V_2,\ T_1 x_{Set} T_2,\ \delta_{0_1} x \delta_{0_2},\ \delta_{1_1} x \delta_{1_2},\ I_1 x_{Set} I_2, init_1 x init_2 >$
$M_1 +_{MPetri} M_2 =< V_1 +_{CMon} V_2,\ T_1 +_{Set} T_2,\ \delta_{0_1} + \delta_{0_2},\ \delta_{1_1} + \delta_{1_2},\ I_1 x_{Set} I_2, init_1 + init_2 >$
$M_1 x_{MPetri_\checkmark} M_2 =< V_1 x_{CMon} V_2,\ T_1 x_{Set^\bullet} T_2,\ \delta_{0_1} x \delta_{0_2},\ \delta_{1_1} x \delta_{1_2},\ I_1 x_{Set^\bullet} I_2, init_1 x init_2 >$
$M_1 +_{MPetri_\checkmark} M_2 =< V_1 +_{CMon} V_2,\ T_1 +_{Set^\bullet} T_2,\ \delta_{0_1} + \delta_{0_2},\ \delta_{1_1} + \delta_{1_2},\ I_1 +_{Set^\bullet} I_2, init_1 + init_2 >$

where the morphisms $\delta_{k1} x \delta_{k2}$, $\delta_{k1} + \delta_{k2}$, $init_1 x\ init_2$ and $init_1 +\ init_2$ are uniquely induced by the product and coproduct constructions in the corresponding categories.

Example 4.4 Consider the Figures 7, 8 and the following symbols for initial markings:

• A • X ■ X⊕Y • A⊕X • A⊕X⊕Y

Then:

a) A coproduct between $\mathbf{M_1}$, $\mathbf{M_2}$ in both categories are the result of putting "side by side" the component nets with **A**, **X**, **X**$\oplus$**Y** as the initial markings, as illustrated in Figure 7;
b) A product between $\mathbf{M_1}$, $\mathbf{M_2}$ in *MPetri* reflects the synchronous composition and has **A**$\oplus$**X**, **A**$\oplus$**X**$\oplus$**Y** as initial markings and a product in *MPetri*$\surd$ reflects the parallel composition and has **A**, **X**, **X**$\oplus$**Y**, **A**$\oplus$**X**, **A**$\oplus$**X**$\oplus$**Y** as initial markings, as illustrated in the Figure 8. □

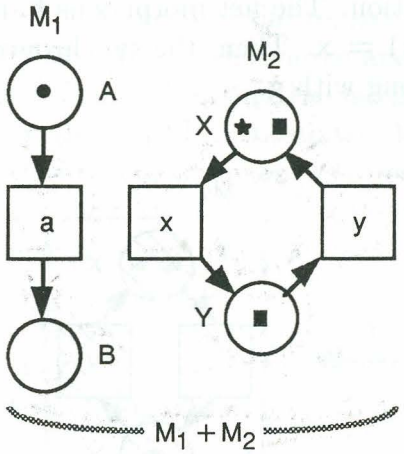

Figure 7. Coproduct in *MPetri* and in *MPetri*$\surd$

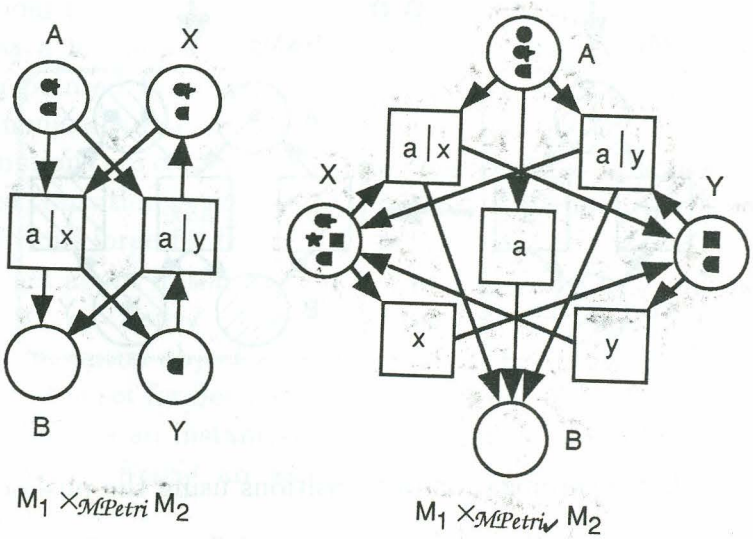

Figure 8. Product in *MPetri* and in *MPetri*$\surd$

For simplicity, in the following examples, dashed boxes and circles identify those parts which are preserved by morphisms.

Example 4.5 Synchronization of MPetri-nets using the pushout construction. Consider the Figure 9. Suppose that we want to compose the nets $\mathbf{M}_1$ and $\mathbf{M}_2$ such that **a** shares **x**, i.e., transitions **a** and **x** communicate without buffer as in CCS [Milner 80] or CSP [Hoare 85]. Let **Sync** be a net with only one isolated transition. The net morphisms **f** and **g** are such that $\mathbf{f}_T(\mathbf{a}\|\&\mathbf{x}) = \mathbf{a}$ and $\mathbf{g}_T(\mathbf{a}\&\mathbf{x}) = \mathbf{x}$. Then, the synchronized net $\mathbf{M}_1\&M_2$ is given by the pushout of **f** along with **g**.

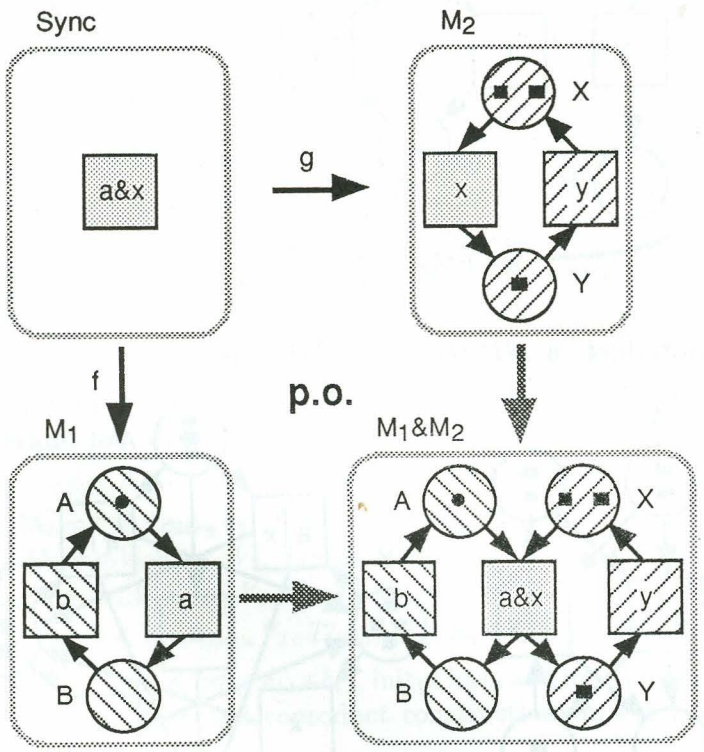

Figure 9. Synchronization of transitions using the pushout construction

Note that the above construction is able to represent only symmetrical synchronization of transitions. See Remark 3.8 for a generalized approach. The above construction may also be used for identifying places or both, places and transitions.

Example 4.6 Transformation of MPetri-nets using the double pushout approach. In the Figure 10, consider the net **Orig** to be transformed, the rule **r** =<**left, right**> which specifies the replacement (the transition **a** is replaced by transitions $\mathbf{a}_1$ and $\mathbf{a}_2$, the source and target states are preserved, the initial marking **A** is forgotten and the initial marking **2B** is introduced) and the morphism **redex** which instantiates the part to be replaced in the original net. The pushout complement *(1)* determines the net **Compl** (in this paper we do not show the existence of pushout complements for nets) and then the pushout *(2)* determines the transformed net **Transf**. This example, illustrates a replacement of a transition (which can be interpreted as a refinement of the transition **a** or as a dynamic specification) and also a replacement of an initial marking (which can be viewed as a modeling of a token game such as the firing of the transition **a**). □

Remark 4.7 Adjunctions Between Categories of Petri Nets. Consider the Figure 11. There is an obvious forgetful functor u_m: *MPetri*✓ → *MPetri* which forgets about the pointed structure of transitions and initial markings. This functor has a left adjoint f_m: *MPetri* → *MPetri*✓ which takes each Petri net into a pointed Petri net where the sets of transitions and initial markings are canonically extended as pointed sets and the source, target and instantiation functions are canonically extended as *Set*•-morphisms such that the distinguished transition is taken into the unity of the monoid. Moreover, there are also obvious forgetful functors *v: MPetri* → *Petri*, v_p: *MPetri*✓ → *Petri*✓ which forget about initial markings. These functors have left adjoints *g: Petri* → *MPetri*, g_p: *Petri*✓ → *MPetri*✓, respectively, such that each net is extended with an initial object as a set of initial markings (an empty set for Set and a singleton set for Set) and the unique morphism from the initial object into the states as an instantiation morphism. The functor *u: Petri*✓ →*Petri* and *f: Petri* → *Petri*✓ are as in Remark 3.9. □

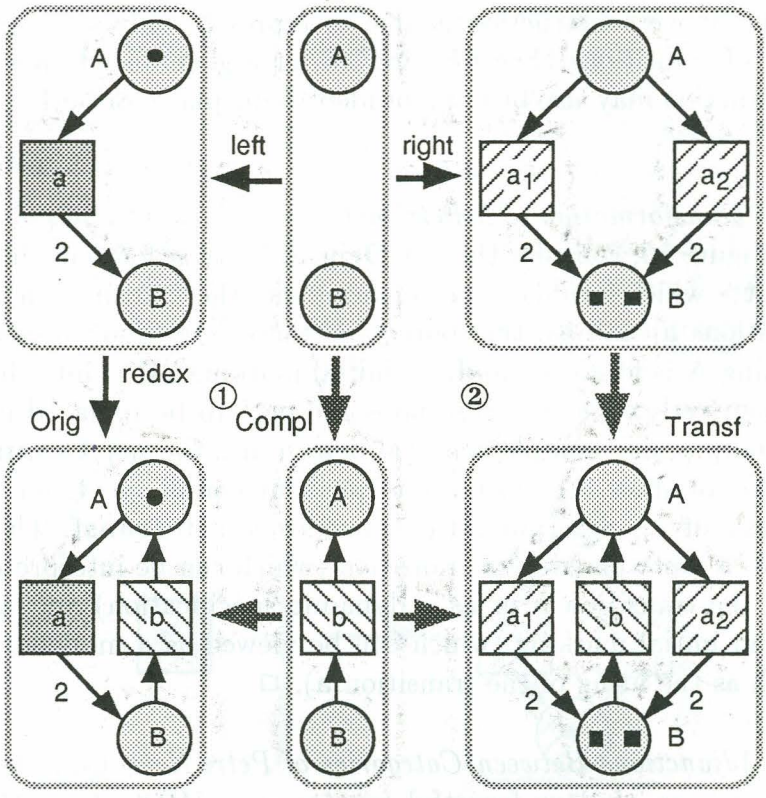

Figure 10. Transformation of a net

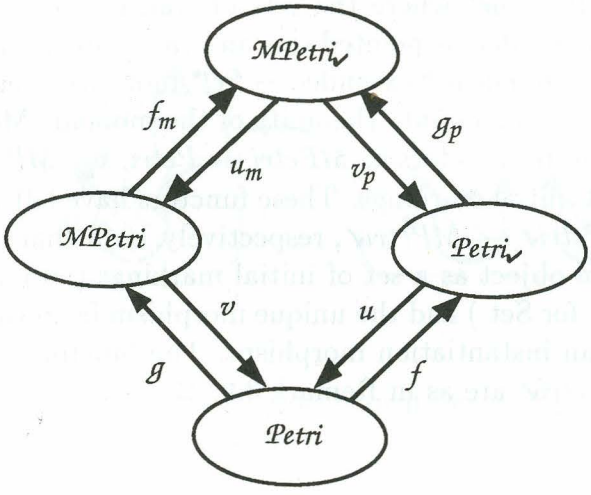

Figure 11. Adjunctions between categories of Petri nets

5. Concluding Remarks

Several categorial frameworks based on Petri nets have been proposed recently for expressing the semantics of concurrent systems in the so called true concurrency approach where structuring of nets is done through categorial constructions based on limits and colimits. However, well know categories of marked Petri nets lack coproducts and some restrictions on nets, morphisms or initial makings are required in order to guarantee the existence of coproducts.
We introduce categories of Petri nets inspired by [Meseguer & Montanari 90], equipped with sets of initial markings (instead of a single initial marking). No further restrictions on nets, morphisms or initial markings are required. We show that the resulting categories of nets are complete and cocomplete. Moreover, the interpretation of limits and colimits (and coproducts in special) are adequate for expressing semantics of concurrent systems. Examples are provided where the categorial coproduct stand for asynchronous composition of nets, pushout constructions are used for synchronizations and the so called double pushout approach for graph transformation [Ehrig 79] is extended for the proposed categories which can be interpreted as dynamic specification of nets, refinement of nets or modeling of token games.
Now, we are working on partial graphs and partial morphisms, where the category of partial marked Petri nets is special case. The main goal is to extend the approach in [Menezes 94] with initial markings, where the so called single pushout approach for graph transformation (see, for instance, [L we 93]) stands for refinement of nets. Also, we are reviewing the works on non-sequential automata in [Menezes & Costa 94] and in [Menezes & Costa 94b] in order to improve with initial states as proposed in this paper.

Acknowledgments

The author is grateful to Amílcar Sernadas for many fruitful discussions. 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 (Information Systems - COrrectness and REusability), BRA WG 6112 COMPASS (COM-Prehensive Algebraic approach to System Specification and development) and ESDI under research contract OBLOG (OBject LOGic) in

Portugal.

6. References

[Asperti & Longo 91] 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.

[Casley 91] R. T. Casley, *On the Specification of Concurrent Systems,* Ph.D. thesis, University of Stanford, 1991.

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

[Degano et al 88] P. Degano, R. De Nicola & U. Montanari,*A Distributed Operational Semantics for CCS Based on Condition/Event Systems,* Acta Informatica 26, pp. 59-91, 1988.

[Degano & Montanari 87] P. Degano & U. Montanari, *Concurrent Histories: A Basis for Observing Distributed Systems J. Comput.* System Sci. 34, pp. 422-462, Nos. 2/3, 1987.

[Ehrig 79] H. Ehrig, *Introduction to the Algebraic Theory of Graph Grammars,* 1st Graph Grammar Workshop (V. Claus, H. Ehrig, G. Rozenberg, Eds.), pp. 1-69, LNCS 93, Spring-Verlag, 1979.

[Glabbeek & Vaandrager 87] R. J. van Glabbeek & F. W. Vaandrager, *Petri Net Model for Algebraic Theories of Concurrency,* pp. 224-242, LNCS 259, Spring-Verlag, 1987.

[Hoare 85] C. A. R. Hoare,*Communicating Sequential Processes,* Prentice Hall, 1985.

[Jonsson 90] B. Jonsson, *A Hierarchy of Compositional Models of I/O-Automata,* Symposium on Mathematical Foundations of Computer Science, pp. 347-354,

LNCS 452, Springer-Verlag, 1990.

[Löwe 93] M. Löwe, *Algebraic Approach to Single Pushout Graph Transformation,* TCS 109, pp. 181-224, 1993.

[Mac Lane 71] S. Mac Lane, *Categories for the Working Mathematician,* Springer-Verlag, 1971.

[Menezes 94] P. B. Menezes, *Compositional Reification of Petri Nets through Graph Transformation,* technical report INESC/RT/74-94, preprint IST/DM /25- 94, INESC/IST, Lisbon, 1994. To be submitted.

[Menezes & Costa 93] P. B. Menezes & J. F. Costa, *Synchronization in Petri Nets,* technical report INESC/RT/71-93, preprint IST/DM/2-94, INESC/IST, Lisbon, 1993. Revised version accepted for publication in Fundamenta Informaticae.

[Menezes & Costa 94] P. B. Menezes, *Compositional Reification of Concurrent Systems,* technical report INESC/RT/75-94, preprint IST/DM/26-94, INESC/IST, Lisbon, 1994. Revised version accepted for publication in the Journal of the Brazilian Computer Society Special - Issue on Parallel Computation.

[Menezes & Costa 94b] P. B. Menezes & J. F. Costa, *Object Reification,* technical report INESC/RT/73-94, preprint IST/DM/24-94, INESC/IST, Lisbon, 1994. Revised version accepted for presentation in EUROCAST 95.

[Meseguer & Montanari 90] J. Meseguer & U. Montanari, *Petri Nets are Monoids,* Information and Computation 88, pp. 105-155, Academic Press, 1990.

[Milner 80] R. Milner, *A Calculus for Communicating Systems,* LNCS 92, Springer-Verlag, 1980.

[Olderog 87] E. R. Olderog, *Operational Petri Net Semantics for CSP* Advances in Petri Nets 1987 (G. Rozenberg, Ed), pp. 196-223, Springer-Verlag, 1987.

[Reisig 85] W. Reisig, *Petri Nèts: An Introduction,* EATCS Monographs on Theoretical Computer Science 4, Springer-Verlag, 1985.

[Sassone et al 93] 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.

[Winskel 84] G. Winskel, *Categories of Models for Concurrency,* LNCS 197, Seminar on Concurrency (S. Brookes, Roscoe, G. Winskel, Eds.), pp. 246-267, Springer-Verlag, 1984.

[Winskel 87] G. Winskel, *Petri Nets, Algebras, Morphisms and Compositionality,* Information and Computation 72, pp. 197-238, Academic Press, 1987.

Author: Paulo Fernando Blauth Menezes, graduated in Mathematics and Master in Computer Science by UFRGS, professor of the Theoretical Informatic Department of UFRGS, taked to make Doctorade in the Instituto Superior Técnico, Lisboa, Portugal.