# On the Algebraic Structure of Petri Nets

Vladimiro Sassone

*Dipartimento di Matematica e Informatica, Università di Catania*

This paper retraces, collects, and summarises the contributions of the author — both individually and in collaboration with others — on the theme of algebraic, compositional approaches to the semantics of Petri nets.

## Introduction

An extremely successful line of research in the *semantics of concurrency*, rooted in the very ideas of denotational semantics, is the one following the *algebraic* approach. It focuses on *structural* and *compositional* aspects of systems and behaviours, and the leading idea is to describe them by means of a few basic building blocks and a small number of *combinators* [**39, 62, 38, 64**]. The appeal of this is that it tends to devise neat algebraic structures that capture the *essential* nature of the class of systems considered.

In this paper, we first survey a line of research — detailed in [**58, 24, 59, 86, 87, 16**] — aimed at recasting Petri net *processes* in lieu of ideas from *process algebras* and *categorical algebra*. In particular, we shall focus on Petri net *concatenable processes* [**24, 86**], on *strongly* concatenable processes [**87, 16**], and on their representation in terms of *symmetric monoidal categories*.

*Petri nets* were introduced in the 1960's by Carl Adam Petri in [**74**] (see also the references [**75, 80, 85, 72, 83**]). They are a widely used model for concurrency, attractive from the theoretical point of view because of its simplicity and its intrinsically concurrent and distributed nature, and very successful in applications such system modeling, analysis, and design (see, e.g., [**81, 82, 46, 99**] and browse through the several available computer-aided design, analysis, and verification tools based on Petri nets [**77**]). Actually, 'Petri net' is a rather generic term: in fact, Petri's original idea can be constrained and generalised in many sensible ways, giving rise to several net-based models widely studied in the literature. These range from the essential *elementary* [**84**] and *place/transition nets* [**26**] to the sophisticated *predicate/transition* [**88**] and *coloured nets* [**47, 48, 49**], including *stochastic Petri nets* [**2**] used in simulation and performance evaluation.

Here we shall be concerned exclusively with place/transition (PT) nets — though it would be interesting to explore to what extent these ideas and techniques can be lifted to classes of high-level nets. The reason why PT nets form an important class is that they formalise a very basic model of distributed systems, in which (instances of) places (i.e., tokens) can be understood as available resources, and transitions as concurrent activities that require exclusive use of some of these resources and that, after completion, release new resources (tokens in places) to the environment. Another suggestive possible interpretation is to look at places as 'mailboxes' and at tokens as messages, portraying a view of place/transition nets as a distributed model of concurrency with a form of asynchronous message passing. We shall study PT nets under the banner '*Petri nets are monoids*', initiated by [**58**]. Our first aim will be to axiomatise the (noninterleaving) computations of a net, i.e., its processes, and their structure. We seek an *algebra* to represent them; an algebra

Research partly supported by MURST project *TOSCA: Tipi, Ordine Superiore e Concorrenza*.

where processes can be seen as terms built up from their atomic components and whose algebraic laws can be used to compute with and reason on and them.

The mathematical structures we shall use to this purpose are the symmetric monoidal categories. *Monoidal categories* date back to [**7**] (see [**55**] for an easy thorough introduction and [**28**] for advanced topics). Essentially, a monoidal category is an algebraic theory of so-called 'arrows', or 'morphisms', and of two operations on them, a (partial) *sequential* composition $\_\,;\_$, and a *parallel* composition $\otimes$, the tensor product. But let us proceed orderly. A *category* is a graph equipped with a self-looping edge $id_u$ for each node $u$, and with an associative binary operation $\_\,;\_$ of composition of adjacent edges. Nodes and edges here are called *objects* and *arrows*, and $id_u$ is the *identity* arrow at object $u$, and behaves as a unit under composition. A *functor* is a mapping between categories that behaves homomorphically with respect to $\_\,;\_$ and *id*, i.e., it maps identities to identities. Adding a tensor product to a category amounts to adding to the graph an operation of parallel composition of objects and arrows that behaves well with respect to $\_\,;\_$. In this paper we shall be concerned only with a particular kind of monoidal categories, namely the 'strict' ones.

A *strict monoidal category* is a structure $(\mathsf{C},\otimes,e)$, where $\mathsf{C}$ is a category, $e$ is an object of $\mathsf{C}$, called the *unit* object, $\otimes\colon \mathsf{C}\times\mathsf{C}\to\mathsf{C}$ is a functor that, as an operation of objects and arrows, is associative and admits $e$ and $id_e$ as, respectively, the unit object and arrow. A monoidal category is *symmetric* if, informally, the tensor product is *commutative* up to a chosen family of isomorphisms $c_{u,v}\colon u\otimes v \xrightarrow{\sim} v\otimes u$, for all objects $u,v\in\mathsf{C}$. The collection of the arrows $c_{u,v}$ must be subject to a *naturality* condition [**55**] and to the all important Kelly-MacLane *coherence* axioms [**54, 51**], and is called the *symmetry* of $\mathsf{C}$.

Another relevant application of Petri nets is their use as a semantic basis to interpret concurrent languages, a task that calls for a compositional, '*process algebra-like*' description of nets. And in fact, the literature is rich of examples of process algebras and concurrent programming languages interpreted over the domain of nets, as, e.g., [**71, 34, 23, 20**], and also of real net-based process algebras, such as [**36, 10, 63**]. In particular, [**34**] uses Petri nets to model an algebra of processes and to infer several noninterleaving behavioural equivalences on it, while [**23**] interprets CCS (cf. [**62**]) on nets — taking up a line of research initiated by [**94**], where event structures (cf. [**96**]) were used — based on an operation of decomposition of processes into sequential agents. The decomposition approach is also followed by [**71**], while the semantics for the $\pi$-calculus (cf. [**64**]) presented in [**20**] is based on nets with inhibitory arcs (see, e.g., [**22, 45**]), a powerful extension of PT nets. A related line of research, as already mentioned, takes inspiration from the work on process algebras and set out to design and study net algebras. One of the most prominent approaches among these is the Petri Box calculus [**10**], centered around operations of asynchronous communication and synchronisation, while [**36**] builds on operations of parallel and non-deterministic composition. In a different context, but with a similar vein, [**63**] introduces the notion of named Petri nets and provides a representation for them as an action calculus.

We proceed in our survey by focusing on the algebra of nets developed in [**68**]. That approach is entirely based on a notion of *interface* for Petri nets that specifies what parts of the net are *public*, i.e., accessible to the environment, and what parts are *private*. Also, it partitions public net components in '*input*' places and '*output*' transitions, and dictates the discipline by which nets are composed via a minimal set of combinators forming a rudimentary calculus of nets. The most important of these is a form of asynchronous communication — message passing — by means of which a net may, via its output transitions, send messages to another net, by delivering tokens to the second net's input places. Net

composition is centred on an interesting form of *recursion* consisting of *feeding back* outputs to inputs, yielding a bridge to structures of recent common interest in category theory and in computer science: the *traced monoidal categories* [**50**].

## PETRI NETS IN THE SMALL

Among the semantics proposed for Petri nets, a role of paramount importance is played by the various notions of *process*, e.g. [**76, 35, 9**], whose merit is to provide a faithful account of computations involving many different transitions and of the *causal connections* between the events occurring in computations. This is, in fact, the essence of the *noninterleaving* approach to the semantics of concurrency, where computations are decorated with additional information describing causes and effects that ruled the occurrences of events in them. The mathematical structures arising naturally form this premises are the partially ordered multisets [**79**], *pomsets* for short. Thus, informally speaking, Petri net processes — whose standard version is given by the Goltz-Reisig *non-sequential processes* [**35**] — are net computations together with an explanation of the cause by which each transition has fired, that be represented abstractly by means of ordered sets whose elements are labelled by transitions.

Bare process models, however, fail to bring to the foreground the *algebraic structure* of the space of computations of a net. Our interest, instead, resides on abstract models that capture the mathematical essence of such spaces, possibly axiomatically, roughly in the same way as a prime algebraic domain (or, equivalently, a prime event structure [**96, 98**]) models the computations of a net (see, e.g., [**70**]). The research detailed in [**58, 24, 59, 86, 87**] identifies such structures as *symmetric monoidal categories* — where objects are states, i.e., multisets of tokens, arrows are processes, and the tensor product and the arrow composition model, respectively, the operations of parallel and sequential composition of processes.

At a higher level of abstraction, the next important question concerns the *global structure* of the collection of such spaces, i.e., the axiomatisation '*in the large*' of net computations. In other words, the space of the spaces of computations of Petri nets. Building on [**24, 86**], the work presented in [**87**] shows that the so-called *symmetric Petri categories*, a class of symmetric strict monoidal categories with free (non-commutative) monoids of objects, provide one such an axiomatisation.

In this part, we retrace and illustrate the main results achieved so far along these lines of research by the author, both in joint and individual work. The next one will look at net algebras 'in the large' from a different angle.

### 1. Petri nets as monoids

The idea of looking at nets as *algebraic structures*, e.g. [**80, 97**], has been interpreted in [**58**] by viewing nets as *internal graphs* in categories of sets with structure and using monoidal categories as a suitable semantic framework for them. Precisely, a net is a graph

$$N = (pre_N, post_N \colon T_N \to \mu(S_N))$$

whose nodes form the free commutative monoid $\mu(S_N)$ of the *finite* multisets of $S_N$. Here, $S_N$ and $T_N$ are sets of, respectively, *places* and *transitions*, and $pre_N$ and $post_N$ are functions assigning a *source* and a *target* multiset of places to each transition. Accordingly, a morphism of nets is graph homomorphism $\langle f_t, f_p \rangle$ whose node component respects the

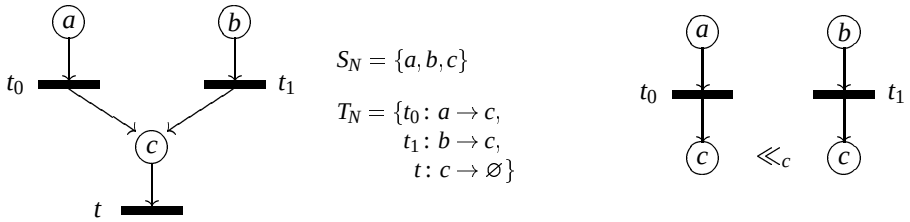


**FIGURE 1.** A net $N$ and one of its two concatenable processes $CP : a + b \to 2c$

monoidal structure on places. This, with the obvious componentwise composition of morphisms, defines the category Petri.

Ideally, Petri net processes are simply computations carrying explicit information about cause/effect relationship between event occurrences. This is conveniently described by defining a process of $N$ to be a map $\pi : \Theta \to N$, where $\Theta$ defines the process 'skeleton' and $\pi$ 'labels' $\Theta$ with places and transitions of $N$ in a way compatible with its structure.

**DEFINITION.** A *process net* is a finite, acyclic net $\Theta$ such that for all $t \in T_\Theta$, $pre_\Theta(t)$ and $post_\Theta(t)$ are sets (as opposed to multisets), and for all $t_0 \neq t_1 \in T_\Theta$,

$$pre_\Theta(t_0) \cap pre_\Theta(t_1) = \varnothing \qquad \text{and} \qquad post_\Theta(t_0) \cap post_\Theta(t_1) = \varnothing.$$

A *process* of $N \in$ Petri is (up to isomorphism) a net morphism $\pi : \Theta \to N$, where $\Theta$ is a process net and $\pi$ maps places to places (as opposed to multisets of places).

Inspired by the work in process algebras, we would like to concatenate a process $\pi_1 : \Theta_1 \to N$ with source $u$ to a process $\pi_0 : \Theta_0 \to N$ with target $u$ by *gluing* appropriately the *terminal* places of $\Theta_0$ and the *initial* places of $\Theta_1$. However, the simple minded attempt fails immediately: due to the ambiguity introduced by multiple instances of places, two processes of $N$ can be composed sequentially in many ways, each of which gives a possibly different process of $N$. In other words, process concatenation has to do with merging *tokens in the process places*, that is instances of places, rather than merging *places*.

## 2. Concatenable processes

It follows from the precedent argument that any attempt to recast the processes of $N$ as an algebra that includes sequential composition must disambiguate each token in a process. This is exactly the idea of *concatenable processes* [**24**]: they are simply processes where, when needed, instances of places (tokens) are distinguished by appropriate decorations, e.g., by ordering the initial and terminal places that carry the same label.

**DEFINITION.** A *concatenable process* of $N$ is a triple

$$(\pi : \Theta \to N, \{<_a\}_{a \in S_N}, \{\ll_a\}_{a \in S_N}),$$

where $\pi$ is a process, and $<_a$ and $\ll_a$ are linear orderings of, respectively, the initial and terminal places of $\Theta$ contained in $\pi_p^{-1}(a)$ (cf. Figure 1).

This immediately yields an operation of concatenation: the ambiguity about multiple tokens is resolved using the additional information given by the orderings (cf. Figure 2).

**DEFINITION.** Let $CP_0 : u \to v$ and $CP_1 : v \to w$ be concatenable processes of $N$, and let $\pi_0 : \Theta_0 \to N$ and $\pi_1 : \Theta_1 \to N$ be their underlying processes. The *sequential composition*, or concatenation, $CP_0 \,;\, CP_1 : u \to w$ is obtained by gluing together $\pi_0$ and $\pi_1$, identifying injectively each terminal place of $\Theta_0$ with an initial place of $\Theta_1$ in the *unique* way compatible with the orderings $\ll_a$ on $\Theta_0$ and $<_a$ on $\Theta_1$ for all $a \in S_N$.

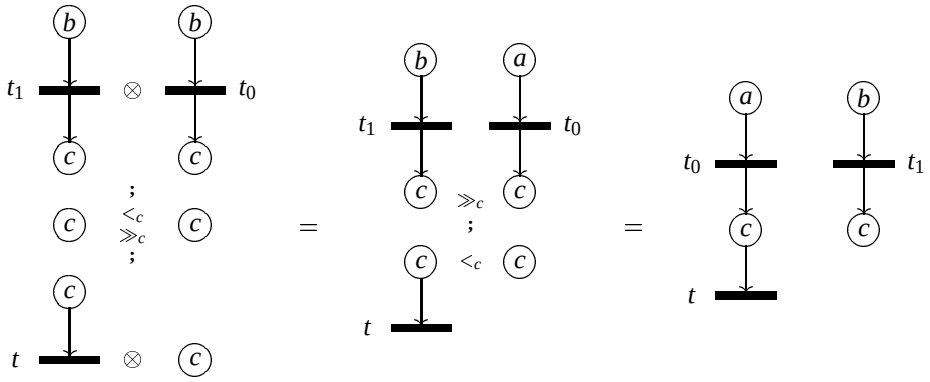


**FIGURE 2.** A net $N$ and its concatenable process $\pi = t_0 \otimes t_1 ; t \otimes id_c$

The existence of concatenation leads easily to the definition of the category of concatenable processes of $N$. It turns out this is a *symmetric strict monoidal category* [**55**] under the tensor product given by the following operation of parallel composition of processes: for $CP_0 : u_0 \to v_0$ and $CP_1 : u_1 \to v_1$, $CP_0 \otimes CP_1 : u_0 + u_1 \to v_0 + v_1$ is obtained by putting $\pi_0$ and $\pi_1$ disjointly side by side and by making the places of $\Theta_0$ precede the places of $\Theta_1$ (cf. Figure 2; consult [**24**] for further examples).

The main result of [**24**] is an axiomatisation of such a category, stated here in the improved enunciation proved in [**86**]. Its relevance is that it describes net behaviours as *algebras* in terms of *universal* constructions.

**THEOREM.** *For any net $N$, there exists a one-to-one correspondence — preserving source, target, sequential and parallel composition (tensor product) of processes (arrows) — between the concatenable processes of $N$ and the arrows of the category $\mathcal{P}(N)$ obtained from the free symmetric strict monoidal category $\mathcal{F}(N)$ on $N$ by imposing the axioms*

$$\begin{aligned} c_{a,b} &= id_{a \otimes b}, && \text{if } a \text{ and } b \text{ are different places of } N, \\ s;t;s' &= t, && \text{if } t \text{ is a transition of } N \text{ and } s \text{ and } s' \text{ are symmetries of } \mathcal{F}(N), \end{aligned}$$

*where $c$, $id$, $\otimes$, and $\_\,;\,\_$ are, respectively, the symmetry isomorphism, the identities, the tensor product, and the composition of $\mathcal{F}(N)$.*

This also yields an equational theory for net processes as, in explicit terms, $\mathcal{P}(N)$ is the category whose arrows are generated by the rules

$$\frac{u \in \mu(S_N)}{id_u : u \to u \text{ in } \mathcal{P}(N)} \qquad \frac{a \text{ and } b \text{ in } S_N}{c_{a,b} : a + b \to b + a \text{ in } \mathcal{P}(N)} \qquad \frac{t : u \to v \text{ in } T_N}{t : u \to v \text{ in } \mathcal{P}(N)}$$

$$\frac{\alpha : u \to v \text{ and } \beta : u' \to v' \text{ in } \mathcal{P}(N)}{\alpha \otimes \beta : u + u' \to v + v' \text{ in } \mathcal{P}(N)} \qquad \frac{\alpha : u \to v \text{ and } \beta : v \to w \text{ in } \mathcal{P}(N)}{\alpha ; \beta : u \to w \text{ in } \mathcal{P}(N)}$$

modulo the axioms expressing that it is a strict monoidal category with composition $\_\,;\,\_$, tensor $\_ \otimes \_$, and symmetry isomorphism $c$ and the two axioms quoted above.

**EXAMPLE.** Figure 2 shows a concatenable process $\pi$ for the net $N$ of Figure 1 that corresponds to the arrow $t_0 \otimes t_1 ; t \otimes id_c$ of $\mathcal{P}(N)$. To exemplify the algebra of processes of $N$, $\pi$ is expressed as parallel ($\_ \otimes \_$) and sequential ($\_\,;\,\_$) composition of simpler processes.

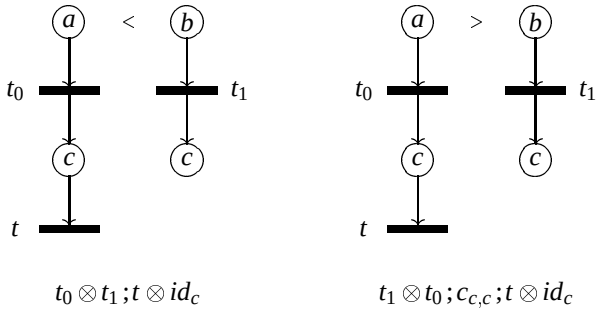


**FIGURE 3.** Two strongly concatenable processes corresponding to π of Figure 2

Such operations are matched precisely by operations and axioms of $\mathcal{P}(N)$, and this is the essence of the theorem above.

The symmetries of $\mathcal{P}(N)$ and the related axiom on the symmetry isomorphism $c$ play in this correspondence a role absolutely fundamental: they account for the families of orderings $\{<_a\}_{a\in S_N}$ and $\{\ll_a\}_{a\in S_N}$, which are the key to concatenable processes, guaranteeing a correct treatment of sequential composition. In other words, they are an algebraic representation of the *'threads of causality'* in process concatenation. On the other hand, the axiom is actually a problematic one: because of its negative premise, *viz.* $a \neq b$, it invalidates the freeness of $\mathcal{F}(N)$ on Petri. Much worse, it makes $\mathcal{P}(\_)$ act *not functorially* on Petri. A detailed study of this issue is undertaken in [**87**], where a functorial and universal construction for net computations is presented, based on a refinement of the notion of concatenable processes that is the topic of next section.

## 3. Strongly Concatenable Processes

*Strongly concatenable processes* are a slight refinement of concatenable processes introduced in [**87**] to yield a *functorial* algebraic description of net computations. The refinement, which consists of decorating initial and terminal places of processes more strongly than in concatenable processes, e.g., by ordering *all* of them (cf. Figure 3), is shown to be — in a very precise mathematical sense — the *slightest* refinement that may achieve this. As for their predecessors, strongly concatenable processes admit an axiomatisation in terms of a universal algebraic construction based on symmetric monoidal categories.

**THEOREM.** *The strongly concatenable processes of a net $N$ are the arrows of $\mathcal{Q}(N)$, obtained from the symmetric strict monoidal category freely generated from the places of $N$ and, for each transition $t$ of $N$, an arrow $t_{u,v}\colon u \to v$ for each pair of linearisations (as strings) $u$ and $v$ of the source and target (multisets) of $t$, by quotienting modulo the axiom*

$$(\Phi) \qquad s; t_{u',v} = t_{u,v'}; s', \qquad \text{for } s\colon u \to u' \text{ and } s'\colon v' \to v \text{ symmetries}.$$

The key point here is to associate to $N$ a category whose objects form a free *non-commutative* monoid (*viz.* $S_N^*$ as opposed to $\mu(S_N)$), i.e., to deal with *strings* as explicit *representatives* of multisets. As a consequence, each transition of $N$ has many corresponding arrows in $\mathcal{Q}(N)$, all however 'related' to each other by the *naturality* condition $(\Phi)$, which is the second relevant feature of $\mathcal{Q}(\_)$, actually the one that keeps the computational interpretation of the category $\mathcal{Q}(N)$ (strongly concatenable processes) so surprisingly close to that of $\mathcal{P}(N)$ (concatenable processes).

Concerning functoriality, $\mathcal{Q}(\_)$ extends to a *coreflection* functor from the category of Petri nets to *a* category of symmetric monoidal categories. Here, as in [**61**], we proceed

using *2-categories*, an high-level approach that has the advantage of hiding some of the gory details.

**DEFINITION.** A *symmetric Petri category* is a symmetric strict monoidal category $\mathsf{C}$ whose monoid of objects is $S^*$, the free monoid on $S$, for some set $S$.

Symmetric Petri categories allow us to capture the essence of the arrows generating $\mathcal{Q}(N)$, i.e., the instances of the transitions of $N$. These have in fact two very special properties that characterise them completely: (1) they are decomposable as tensors only trivially, and as compositions only by means of symmetries, and (2) they satisfy axiom $(\Phi)$. We then use such properties, expressed in abstract categorical terms, to define the notion of *transition* in a general symmetric Petri category.

**DEFINITION.** Let $\mathsf{C}$ be a symmetric Petri category and $S^*$ its monoid of objects. An arrow $\tau$ in $\mathsf{C}$ is *primitive* if (denoting by $\varepsilon$ the empty word in $S^*$)

- $\triangleright$ $\tau$ is *not* a symmetry;
- $\triangleright$ $\tau = \alpha;\beta$ implies $\alpha$ is a symmetry and $\beta$ is primitive, or vice versa;
- $\triangleright$ $\tau = \alpha \otimes \beta$ implies $\alpha = id_\varepsilon$ and $\beta$ is primitive, or vice versa.

A *transition* $\tau\colon \overline{u} \to \overline{v}$ of $\mathsf{C}$, for $\overline{u}, \overline{v} \in \mu(S)$, is a family $\{\tau_{u,v}\colon u \to v \text{ in } \mathsf{C}\}$ of *primitive* arrows indexed by those pairs of strings $u$ and $v$ with underlying multisets $\overline{u}$ and $\overline{v}$, respectively, and such that $s;\tau_{u',v} = \tau_{u,v'};s'$, for $s\colon u \to u'$ and $s'\colon v' \to v$ symmetries of $\mathsf{C}$.

The definition above — that can also be formalised stating that transitions are natural transformations between appropriate functors — captures the essence of $\mathcal{Q}(N)$: the transitions in $\mathcal{Q}(N)$ are *all* and *only* the families $\{t_{u,v} \mid t\colon \overline{u} \to \overline{v} \in T_N\}$. This leads us to the following characterisation the *category* (of the categories) *of net computations*. The 2-categorical notions used in the theorem below are natural extensions of the corresponding (1-)categorical concepts; the interested reader will find the detailed definitions in [**52**].

**THEOREM.** *Let* $\mathsf{SPetriCat}$ *be the 2-category whose objects are the symmetric Petri categories, whose arrows are the symmetric strict monoidal functors that respect transitions, and with a 2-cell $F \Rightarrow G$ if there exists a monoidal natural isomorphism between $F$ and $G$ whose components are all symmetries.*

*Then,* $\mathcal{Q}(\_)\colon \mathsf{Petri} \to \mathsf{SPetriCat}$ *is a pseudo 2-functor (considering the category* $\mathsf{Petri}$ *of Petri nets as a trivial 2-category) that admits a pseudo right adjoint $\mathcal{N}(\_)$ forming with $\mathcal{Q}(\_)$ a pseudo coreflection.*

## 4. Pre-Nets

Although strongly concatenable processes settle the token ambiguity problem of §1, they yield a construction that is functorial only up to isomorphism, thus needing a complex quotient operation [**87**] or, equivalently, the 2-categorical treatment outlined above.

In [**15, 16**] we proposed an alternative construction centred on the notion of pre-net. *Pre-nets* are nets whose states are *strings* of tokens (as opposed to *multisets*). Such states can be seen as totally ordered markings, a more concrete representation of multisets. The idea is that each transition of a pre-net must specify the precise order in which the required resources are fetched and the results are produced, as if it were an elementary strongly concatenable process.

**DEFINITION.** A *pre-net* is a tuple $R = (\zeta_0, \zeta_1\colon T_R \to S_R^*)$, where $S_R$ is a set of *places*, $T_R$ is a set of *transitions*, and $\zeta_0$ and $\zeta_1$ are functions assigning, respectively, source and target to each transition.

A pre-net can be thought of as an implementation of a net, where an abstract data structure, the multiset, is refined into a more concrete implementation data structure, the string, and where each transition $t\colon \overline{u} \to \overline{v}$ is simulated by *one* linear implementation $t_{u,v}\colon u \to v$ arbitrarily fixed for some linearisations $u$ and $v$ of $\overline{u}$ and $\overline{v}$. For each PT we can arbitrarily choose a pre-net representation. This corresponds to fix a total order for the pre- and post-set of each transition, and differs from the approach recalled in §3 where, in order to avoid a choice, *all* the possible linearisations of the pre- and post-sets are considered in the alternative presentation of the net. We shall see that, in order to capture the standard process semantics of nets, choosing one representative for each transition suffices. In particular, although abandoning multisets might appear at first unnatural, this approach enjoys some good properties. Here we limit ourselves to the following two.

- ▷ All pre-net implementations of the same net share the same semantic model, i.e., the semantics is independent of the choice of linearisations;
- ▷ The semantic model for the implemented net given by the construction $\mathcal{Q}(\_)$ can be recovered from any pre-net implementation.

We shall use PreNet to indicate the category of pre-nets with the obvious notion of morphisms, i.e., a graph morphism whose node component is a monoid homomorphism. Let $\mu_R\colon S_R^* \to \mu(S_R)$ denote the function that maps $u$ to $\overline{u}$, the multiset consisting of the symbols in $u$. Then, the map $\mathcal{A}$, from pre-nets to PT nets, sending the pre-net $R = (\zeta_0, \zeta_1\colon T_R \to S_R^*)$ to the net $\mathcal{A}(R) = (\mu_R \circ \zeta_0, \mu_R \circ \zeta_1\colon T_R \to \mu(S_R))$ extends to a functor from PreNet to Petri.

The functor $\mathcal{A}(\_)\colon \mathsf{PreNet} \to \mathsf{Petri}$ is neither full, nor faithful. However, if we consider the category Net whose objects are either PT nets or pre-nets and whose morphisms are graph morphisms with monoid homomorphism as node components, then Petri is the quotient of Net modulo commutativity of the monoidal structure of nodes. This establishes a strong relationship, between PT nets and pre-nets, expressible via a coreflection between Petri and Net, which supports and further motivates our approach.

The natural algebraic models for representing concurrent computations on pre-nets live in the category SSMC of symmetric strict monoidal categories. More precisely, we are only interested in the full subcategory consisting of categories whose monoid of objects is freely generated. We denote it by FSSMC. The obvious forgetful functor from the category FSSMC to the category PreNet admits a left adjoint $\mathcal{Z}$. The category $\mathcal{Z}(R)$ has as objects the strings of $S_R^*$, and as arrows those generated by the rules below, modulo the axioms of monoidal categories (associativity, functoriality, identities, unit), including the coherence axioms that make of $c$ the symmetry natural isomorphism.

$$\frac{w \in S_R^*}{id_w\colon w \to w \in \mathcal{Z}(R)} \qquad \frac{a \text{ and } b \text{ in } S_R^*}{c_{a,b}\colon ab \to ba \in \mathcal{Z}(R)} \qquad \frac{t\colon u \to v \text{ in } T_R}{t\colon u \to v \in \mathcal{Z}(R)}$$

$$\frac{\alpha\colon u \to v \text{ and } \beta\colon u' \to v' \in \mathcal{Z}(R)}{\alpha \otimes \beta\colon uu' \to vv' \in \mathcal{Z}(R)} \qquad \frac{\alpha\colon u \to v \text{ and } \beta\colon v \to v' \in \mathcal{Z}(R)}{\alpha;\beta\colon u \to v' \in \mathcal{Z}(R)}$$

The above construction bears strong similarities to the work on coherence by MacLane and Kelly, and even more closely to Pfender's construction of the free $S$-monoidal category [**78**]. In computer science, similar constructions are given by Hotz's $X$-categories [**44**], and by Benson [**8**], with grammars as the primary area of application.

As anticipated, corresponding to the two features of our approach, we have the following results. The first states that pre-nets representing isomorphic PT nets yield the same

algebraic net semantics. The second relates $\mathcal{Q}$, $\mathcal{Z}$, and $\mathcal{A}$, and contains the entire essence of the pre-net approach: *any pre-net representation of the net $\mathcal{A}(R)$ is as good as $R$.*

**THEOREM.** *Let $R, R' \in$ PreNet. If $\mathcal{A}(R) \simeq \mathcal{A}(R')$, then $\mathcal{Z}(R) \simeq \mathcal{Z}(R')$.*

**THEOREM.** *For $R$ a pre-net, the category $\mathcal{Z}(R)$ quotiented out by the axiom $t = s_0;t;s_1$, for each transition $t : u \to v$ and symmetries $s_0 : u \to u$ and $s_1 : v \to v$ is equivalent to the category $\mathcal{Q}(\mathcal{A}(R))$ of strongly concatenable processes.*

## 5. Related Work

An alternative important line of research on Petri nets semantics is the so-called *unfolding* approach, initiated by Nielsen et al. in [**67**] and further developed by Winskel in [**96, 98**], according to which the *'dynamic'* structure of nets is 'unrolled', 'unfolded' to the *'static'* structure of event structures or, equivalently, of so-called occurrence nets. Its main merit is to assign to each net a single object that represents its entire behaviour and explains in a uniform, appealing way the interplay between non-determinism and concurrency. This fact can be justified formally by considering that the unfolding is a special (co)limit construction that gives rise to a coreflection between the categories of (safe) Petri nets and prime event structures. An alternative unfolding construction is described in [**69**], while Engelfriet in [**32**] consider a wider class of nets. Meseguer et al. [**60**] extend the construction of [**67**] to the entire category of place/transition nets and, in [**59**], study the relationships between unfolding and process semantics.

Other semantic investigations have capitalized directly on the *algebraic structure* of Petri nets, noticed by Reisig [**80**], by Winkowski [**92, 93**], and later exploited by Winskel to identify a sensible notion of *morphism* between nets [**95, 97**] and open the way to categorical treatments. Among the algebraic/categorical approaches, a relevant place is occupied by those drawing on the analogy between nets and proofs in *liner logic*, first noticed by Asperti [**3**]. Among these, we mention [**13, 14, 31**]. A really excellent survey is given by Martí-Oliet and Meseguer in [**57**]. Other relevant approaches are by Mukund [**66**], which provides an account of net behaviours in terms of (step) transition systems, and by Hoogers et al. in [**42**], that uses (generalised) trace theory (cf. [**56, 27**]) to the same purpose, and in [**43**], where a notion of net unfolding is explained in terms of a notion of local event structure.

More recently, Ehrig and Padberg [**30, 73**], inspired by the 'Petri nets as monoid' approach, give a uniform algebraic presentation of several classes of nets based on the idea of a parameterized abstract Petri net. Desel et al. [**25**] attain results on the representation of net processes similar to those presented here using partial algebras, in a fashion not unlike the early work of Winkowski [**92, 93**].

The 'Petri nets as monoids' paradigm has been applied successfully to the semantics of several extensions of place/transition nets. Among these, two recent interesting results concern *zero-safe* and *contextual nets*. Zero-safe nets, introduced by Bruni and Montanari [**17, 18**], extend Petri nets with a simple mechanism to model transactions, i.e., two or more transitions that must always occur without any other transition occurring in between. Contextual nets [**22, 65, 45**] (see also [**21, 91, 6, 5**]) are nets with 'read-arcs' used to 'read' without consuming, so allowing multiple, non-exclusive, concurrent uses of the same resource (token) and, therefore, the modeling of shared resources. Bruni and Sassone in [**19**] extend the categorical process semantics approach surveyed here satisfactorily to contextual nets, building on previous work by Gadducci and Montanari [**33**].

## PETRI NETS IN THE LARGE

The previous sections have mainly focused our attention *'in the small'*, at level of single nets, whereas Petri nets are often used *'in the large'*, for instance as a semantic basis to interpret concurrent languages, which calls for the study of *algebras of nets* 'in the large' and, possibly, for their abstract characterisations. Among several existing approaches, we recall the fundamental ideas underlying the work presented in [**68**], focusing on *finite* nets whose transitions are *labelled* by (possibly silent) actions. We shall use a *countable* set *Act* of *visible* actions $\alpha_1, \alpha_2, \alpha_3, \ldots$, and a distinguished *silent* action $\tau$.

**DEFINITION.** A labelled *Petri net* is a Petri net $N$ together with an *initial state* $s_N \in \mu(S_N)$, and a *labelling* function $\lambda_N \colon T_N \to Act \cup \{\tau\}$.

### 6. An Algebra of Nets

Similarly to [**10, 63**], everything is based on a notion of *interface* for Petri nets. These are ordered selections of places, the 'input', and transitions, the 'output', that specify what parts of $N$ are *public*, i.e., accessible from the environment, and what parts are *private* to the net. The private places and transitions cannot be accessed and, therefore, cannot be used directly for connecting nets to each other.

**DEFINITION.** A *net with interface* is a structure $p_1, \ldots, p_n; t_1, \ldots, t_m \triangleright N$, where $N$ is a finite labelled net, and $p_1, \ldots, p_n \in S_N$, $t_1, \ldots, t_m \in T_N$ are all distinct, and $\lambda_N(t_i) \neq \tau$.

Drawing on the experience of developments in concurrency theory, a minimal yet expressive, set of combinators should certainly include operations allowing (forms of) *interaction/communication*, *parallel composition*, *recursion*, and — to facilitate the description of modular systems — operations such as *relabelling* and *hiding*. However, in order to avoid a chaotic 'structural' calculus where everything is permitted, it is obvious that some restrictions on the allowed connections of places and transitions must be imposed. The input/output partition of interfaces readily suggest a reasonable discipline of interaction: connections between nets should go from outputs to inputs, involving *only* public components. This formalises the well-motivated and solid intuition that the only allowed interactions are achieved by *sending* and *receiving* along interfaces, thought of as communication channels, the input interfaces providing 'buffers' in which the tokens arriving from the environment are gathered, the output interfaces sending tokens out to the environment. In other words, interfaces provide the notions of 'private' and 'public' channels for nets, and their input/output partition suggests a discipline for net cooperation.

**DEFINITION.** The set **CM** of combinators of nets with interface consists of the combinators defined by the following rules.

- $\triangleright$ $\dfrac{\vec{p}_0; \vec{t}_0 \triangleright N_0 \quad \text{and} \quad \vec{p}_1; \vec{t}_1 \triangleright N_1 \quad \text{disjoint}}{par(\vec{p}_0; \vec{t}_0 \triangleright N_0, \vec{p}_1; \vec{t}_1 \triangleright N_1) = \vec{p}_0, \vec{p}_1; \vec{t}_0, \vec{t}_1 \triangleright N_0 \| N_1}$

  where $N_0 \| N_1$ is the (componentwise) union of $N_0$ and $N_1$;

- $\triangleright$ $\dfrac{1 \leq i \leq |\vec{p}| \quad \text{and} \quad 1 \leq j \leq |\vec{t}|}{add(i, j, \vec{p}; \vec{t} \triangleright N) = \vec{p}; \vec{t} \triangleright N\langle p_i \hookleftarrow t_j \rangle}$

  where $N\langle p \hookleftarrow t \rangle$ is the net $N$ augmented with an arc from $t$ to $p$;

- $\triangleright$ $rel(\phi, \vec{p}; \vec{t} \triangleright N) = \vec{p}; \vec{t} \triangleright N[\phi]$,

  where $\phi \colon Act \to Act \cup \{\tau\}$ is a 'relabelling' function, and $N[\phi]$ is obtained from $N$ by relabelling via $\phi$ the transitions that carry *non*-$\tau$ actions;

$$\triangleright \quad \frac{\max(P) \leq |\vec{p}| \quad \text{and} \quad \max(T) \leq |\vec{t}|}{hide(P,T,\vec{p};\vec{t} \triangleright N) = \vec{p} \setminus P; \vec{t} \setminus T \triangleright N}$$

where $P$ and $T$ are finite sets of positive natural numbers ($\max(\varnothing) = 0$), and $\vec{x} \setminus X$ is the string obtained from $\vec{x}$ by removing $x_i$, for all $i \in X$;

$$\triangleright \quad \frac{1 \leq i \leq |\vec{p}|}{mark(i,\vec{p};\vec{t} \triangleright N) = \vec{p};\vec{t} \triangleright N \bullet p_i}$$

where $N \bullet p$ is the net $N$ augmented with a token in $p$.

Observe that, since the $par(\_,\_)$ combinator is defined explicitly only for disjoint nets, a 'renaming' is generally needed before applying it to its arguments. This implies that no 'fusion' of nets is allowed by **CM**. Combinator $add(i,j,\_)$ adds an arc from the $i$th place to the $j$th transitions of the interface. It provides both a form of recursion and, used in connection with $par(\_,\_)$, a form of 'asynchronous message passing' which feeds the inputs of a net with the outputs of another one.

## 7. Congruences and Contexts for Labelled Petri Nets

The semantic equivalence of concurrent systems can be described in terms of several kinds of models, e.g., languages, traces, pomsets, event structures, etc., which reflect different assumptions about how behaviour is to be observed. Each of these notions of 'observation' gives rise to standard equivalences: a *linear* equivalence, a *bisimulation* and possibly, fixed a set of operators, their congruence closures. A thorough study of sixteen such behavioural equivalences for nets with interfaces is exposed in [**68**]. Here we treat a single, yet typical, case: the step bisimulation.

**DEFINITION.** A *step bisimulation* of $N_0$ and $N_1$ is a relation $\mathcal{R} \subseteq \mu(S_{N_0}) \times \mu(S_{N_1})$ such that $s_{N_0}\ \mathcal{R}\ s_{N_1}$, and whenever $s\ \mathcal{R}\ \overline{s}$, then (1) for each step (fireable multiset of transitions) $s[X\rangle s'$ of $N_0$, there exists a sequence of steps $\overline{s}[Y_1 \cdots Y_n\rangle \overline{s'}$ of $N_1$ resulting in the same *multiset* of *non*-$\tau$ labels as $X$, and with $s'\ \mathcal{R}\ \overline{s'}$; (2) vice versa swapping the roles of $N_0$ and $N_1$. Nets $\vec{p}_0;\vec{t}_0 \triangleright N_0$ and $\vec{p}_1;\vec{t}_1 \triangleright N_1$ are *step bisimilar*, written $\vec{p}_0;\vec{t}_0 \triangleright N_0 \leftrightarrows \vec{p}_1;\vec{t}_1 \triangleright N_1$, if there exists a step bisimulation of $N_0$ and $N_1$.

For engineering reasons, related to feasibility of correctness *verification* for complex systems, for mathematical reasons, related to the simplicity of *equational reasoning*, and for conceptual reasons, related to common intuitions about system equivalence, it is important to consider equivalences which are *congruences* for a chosen set of system constructors. This guarantees that systems can be replaced by equivalent ones in any context. Since it easy to see that $\leftrightarrows$ is not a congruence for $add(i,j,\_)$, we are led to $\leftrightarrows^c$, the largest congruence contained in it, *viz.* $\vec{p}_0;\vec{t}_0 \triangleright N_0 \leftrightarrows^c \vec{p}_1;\vec{t}_1 \triangleright N_1$ if and only if, for each **CM**-context $\mathcal{C}$, either *both* nets are incompatible with it, or $\mathcal{C}[\vec{p}_0;\vec{t}_0 \triangleright N_0] \leftrightarrows \mathcal{C}[\vec{p}_1;\vec{t}_1 \triangleright N_1]$.

This universal quantification over *all* contexts has however obvious drawbacks. The main result of [**68**] is to show that it can actually be dispensed with by identifying a minimal set of context which is *universal* for it. More precisely, for each pair $N_0$ and $N_1$ of nets with interface there exists a readily-identified context $\mathcal{C}$ such that $N_0$ and $N_1$ are $\leftrightarrows$-congruent if and only if $\mathcal{C}$ does not $\leftrightarrows$-distinguish them. Here follow some of the details.

Recalling that *Act* is equipped with an enumeration $\alpha_1, \alpha_2, \ldots$, let $\psi$ be the relabelling function $\alpha_i \mapsto \alpha_{3i}$, $i \in \omega$.

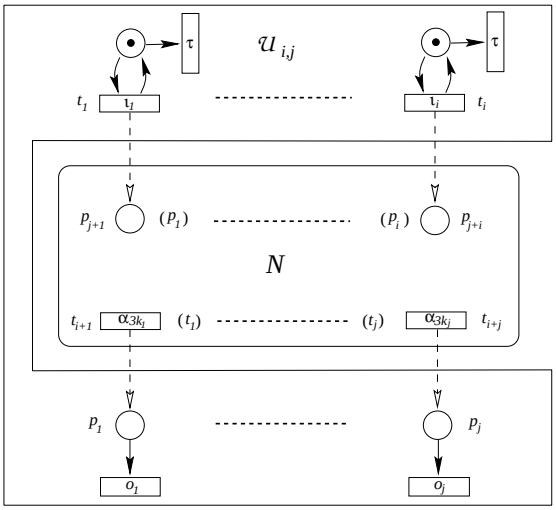


**FIGURE 4**

**DEFINITION.** Let $\varnothing;t \triangleright U$ and $p;\varnothing \triangleright U'$ be the nets with interface shown below.


Let $\mathcal{C}_{i,j}$ and $\mathcal{U}_{i,j}$, $i,j \in \omega$, be the contexts defined below (with self-explanatory shorthands)

$$\mathcal{C}_{i,j} = par\Big(par^i_{k=1}\big(\varnothing;t \triangleright U[\alpha_{3k-2}/\alpha]\big), par^j_{k=1}\big(p;\varnothing \triangleright U'[\alpha_{3k-1}/\alpha]\big)\Big),$$
$$\mathcal{U}_{i,j} = add^j_{k=1}\Big(k, i+k, add^i_{k=1}\Big(j+k, k, par\big(\mathcal{C}_{i,j}, rel(\psi,\_)\big)\Big)\Big).$$

Figure 4 presents $\mathcal{U}_{i,j}[\vec{p};\vec{t} \triangleright N]$ for a $\vec{p};\vec{t} \triangleright N$ with $|\vec{p}| = i$ and $|\vec{t}| = j$. The interface of $\mathcal{U}_{i,j}[\vec{p};\vec{t} \triangleright N]$ is shown by naming and numbering the places and transitions which belong to it. The information in parentheses concern the orderings in $\vec{p};\vec{t}$. Concerning the labels, we use $\iota_k$ for $\alpha_{3k-2}$, $k = 1,\ldots,i$, $o_k$ for $\alpha_{3k-1}$, $k = 1,\ldots,j$, and $\alpha_{k_1},\ldots\alpha_{k_j}$ for the labels of $\vec{t}$ in $N$. The dashed arrows are those inserted by *add*.

The contexts $\mathcal{U}_{i,j}$ are conceptually very simple. They provide a copy of $\varnothing;t \triangleright U$ for each place in $\vec{p}$, and a copy of $p;\varnothing \triangleright U'$ for each transition in $\vec{t}$. The cascade of $add(i,j,\_)$ connects together the transition-place pairs so created. The role of the collection of $\varnothing;t \triangleright U$ is to test the 'reactivity' of the 'input' sites of $\vec{p};\vec{t} \triangleright N$ by sending in any number of tokens, at any relative speed and independently for each place in $\vec{p}$. The collection of $p;\varnothing \triangleright U'$ tests the 'output'-behaviour by recording the firings of the transitions in $\vec{t}$.

In order for these contexts to form universal collections, it is necessary to distinguish in the behaviour of $\mathcal{U}_{i,j}[\vec{p};\vec{t} \triangleright N]$ the actions stemming from $\mathcal{U}_{i,j}$ from those stemming from $N$. This is achieved by the $rel(\psi,\_)$ combinator: since the actions of $N$ are uniformly 'remapped' to $3k$-numbered actions, we are free to use differently numbered actions in the contexts. The soundness of this technique relies on the fact that $\psi$ is injective and, therefore, no equivalences are enforced by the $\psi$-relabelling. We thus conclude as follows.

**THEOREM.** *For $\vec{p}_0;\vec{t}_0 \triangleright N_0$ and $\vec{p}_1;\vec{t}_1 \triangleright N_1$ nets with interface,*

$$\vec{p}_0;\vec{t}_0 \triangleright N_0 \leftrightarrows^c \vec{p}_1;\vec{t}_1 \triangleright N_1 \iff |\vec{p}_0| = |\vec{p}_1| = i,\ |\vec{t}_0| = |\vec{t}_1| = j,\ \textit{and}\ \mathcal{U}_{i,j}[\vec{p}_0;\vec{t}_0 \triangleright N_0] \leftrightarrows \mathcal{U}_{i,j}[\vec{p}_1;\vec{t}_1 \triangleright N_1].$$

## 8. Related Work

The work outlined in the second part of this survey relates to several Petri net calculi proposed in the literature. Among these, we mention Gorrieri and Montanari's SCONE [**36**], defined around operations of prefixing, parallel, and non-deterministic composition, and used to give semantics to a fragment of CCS. Differently from most other calculi, SCONE is not based on an explicit notion of interface by means of which nets are composed. It aims at describing behavioural more than structural composition, and is essentially a big net whose markings represent concurrent processes behaviours, in the same sense as CCS can be seen as a big transition system whose states represent processes.

The Petri Box calculus [**10**], by Best et al., has received much attention in the literature. It is inspired by CCS and motivated by the need to simplify the task of giving compositional denotational semantics to concurrent programming languages. The calculus has a very rich collection of operations, including sequential, non-deterministic, and asynchronous parallel composition with explicit multiple synchronisation based on a notion of interface constituted by designated entry and exit places. The Box calculus has been used to describe distributed algorithms, to give semantics to concurrent programming languages [**11**], and has been embedded in the PEP computer-aided tool [**37**].

A highly elegant calculus is Milner's calculus of named nets [**63**], arisen in the context of control structures and, as such, inspired by name passing calculi, as the $\pi$-calculus. It focuses on very few basic 'controls' by means of which places, tokens, and transitions can be glued together to form any finite net. A notable difference with the work surveyed here is that Milner's calculus is definitely more structure-oriented. The controls are in fact suggested by structural considerations rather than by behavioural intuitions such as asynchronous message passing underlying **CM**. The dynamics of named nets is demanded to an elegant reduction relation, and the question of behavioural congruences is open.

## 9. Conclusions and Future Work

Algebraic structures based on a central operation of *iteration*, or *feedback* — inspired by flowcharts and program schemata — have appeared rather early in computer science, see, e.g., [**29, 4, 89, 90, 63**] and [**12**], that offers for a thorough exposition of so-called '*iteration theory*' and more references. The advent of *traced monoidal categories* [**50**], i.e., monoidal categories equipped with a *feedback* operation completely analogous to the one considered in **CM** has recently revived interest in using such abstract structures in semantics of computation, as e.g., in [**1, 53, 40, 41**]. Obviously, the calculus of [**68**] fits nets into this framework very nicely, although some of the details still need to be clarified. In particular, it may still lack some important operations, most notably synchronisation.

Finally, as already mentioned, it would be interesting to know how well and how uniformly can the 'Petri nets as monoids' approach be lifted to high level nets.

Acknowledgements. I would like to thank the colleagues who have been part of developing the material behind this survey. In particular, I acknowledge close collaboration with R. Bruni, J. Meseguer, U. Montanari, M. Nielsen, and L. Priese on many parts of this work.

## References


[1] S. ABRAMSKY (1996), Retracing Some Paths in Process Algebra, in *Proceedings of CONCUR 96*, U. Montanari and V. Sassone (Eds.), *Lecture Notes in Computer Science* 1119, 1–17, Springer-Verlag.

[2] S. AJMONE MARSAN, A. BOBBIO, AND S. DONATELLI (1998), Petri Nets in Performance Analysis: An introduction, in *Advances in Petri Nets, Lectures on Petri Nets I: Basic Models*, W. Reisig and G. Rozenberg (Eds.), *Lecture Notes in Computer Science* 1491, 122–173, Springer-Verlag.

[3] A. ASPERTI, G. FERRARI, AND R. GORRIERI (1990), Implicative Formulae in the "Proofs as Computations" Analogy, in *Proceedings of POPL 91*, 59–71, ACM Press.

[4] E.S. BAINBRIDGE (1976), Feedback and Generalized Logic. *Information and Control* 31, 75–96, Academic Press.

[5] P. BALDAN (2000), *Modelling Concurrent Computations: From Contextual Petri Nets to Graph Grammars*. Ph.D. thesis, TD-1/00, Dipartimento di Informatica, Università di Pisa.

[6] P. BALDAN, A. CORRADINI, AND U. MONTANARI (1998), An event structure semantics for P/T contextual nets: Asymmetric event structures, in *Proceedings FoSSaCS'98*, M. Nivat (Ed.), *Lecture Notes in Computer Science* 1378, 63–80, Springer-Verlag.

[7] J. BÉNABOU (1963), Categories with Multiplication. *Comptes Rendus Académie Science Paris* 256, 1887–1890.

[8] D.B. BENSON (1975) , The Basic Algebraic Structures in Categories of Derivations. *Information and Control* 28(1), 1–29, Academic Press.

[9] E. BEST AND R. DEVILLERS (1987), Sequential and Concurrent Behaviour in Petri Net Theory. *Theoretical Computer Science* 55, 87–136, Elsevier.

[10] E. BEST, R. DEVILLERS, AND J. HALL (1992), The Petri Box Calculus: a New Causal Algebra with Multilabel Communication, in *Advances in Petri Nets 92*, G. Rozenberg (Ed.), *Lecture Notes in Computer Science* 609, 21–69, Springer-Verlag.

[11] E. BEST, R. DEVILLERS, AND M. KOUTNY (1998), Petri Nets, Process Algebras, and Concurrent Programming Languages, in *Advances in Petri Nets, Lectures on Petri Nets II: Applications*, W. Reisig and G. Rozenberg (Eds.), *Lecture Notes in Computer Science* 1492, 1–84, Springer-Verlag.

[12] S.L. BLOOM AND Z. ÉSIK (1991), *Iteration Theories: the Equational Logic of Iterative Processes*. EATCS Monographs on Theoretical Computer Science 30, Springer-Verlag.

[13] C. BROWN AND D. GURR (1992), Temporal Logic and Categories of Petri Nets, in *Proceedings of ICALP 93*, A. Lingas *et al.* (Eds.), *Lecture Notes in Computer Science* 700, 570–581, Springer-Verlag.

[14] C. BROWN, D. GURR, AND V. DE PAIVA (1991), *A Linear Specification Language for Petri Nets*. Technical Report DAIMI PB-363, Computer Science Dept., University of Aarhus.

[15] R. BRUNI, J. MESEGUER, U. MONTANARI, AND V. SASSONE (1999) , Functorial Semantics for Petri Nets under the Individual Token Philosophy, in *Proceedings of CTCS 99*, M. Hofmann, G. Rosolini, and D. Pavlovic (Eds.), *Electronic Notes in Theoretical Computer Science* 29, Elsevier.

[16] R. BRUNI, J. MESEGUER, U. MONTANARI, AND V. SASSONE (2000), Functorial Models for Petri nets. *Information and Computation*, Academi Press. To appear.

[17] R. BRUNI AND U. MONTANARI (2000), Zero-Safe Nets: Comparing the Collective and Individual Token Approaches. *Information and Computation* 156 46–89, Academi Press.

[18] R. BRUNI AND U. MONTANARI (2000), Executing Transactions in Zero-safe Nets, in *Proceedings of ICATPN 2000*, D. Simpson and M. Nielsen (Eds.), *Lecture Notes in Computer Science* 1825, 83–102, Springer-Verlag.

[19] R. BRUNI AND V. SASSONE (2000), Algebraic Models for Contextual Nets, in *Proceedings of ICALP 2000*, U. Montanari *et al.* (Eds.), *Lecture Notes in Computer Science* 1853, 175–186, Springer-Verlag.

[20] N. BUSI AND R. GORRIERI (1995), A Petri Net Semantics for the Pi-Calculus, in *Proceedings of CONCUR 95*, I. Lee and S. Smolka (Eds.), *Lecture Notes in Computer Science* 962, 145–159, Springer-Verlag.

[21] N. BUSI AND M. PINNA (1996), Non Sequential Semantics for Contextual P/T Nets, in *Proceedings of ICATPN 96*, J. Billington and W. Reisig (Eds.), *Lecture Notes in Computure Science* 1091, 113–132, Springer-Verlag.

[22] S. CHRISTENSEN AND N.D. HANSEN (1993), Coloured Petri Nets extended with Place Capacities, Test Arcs and Inhibitor Arcs, in *Proceedings of ICATPN 93*, S. Ajmone Marsan (Ed.), *Lecture Notes in Computer Science* 691, 186–205, Springer-Verlag.

[23] P. DEGANO, R. DE NICOLA, AND U. MONTANARI (1988), A Distributed Operational Semantics for CCS based on Condition/Event Systems. *Acta Informatica* 26, 59–91, Springer-Verlag.

[24] P. DEGANO, J. MESEGUER, AND U. MONTANARI (1996), Axiomatizing the Algebra of Net Computations and Processes. *Acta Informatica* 33, 641–667, Springer-Verlag.

[25] J. DESEL, G. JUHÁS, AND R. LORENZ (2000), Process Semantics of Petri Nets over Partial Algebra, in *Proceedings of ICATPN 2000*, D. Simpson and M. Nielsen (Eds.), *Lecture Notes in Computer Science* 1825, 146–165, Springer-Verlag.

[26] J. DESEL AND W. REISIG (1998), Place/Transition Petri Nets, in *Advances in Petri Nets, Lectures on Petri Nets I: Basic Models*, W. Reisig and G. Rozenberg (Eds.), *Lecture Notes in Computer Science* 1491, 122–173, Springer-Verlag.

[27] V. DIEKERT AND G. ROZENBERG (EDS.) (1995), *The Book of Traces*, World Scientific.

[28] S. EILENBERG, AND G.M. KELLY (1966), Closed Categories, in *Proceedings of the Conference on Categorical Algebra*, S. Eilenberg *et. al.* (Eds.), 421–562, Springer-Verlag.

[29] C. ELGOT (1975), Monadic Computation and Iterative Algebraic Theories, in *Logic Colloquium '73*, H.E. Rose and J.C. Shepherdson (Eds.), 175–230, North-Holland.

[30] H. EHRIG AND J. PADBERG (1997), A Uniform Approach to Petri Nets, in *Proceedings of FCT 97*, C. Freksa *et al.* (Eds.), *Lecture Notes in Computer Science* 1337, 219–231, Springer-Verlag.

[31] U. ENGBERG AND G. WINSKEL (1993), Completeness Results for Linear Logic on Petri Nets, in *Proceedings of MFCS 93*, A. Borzyszkowski and S. Sokołowski (Eds.), *Lecture Notes in Computer Science* 711, 442–452, Springer-Verlag.

[32] J. ENGELFRIET (1991), Branching Processes of Petri Nets. *Acta Informatica* 28, 575–591, Springer-Verlag.

[33] F. GADDUCCI AND U. MONTANARI (1998), Axioms for contextual net processes, *Proceedings of ICALP 98*, K. Larsen *et al.* (Eds.), *Lecture Notes in Computer Science* 1443, 296–308, Springer-Verlag.

[34] R.J. VAN GLABBEEK AND F. VAANDRAGER (1987), Petri Net Models for Algebraic Theories of Concurrency, in *Proceedings of PARLE 87*, J.W. de Bakker *et al.* (Eds.), *Lecture Notes in Computer Science* 259, 224–242, Springer-Verlag.

[35] U. GOLTZ AND W. REISIG (1983), The Non-Sequential Behaviour of Petri Nets. *Information and Computation* 57, 125–147, Academic Press.

[36] R. GORRIERI AND U. MONTANARI (1990), Scone: A Simple Calculus of Nets, in *Proceedings of CONCUR 90*, J.C.M. Baeten and J.W. Klop (Eds.), *Lecture Notes in Computer Science* 458, 2–31, Springer-Verlag.

[37] B. GRAHLMANN (1998), The State of PEP, in *Proceedings of AMAST 98*, A.M. Haeberer (Ed.), *Lecture Notes in Computer Science* 1548, 522–526, Springer-Verlag.

[38] M. HENNESSY (1988), *Algebraic Theory of Processes*. MIT Press.

[39] C.A.R. HOARE (1985), *Communicating Sequential Processes*. Prentice Hall.

[40] M. HASEGAWA (1997), Recursion from Cyclic Sharing: Traced Monoidal Categories, in *Proceedings of TLCA 97*, Ph. de Groote and J.R. Hindley (Eds.), *Lecture Notes in Computer Science* 1210, 196–213, Springer-Verlag.

[41] T.T. HILDEBRANDT, P. PANANGADEN, AND G. WINSKEL (1998), Relational Semantics of Non-Deterministic Dataflow, in *Proceedings of CONCUR 98*, D. Sangiorgi and R. de Simone (Eds.), *Lecture Notes in Computer Science* 1466, 613–628, Springer-Verlag.

[42] P.W. HOOGERS, H.C.M. KLEIJN, AND P.S. THIAGARAJAN (1992), A Trace Semantics for Petri Nets, in *Proceedings of ICALP 92*, W. Kuich (Ed.), *Lecture Notes in Computer Science* 623, 595–604, Springer-Verlag.

[43] P.W. HOOGERS, H.C.M. KLEIJN, AND P.S. THIAGARAJAN (1993), Local Event Structures and Petri Nets, in *Proceedings of CONCUR 93*, E. Best (Ed.), *Lecture Notes in Computer Science* 715, 462–476, Springer-Verlag.

[44] G. HOTZ (1965) , Eine Algebraisierung des Syntheseproblemen von Schaltkreisen, I and II. *Journal of Information Processing and Cybernetics, EIK* 1, 185–206, 209–231. Otto-von-Guericke-Universität, Magdeburg, Germany.

[45] R. JANICKI AND M. KOUTNY (1995), Semantics of Inhibitor Nets. *Information and Computation.* 123, 1–16, Academic Press.

[46] K. JENSEN (1998), An Introduction to the Practical Uses of Coloured Petri Nets, in *Advances in Petri Nets, Lectures on Petri Nets II: Applications*, W. Reisig and G. Rozenberg (Eds.), *Lecture Notes in Computer Science* 1492, 237–292, Springer-Verlag.

[47] K. JENSEN (1992), *Coloured Petri Nets – Basic Concepts, Analysis Methods and Practical Use, Vol. 1: Basic Concepts.* EATCS Monographs in Theoretical Computer Science, Springer-Verlag.

[48] K. JENSEN (1994), *Coloured Petri Nets – Basic Concepts, Analysis Methods and Practical Use, Vol. 2: Analysis Methods*. EATCS Monographs in Theoretical Computer Science, Springer-Verlag.

[49] K. JENSEN (1997), *Coloured Petri Nets – Basic Concepts, Analysis Methods and Practical Use, Vol. 3: Practical Use*. EATCS Monographs in Theoretical Computer Science — Berlin: Springer-Verlag.

[50] A. JOYAL, R. STREET, AND D. VERITY (1996), Traced Monoidal Categories. *Mathematical Proceedings of the Cambridge Philosophical Society* 119, 447–468, Cambridge University Press.
[51] G.M. KELLY (1964), On MacLane's Conditions for Coherence of Natural Associativities, Commutativities, etc. *Journal of Algebra*, n. 1, 397–402, Acamedic Press.
[52] G.M. KELLY, AND R. STREET (1974), Review of the Elements of 2-Categories, in *Category Seminar Sidney*, *Lecture Notes in Mathematics* 420, 75–103, Springer-Verlag.
[53] P. KATIS, N. SABADINI, AND R. WALTERS (1997), Bicategories of Processes. *Journal of Pure and Applied Algebra* 115, 141–178, North-Holland.
[54] S. MACLANE (1963), Natural Associativity and Commutativity. *Rice University Studies* 49, 28–46, Rice University Press.
[55] S. MACLANE (1971), *Categories for the Working Mathematician*. Springer-Verlag.
[56] A. MAZURKIEWICZ (1987), Trace theory, in *Petri Nets, Applications and Relationship to other Models of Concurrency*, W. Brauer *et al.* (Eds.), *Lecture Notes in Computer Science* 255, 279–324, Springer-Verlag.
[57] N. MARTÍ-OLIET AND J. MESEGUER (1991), From Petri Nets to Linear Logic through Categories: A Survey. *International Journal of Foundations of Computer Science* 2, 297–399, World Scientific.
[58] J. MESEGUER AND U. MONTANARI (1990), Petri Nets are Monoids. *Information and Computation* 88, 105–155, Academic Press.
[59] J. MESEGUER, U. MONTANARI, AND V. SASSONE (1996), Process versus Unfolding Semantics for Place/Transition Petri Nets. *Theoretical Computer Science* 153, 171–210, Elsevier.
[60] J. MESEGUER, U. MONTANARI, AND V. SASSONE (1997), On the Semantics of Place/Transition Petri Nets. *Mathematical Structures in Computer Science* 7, 359–397, Cambridge University Press.
[61] J. MESEGUER, U. MONTANARI, AND V. SASSONE (1997), Representation Theorems for Petri Nets, in *Foundations of Computer Science*, C. Freksa *et al.* (Eds.), *Lecture Notes in Computer Science* 1337, 239–249, Springer-Verlag.
[62] R. MILNER (1989), *Communication and Concurrency*. Prentice-Hall.
[63] R. MILNER (1993), Action Calculi or Syntactic Action Structures, in *Proceedings of MFCS 93*, A. Borzyszkowski and S. Sokołowski (Eds.), *Lecture Notes in Computer Science* 711, 105–121, Springer-Verlag.
[64] R. MILNER (1999), *Communicating and Mobile Systems: the π-Calculus*. Cambridge University Press.
[65] U. MONTANARI AND F. ROSSI (1995), Contextual Nets. *Acta Informatica* 32, 545–596, Springer-Verlag.
[66] M. MUKUND (1992), Petri Nets and Step Transition Systems. *International Journal of Foundations of Computer Science* 3, 443- 478, World Scientific.
[67] M. NIELSEN, G. PLOTKIN, AND G. WINSKEL (1981), Petri Nets, Event Structures and Domains, Part 1. *Theoretical Computer Science* 13, 85–108, Elsevier.
[68] M. NIELSEN, L. PRIESE, AND V. SASSONE (1995), Characterizing Behavioural Congruences for Petri Nets, in *Proceedings of CONCUR 95*, I. Lee and S. Smolka (Eds.), *Lecture Notes in Computer Science* 962, 175–189, Springer-Verlag.
[69] M. NIELSEN, G. ROZENBERG, AND P.S. THIAGARAJAN (1995), Transition Systems, Event Structures and Unfoldings. *Information and Computation* 118, 191–207, Academic Press.
[70] M. NIELSEN AND V. SASSONE (1998), Petri Nets and Other Models of Concurrency, in *Advances in Petri Nets*, *Lectures on Petri Nets I: Basic Models*, W. Reisig and G. Rozenberg (Eds.), *Lecture Notes in Computer Science* 1491, 587–642, Springer-Verlag.
[71] E.R. OLDEROG (1987), A Petri Net Semantics for CCSP, in *Advances in Petri Nets 86*, G. Rozenberg (Ed.), *Lecture Notes in Computer Science* 266, 196–223, Springer-Verlag.
[72] E.R. OLDEROG (1991), *Nets, Terms and Formulas*. Cambridge Tracts in Theoretical Computer Science, Cambridge University Press.
[73] J. PADBERG (1999), Abstract Petri nets as a Uniform Approach to High-Level Petri Nets, in *Proceedings of WADT 98*, C. Freksa *et al.* (Eds.), *Lecture Notes in Computer Science* 1589, 241–260, Springer-Verlag.
[74] C.A. PETRI (1962), *Kommunikation mit Automaten*. Ph.D. thesis, Institut für Instrumentelle Mathematik, Bonn.
[75] C.A. PETRI (1973), Concepts of Net Theory, in *Proceedings of MFCS 73*, 137–146, Mathematics Institute of the Slovak Academy of Science.
[76] C.A. PETRI (1977), *Non-Sequential Processes*. Interner Bericht ISF–77–5, Gesellschaft für Mathematik und Datenverarbeitung, Bonn.
[77] *Petri Net Tools on the Web*, web page. `http://www.daimi.au.dk/ petrinet/tools/`, DAIMI, University of Aarhus.
[78] M. PFENDER (1974) , *Universal Algebra in S-Monoidal Categories*, Algebra-Berichte 20, Department of Mathematics, University of Munich.

[79] V. PRATT (1986), Modelling Concurrency with Partial Orders. *International Journal of Parallel Programming* 15, 33–71, Plenum.
[80] W. REISIG (1985), *Petri Nets (an Introduction)*. EATCS Monographs on Theoretical Computer Science 4, Springer-Verlag.
[81] W. REISIG (1991), *System Design Using Petri Nets*. Springer-Verlag.
[82] W. REISIG (1998), *Elements of Distributed Algorithms: Modeling and Analysis with Petri Nets*. EATCS Monographs on Theoretical Computer Science, Springer-Verlag.
[83] W. REISIG AND G. ROZENBERG (1998), Informal Introduction to Petri Nets, in *Advances in Petri Nets, Lectures on Petri Nets I: Basic Models*, W. Reisig and G. Rozenberg (Eds.), *Lecture Notes in Computer Science* 1491, 1–11, Springer-Verlag.
[84] G. ROZENBERG AND J. ENGELFRIET (1998), Elementary Net Systems, in *Advances in Petri Nets, Lectures on Petri Nets I: Basic Models*, W. Reisig and G. Rozenberg (Eds.), *Lecture Notes in Computer Science* 1491, 12–121, Springer-Verlag.
[85] G. ROZENBERG AND P.S. THIAGARAJAN (1986), Petri Nets: Basic Notions, Structure, Behaviour, in *Current Trends in Concurrency*, J. de Bakker *et al.* (Eds.), *Lecture Notes in Computer Science* 224, 585–668, Springer-Verlag.
[86] V. SASSONE (1996), An Axiomatization of the Algebra of Petri Net Concatenable Processes. *Theoretical Computer Science* 170, 277–296, Elsevier.
[87] V. SASSONE (1998), An Axiomatization of the Category of Petri Net Computations. *Mathematical Structures in Computer Science* 8, 117–151, Cambridge University Press.
[88] E. SMITH (1998), Principles of High-Level Net Theory, in *Advances in Petri Nets, Lectures on Petri Nets I: Basic Models*, W. Reisig and G. Rozenberg (Eds.), *Lecture Notes in Computer Science* 1491, 174–210, Springer-Verlag.
[89] G. ŞTEFANĔSCU (1987), On Flowchart Theories: Part I. The Deterministic Case. *Journal of Computer and System Sciences* 35, 163–191, Academic Press.
[90] G. ŞTEFANĔSCU (1987), On Flowchart Theories: Part II. The Nondeterministic Case. *Theoretical Computer Science* 52, 307–340, Elsevier.
[91] W. VOGLER (1997), Partial order semantics and read arcs, in *Proceedings of MFCS'97*, P. Degano *et al.* (Eds.), *Lecture Notes in Computer Science* 1295, 508–517, Springer-Verlag.
[92] J. WINKOWSKI (1980), Behaviours of Concurrent Systems. *Theoretical Computer Science* 12, 39- 60, Elsevier.
[93] J. WINKOWSKI (1982), An Algebraic Description of System Behaviours. *Theoretical Computer Science* 21, 315- 340, Elsevier.
[94] G. WINSKEL (1982), Event Structure Semantics for CCS and related languages, in *Proceedings of ICALP 82*, M. Nielsen and E.M. Schmidt (Eds.), *Lecture Notes in Computer Science* 140, 561–576, Springer-Verlag.
[95] G. WINSKEL (1984), A New Definition of Morphism on Petri Nets, in *Proceedings of STACS 84*, M. Fontet and K. Mehlhorn (Eds.), *Lecture Notes in Computer Science* 166, 140–150, Springer-Verlag.
[96] G. WINSKEL (1986), Event Structures, in *Advances in Petri Nets 86*, W. Brauer *et al.* (Eds.), *Lecture Notes in Computer Science* 255, 325–392, Springer-Verlag.
[97] G. WINSKEL (1987), Petri Nets, Algebras, Morphisms and Compositionality. *Information and Computation* 72, 197- 238, Academic Press.
[98] G. WINSKEL (1988), An Introduction to Event Structures, in *Linear time, branching time, and partial order in logics and models for concurrency*, J.W. de Bakker *et al.* (Eds.), *Lecture Notes in Computer Science* 354, 365–397, Springer-Verlag.
[99] A. YAKOLEV AND A. KOELMANS (1998), Petri Nets and Digital Hardware Design, in *Advances in Petri Nets, Lectures on Petri Nets II: Applications*, W. Reisig and G. Rozenberg (Eds.), *Lecture Notes in Computer Science* 1492, 154–236, Springer-Verlag.