Electronic Notes in Theoretical Computer Science 29 (1999)
URL: http://www.elsevier.nl/locate/entcs/volume29.html 18 pages

Functorial Semantics for Petri Nets under the Individual Token Philosophy [1]

R. Bruni [a], J. Meseguer [b], U. Montanari [a] and V. Sassone [c]

[a] *Dipartimento di Informatica, Università di Pisa, I-56125 Pisa, Italia,* {bruni,ugo}@di.unipi.it

[b] *Computer Science Laboratory, SRI International, Menlo Park, CA 94025, USA,* meseguer@csl.sri.com

[c] *Dipartimento di Matematica, Università di Catania, I-95125 Catania, Italia,* vs@cs.unict.it

Abstract

Although the algebraic semantics of *place/transition Petri nets* under the *collective token philosophy* has been fully explained in terms of *(strictly) symmetric (strict) monoidal categories*, the analogous construction under the *individual token philosophy* is not completely satisfactory because it lacks *universality* and also *functoriality*. We introduce the notion of *pre-net* to recover these aspects, obtaining a fully satisfactory categorical treatment centered on the notion of *adjunction*. This allows us to present a purely logical description of net behaviours under the individual token philosophy in terms of theories and theory morphisms in *partial membership equational logic*, yielding a complete match with the theory developed by the authors for the *collective token* view of nets.

Key words: Petri Nets, Pre-Nets, Individual Token Philosophy, Monoidal Categories, Partial Membreship Equational Logic

[1] The first three authors have been partly supported by Office of Naval Research Contracts N00014-95-C-0225 and N00014-96-C-0114, by National Science Foundation Grant CCR-9633363, and by the Information Technology Promotion Agency, Japan, as part of the Industrial Science and Technology Frontier Program 'New Models for Software Architecture' sponsored by NEDO (New Energy and Industrial Technology Development Organization). Also research supported in part by US Army contract DABT63-96-C-0096 (DARPA); CNR Integrated Project *Metodi e Strumenti per la Progettazione e la Verifica di Sistemi Eterogenei Connessi mediante Reti di Comunicazione*; by Esprit Working Groups *CONFER2* and *COORDINA*; and by MURST project *Tecniche Formali per Sistemi Software*. The fourth author thanks the support by **BRICS** — **B**asic **R**esearch **i**n **C**omputer **S**cience, *Centre of the Danish National Research Foundation.*

Introduction

Petri nets, introduced by Petri in [15] (see also [17]), are one of the most widely used and evocative *models for concurrency*, because of the simple formal description of the net model, and of its natural characterisation of *concurrent* and *distributed systems*. The extensive use of Petri nets has given rise to different schools of thought concerning their semantical interpretation. In particular, we have the main distinction between *collective* and *individual token philosophies* (e.g., see [6]).

According to the collective token philosophy (CTph), net semantics should not distinguish among different instances of the idealised resources (the so-called 'tokens') that are at the basis of net behaviour, because any such instance is *operationally* equivalent to all the others. This view disregards that operationally equivalent resources may have different origins and histories, carrying different *causality* information. Selecting one instance of a resource rather than another, may be as different as being or not causally dependent on some previous event. And this may well be an information one is not ready to discard, which is the point of view of the individual token philosophy (ITph).

The net theory developed under the CTph is very well consolidated. The relationships between its *computational*, *algebraic* and *logical* interpretations has been precisely stated (see, e.g., [10,3]). In particular, the concurrent operational behaviour of a net is characterized by *commutative processes* [2], whose construction exactly corresponds: (1) at the semantic level, to the universal construction $\mathcal{T}(\_)$ of a *strictly symmetric (strict) monoidal category* (the arrows of $\mathcal{T}(N)$ represent the commutative processes of the net N), and (2) at the logical level, to a suitable morphism between theories in *partial membership equational logic* (**PMEqtl**), a logical framework introduced in [9].

Under the ITph, the relationships between these different interpretations are more complicated. Building on the notion of *process* presented in [7], several authors have shown that the semantics of a net can still be understood in terms of *symmetric monoidal categories*, but their constructions do not work properly 'in the large,' i.e., they fail to preserve at the semantic level some ordinary *simulation morphisms* between nets given at the level of theories (cf. [5,19], see [13] for an overview).

More precisely, a simple variation of processes called *concatenable processes* is introduced in [5], which admits sequential composition and yields a symmetric strict monoidal category $\mathcal{P}(N)$ for each net N, but such construction is not functorial. Indeed, for N and N' two nets such that the structure of N can be embedded in that of N', it might be the case that the concurrent behaviour of N cannot be recovered by N', because equivalent computations in N must now be distinguished as simulated computations in N' (see Example 1.9). In [19] the situation is improved by introducing the notion of *strongly concatenable processes* as a slight refinement of concatenable processes, where a linear ordering is given to minimal and maximal places (whereas in concatenable processes, only instances of the same resource are ordered). The construction of strongly concatenable processes can be expressed via a *pseudo-functor* $\mathcal{Q}(\_)$, mapping net morphisms to symmetric strict monoidal functors in a way that preserves *identities* strictly, but composition of net morphisms only up to a natural isomorphism. The construction provided by $\mathcal{Q}(\_)$ is almost satisfactory, and indeed it extends to a functor from **Petri**, the category of Petri nets introduced in [10], to a quotient category of a suitable full subcategory of **SSMC**, the category of symmetric strict monoidal categories, and defines a left adjoint to a subcategory of such a quotient category. However the characterisation of such a subcategory is rather involved and ad hoc.

The main difficulty in extending the nice algebraic framework to the *ITph* is related to the fact that net morphisms in **Petri** allow replacing two different resources a and b by two not necessarily disjoint multisets u and v in the target net, in such a way that instances in their union $u + v$ can be partitioned only up to a certain degree of ambiguity, whereas the *ITph* would require a precise correspondence between the instances of such resources. In [19] such ambiguity is solved by carrying information about the mappings of all the possible *linear implementations* of a multiset, that is, for each transition $t: u \to v$, a basic arrow $t_{\bar{u},\bar{v}}: \bar{u} \to \bar{v}$ is introduced (and suitably axiomatized) in the semantic model, for any linearizations $\bar{u}$ and $\bar{v}$ (i.e., strings of places) of u and v.

We present an analogous construction based on the notion of *pre-net*. A pre-net can be thought of as a precise implementation of a net, where the abstract data structure of multisets is refined into a more concrete *string* structure, and where each transition $t: u \to v$ is simulated by *one* arbitrarily chosen (instead of *all*) linear implementation $t_{\bar{u},\bar{v}}: \bar{u} \to \bar{v}$ for some linearizations $\bar{u}$ and $\bar{v}$ of u and v. Note that pre-nets have a different computational interpretation than phrase-structure grammars, since we do not distinguish between terminal and non-terminal symbols and strings can be permuted before performing any step, i.e., ordinary grammars would generate just monoidal categories, without symmetries.

Although abandoning multisets might appear at first unnatural to net enthusiasts, our formal approach to the *ITph* enjoys several good properties.

- ▷ All the pre-net implementations of the same net share the same semantic model, i.e., the semantics is independent of the choice of linearizations.
- ▷ Algebraic models of pre-nets are freely generated and, therefore, preserve colimit constructions on nets, adding compositionality to the framework.
- ▷ The semantic model of a pre-net implementation coincides with the semantic model given by $\mathcal{Q}(\_)$ for the implemented net.
- ▷ The algebraic semantics of pre-nets can be rephrased also in the logical framework of **PMEqtl**.

This means that the investigation of the behavioural, algebraic and logical aspects of PT nets presented in [3], where the computational interpretation was driven by the *CTph*, is successfully extended to the individual token approach, using pre-nets to accomplish a better categorical construction of the algebraic view. Moreover, we are able to formalize the construction using **PMEqtl** techniques, as a straightforward theory morphism from the theory `PRE-NET` of pre-nets to the theory `SMONCAT` of symmetric strict monoidal categories. Such a characterisation can also have practical applications, as there are tools available that support executability of specification in partial algebras.

Structure of the Paper

In Section 1.1 we recall the basic definitions about PT nets, making clear the distinction between their two computational interpretations (individual vs collective), and summarizing the process-based approaches presented in the literature to accommodate the *ITph*. The corresponding categorical constructions are presented in Section 1.2, explaining their weaknesses.

Section 2 introduces pre-nets, defining their categorical semantics and the relationship with ordinary PT nets. In Section 3 we employ **PMEqtl** to formalize the logical aspects of our approach. A short Appendix recalls the basics of partial membership equational logic and the definition of the theory `MONCAT` of strict monoidal categories.

1 Background

1.1 Process View of PT Petri Nets

Place/transition Petri nets, the most widespread type of Petri nets, are graphs with distributed states described by (finite) distributions of resources (called 'tokens') in 'places.' These are usually called *markings* and are represented as multisets $u\colon S \to \mathbb{N}$, where $u(a)$ is the number of tokens that place a carries in u.

As a matter of notation, we shall use $\mu(S) = \{u\colon S \to \mathbb{N}\}$ to indicate the set of *finite* multisets on S, i.e., multisets with multiplicity zero on all but finitely many $a \in S$. Multiset union makes $\mu(S)$ a free commutative monoid on S (unit is $\varnothing$).

Definition 1.1 *A place/transition Petri net N (PT net for short) is a 4-tuple $(\partial_0, \partial_1, S, T)$, where S is a set of places, T is a set of transitions, and the functions $\partial_0, \partial_1\colon T \to \mu(S)$ assign, respectively, source and target to each transition. A marked PT net is a PT net together with an initial marking $u_0 \in \mu(S)$.*

Informally, $\partial_0(t)$ prescribes the minimum amount of resources needed to enable the transition t, whilst $\partial_1(t)$ describes the resources that the occurrence of t contributes to the global state. This is made explicit in the following definition, where we shall indicate multiset inclusion, union, and difference by, respectively, $\subseteq$, $+$, and $-$.

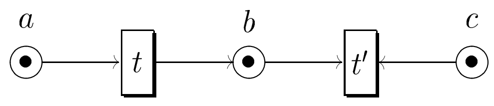

Fig. 1.

Definition 1.2 *Let u and v be markings and $X: T \to \mathbb{N}$ a finite multiset of transitions of a net N. We say that u evolves to v under the step X, in symbols $u\ [X\rangle\ v$, if the transitions in X are concurrently enabled at u, i.e., $\sum_{t \in T_N} X(t) \cdot \partial_0(t) \subseteq u$, and*

$$v = u + \textstyle\sum_{t \in T_N} X(t) \cdot (\partial_1(t) - \partial_0(t)).$$

A step sequence from u_0 to u_n is a sequence $u_0\ [X_1\rangle\ u_1 ... u_{n-1}\ [X_n\rangle\ u_n$.

In order to equip PT nets with a natural notion of morphism, we consider maps of transition systems that preserve the monoidal structure of states.

Definition 1.3 *A morphism of nets from $(\partial_0, \partial_1, S, T)$ to $(\partial'_0, \partial'_1, S', T')$ is a pair $f = \langle f_t, f_p \rangle$ where $f_t: T \to T'$ is a function, and $f_p: \mu(S) \to \mu(S')$ is a monoid homomorphism such that $\partial'_i \circ f_t = f_p \circ \partial_i$, for $i = 0, 1$. A morphism of marked nets is a morphism of nets such that $f_p(u_0) = u'_0$.*
We shall use **Petri** *to indicate the category of PT nets and their morphisms with the obvious componentwise composition of arrows.*

To compare the effects of the collective and the individual token philosophies on observing causal relations between fired transitions, let us consider the example in Figure 1 that we adapt from [6]. (As usual, boxes stand for transitions, circles for places, dots for tokens, and oriented arcs represent the functions ∂_0 and ∂_1.)

Both transitions t and t' are enabled in the initial marking $\{a, b, c\}$, but observe that the firing of t produces a second token in place b. According to the *ITph*, it makes a difference whether t' consumes the token in b originated from the firing of t, or the one coming from the initial marking. In the first case the occurrence of t' causally depends on that of t, whilst in the second the two firings are independent. In the *CTph*, instead, the two firings are always considered to be concurrent, because the firing of t does not affect the enabling condition of t'.

Since the *ITph* makes a distinction between resources in the same class that have different origins and histories, it is well supported by the *process*-based approach. Ideally, (*deterministic*) processes are computations carrying some explicit causal information between transition firings (*events*). This originates in an abstract view of processes as posets whose elements are labelled by transitions of the net [14,20,21].

Concretely, such computations are represented by suitable, structure preserving maps from a special class of nets into the net under inspection. The role of such maps is to disambiguate different firings of the same transition, and, at the same time, to give a precise account of the causal and distributed nature of the computations they represent.

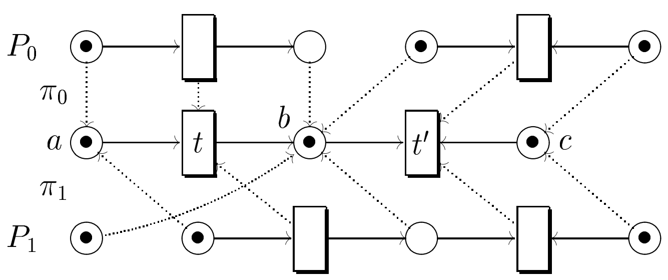

Fig. 2.

Definition 1.4 *A process net is a finite, acyclic net $P = (\partial_0, \partial_1, S, T)$ such that for all $t \in T$, $\partial_0(t)$ and $\partial_1(t)$ are sets (as opposed to multisets), and for all $t_0 \neq t_1 \in T$, $\partial_i(t_0) \cap \partial_i(t_1) = \varnothing$, for $i = 0, 1$.*
*A process of $N \in$ **Petri** is a morphism $\pi: P \to N$, where P is a process net and π is a net morphism which maps places to places (as opposed to more general morphisms which map places to markings).*
Processes $\pi: P \to N$ and $\pi': P' \to N$ are isomorphic, and thus identified, if there exists a net isomorphism $\psi: P \to P'$ such that $\pi = \psi; \pi'$.

We shall use $O(P)$ and $D(P)$ to denote the *minimal* (i.e., with empty pre-set) and *maximal* (i.e., with empty post-set) places of a process net P. (O stands for 'origins,' D for 'destinations.') For a process $\pi: P \to N$, the multiset $\pi(O(P))$ (with $\pi(O(P))(a) = |\pi^{-1}(a) \cap O(P)|$, for each place $a \in N$) represents the resources available to N before the *execution* of π, and $\pi(D(P))$ those available in N when the execution of π is completed.

Two processes for the (marked) net of Figure 1 are represented by the mappings π_0 and π_1 from the process nets P_0 and P_1 in Figure 2, where dotted arrows show the images of places and transitions (for readability, we have omitted the naming of the elements of P_0 and P_1).

Since processes represent computations, it is natural to seek a notion of sequential composition of those processes π and π' with $\pi(D(P)) = \pi'(O(P'))$, that is π' starts from the marking π terminates in. In general, there are several ways to do this, each corresponding to a different assignment of instances of the same place between $D(P)$ and $O(P')$. To overcome this ambiguity, *concatenable processes* were introduced in [5] by imposing a total ordering on origins and destinations that are instances of the same place.

Definition 1.5 *Given a labelling function $l: X \to Y$, a label-indexed ordering function for l is a family $\beta = \{\beta_y\}_{y \in Y}$ of bijections indexed by the elements of Y, where $\beta_y: l^{-1}(y) \to \{1, ..., |l^{-1}(y)|\}$.*

The idea is that Y is the set of places of a given net N, whilst X is (a subset of) the set of places of a process π for N such that l coincides with π on X. Then, for each place y in Y, we consider its inverse image through l, given by the set $l^{-1}(y) = \{x \in X \mid l(x) = y\}$. Basically, each β_y yields a total order over the elements in $l^{-1}(y)$, by stating a (bijective) correspondence with their positions in the ordering.

Definition 1.6 *A concatenable process θ for a PT net N is a triple (π, ℓ_O, ℓ_D), where $\pi: P \to N$ is a process for N, and ℓ_O, ℓ_D are label-indexed ordering functions for the labelling function π restricted to $O(P)$ and $D(P)$ respectively.*

A partial operation $\_;\_$ (associative and with identities) of concatenation can be defined for concatenable processes. They also admit a monoidal *parallel* composition $\_ \otimes \_$, yielding a symmetric strict monoidal category, whose symmetries are given by concatenable processes consisting only of places.

Due to space limitation, we refer the interested reader to [5] for the formal definitions of such operations. However, concatenable processes are still not completely satisfactory, because several net morphisms cannot be lifted to the level of behaviours (see Example 1.9 in Section 1.2).

To improve such a situation, the notion of *strongly concatenable process* has been introduced in [19], where a total order is imposed over origins and destinations, and not only on the instances of the same place. Strongly concatenable processes also admit sequential and parallel composition, yielding a symmetric strict monoidal category.

Definition 1.7 *A strongly concatenable process for a PT net N is a triple (π, ℓ_O, ℓ_D), where $\pi: P \to N$ is a process for N, and ℓ_O, ℓ_D are bijections $\ell_O: O(P) \to |O(P)|$ and $\ell_D: D(P) \to |D(P)|$ respectively.*

1.2 Categorical Semantics

Several aspects of Petri net theory can be profitably developed within category theory, see e.g. [20,10]. Here we focus on the approach initiated in [10] (other relevant references are [5,12,18,13,19]) which exposes the monoidal structure of Petri nets under the operation of parallel composition. In [10,5] it is shown that the sets of transitions can be endowed with appropriate algebraic structures in order to capture some basic constructions on nets.

For example, the *commutative processes* of [2], which represent the natural behavioural model for PT nets under the *CTph*, can be characterised adding a functorial *sequential* composition on the *monoid* of steps, thus yielding a strictly symmetric strict monoidal category $\mathcal{T}(N)$. Using **CMonCat** to denote the category of strictly symmetric strict monoidal categories (as objects) and strict monoidal functors (as arrows), $\mathcal{T}(\_)$ is a functor from **Petri** to **CMonCat**, and $\mathcal{T}(N)$ is the strictly symmetric strict monoidal category freely generated by N (seen as a graph whose nodes have a free monoidal structure).

The intuition here is that arrows of $\mathcal{T}(N)$ represent step sequences and arrow composition is their concatenation, whereas the monoidal operator $\oplus$ allows for parallel composition. It is shown in [5] that this algebraic structure describes precisely the processes à la Best and Devillers [2]. Indeed, the category $\mathcal{T}(N)$ can be inductively defined by simple inference rules and axioms, providing a complete and sound axiomatization of the algebra of the commutative processes of N.

Under the *ITph*, one might expect analogous results to hold, relating symmetric strict monoidal categories and (strongly) concatenable processes, introducing *symmetries* to model the possible reorganizations of minimal and maximal places of a process. Let us consider concatenable processes first, where the ordering on minimal and maximal places is imposed on instances of the same place only. We recall here the definition of the category $\mathcal{P}(N)$ introduced in [5] and finitely axiomatized in [18].

Definition 1.8 *Let N be a PT net. The category $\mathcal{P}(N)$ is the monoidal quotient of the free symmetric strict monoidal category $\mathcal{F}(N)$ generated by N, modulo the axioms*

$$\gamma_{a,b} = id_a \otimes id_b \qquad \textit{if } a,b \in S_N, \textit{ and } a \neq b$$
$$s;t;s' = t \qquad \textit{if } t \in T_N \textit{ and } s,s' \textit{ are symmetries}$$

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

Though the construction $\mathcal{P}(N)$ axiomatizes exactly the concatenable processes of N, it lacks functoriality, as shown by the following example.

Example 1.9 *Consider the nets N and N' shown in Figure 3 and the net morphism $f: N \to N'$ such that $f_t(t_i) = t'_i$, $f_p(a_i) = a'$, and $f_p(b_i) = b'_i$ for $i = 0,1$. The morphism f cannot be extended to a monoidal functor $\mathcal{P}(f): \mathcal{P}(N) \to \mathcal{P}(N')$. In fact, if such an extension F existed, then*

$$F(t_0 \otimes t_1) = F(t_0) \otimes F(t_1) = t'_0 \otimes t'_1$$
$$F(t_1 \otimes t_0) = F(t_1) \otimes F(t_0) = t'_1 \otimes t'_0$$

by the monoidality of F, but since $\gamma_{a_0,a_1} = id_{a_0} \otimes id_{a_1}$ and $\gamma_{b_0,b_1} = id_{b_0} \otimes id_{b_1}$ in $\mathcal{P}(N)$ (by the first axiom in Definition 1.8), then

$$t_0 \otimes t_1 = (t_0 \otimes t_1); \gamma_{b_0,b_1} = \gamma_{a_0,a_1}; (t_1 \otimes t_0) = t_1 \otimes t_0$$

in $\mathcal{P}(N)$ (by the naturality of γ). Thus, it would follow that $t'_0 \otimes t'_1 = t'_1 \otimes t'_0$ in $\mathcal{P}(N')$, which is absurd because $\gamma_{a',a'} \neq id_{a'} \otimes id_{a'}$.

The problem is of course due to the fact that when two different places a_0 and a_1 are mapped into the same place a' via a net morphism, then it should be the case that $\gamma_{a,b} = id_a \otimes id_b$ be mapped into $\gamma_{a',a'} \neq id_{a'} \otimes id_{a'}$ via a monoidal functor, which is not possible. The valuable *pseudo functorial* construction $\mathcal{Q}(\_): \mathbf{Petri} \to \mathbf{SSMC}$ of [19], recovering strongly concatenable processes, improves the situation, in the sense that it preserves composition only up to a monoidal natural isomorphism (details in [19]).

Definition 1.10 *Let N be a PT net. The category $\mathcal{Q}(N)$ is obtained from the symmetric strict monoidal category freely generated from the places of N and, for each transition $t: u \to v$ of N, an arrow $t_{\bar{u},\bar{v}}: \bar{u} \to \bar{v}$ for each pair of linearizations (as strings) of the pre- and post-sets of t, by quotienting modulo the axiom*

(1) $s; t_{\bar{u},\bar{v}}; s' = t_{\bar{u}',\bar{v}'}$ *for $s: \bar{u}' \to \bar{u}$ and $s': \bar{v} \to \bar{v}'$ symmetries.*

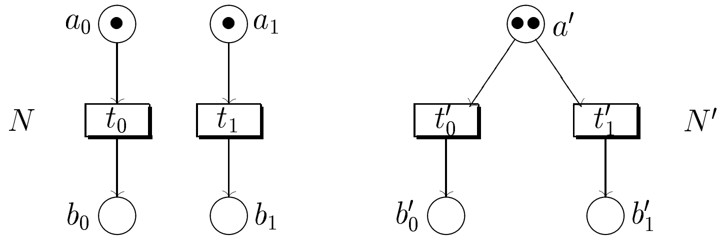

Fig. 3.

Our point in this paper is that functoriality is missing in these constructions because PT nets rely on a 'state as multiset' paradigm, whereas the *ITph* imposes a distinction on different instances of the same resource. Hence, as a solution to this problem, we propose a refined view of nets, one such that the associated notion of morphism behaves better w.r.t. the construction of the category of processes.

2 Pre-Nets

Pre-nets are nets whose states are *strings* of tokens (as opposed to *multisets*). Such states, called *records*, can be seen as totally ordered markings, and also as 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 consumed and the results are produced, as if it were an elementary strongly concatenable process.

We shall use $\lambda(S)$ to indicate the set of finite strings on S. String concatenation (denoted by *juxtaposition*) makes $\lambda(S)$ a free monoid on S, the unit being the empty string ϵ. Moreover, for $w \in \lambda(S)$, we write $|w|$ to denote the length of w, w_i to denote the i-th element of w, and $\mu(w)$ to denote the underlying multiset of w.

Definition 2.1 *A pre-net is a tuple $R = (\zeta_0, \zeta_1, S, T)$, where S is a set of places, T is a set of transitions, and $\zeta_0, \zeta_1 \colon T \to \lambda(S)$ are functions assigning, respectively, source and target to each transition.*

The idea is that, given a PT net N, we can arbitrarily choose a pre-net representation of N. This corresponds to fixing a total order for the pre- and post-set of each transition. This differs from the approach proposed in [19], where, in order to avoid a choice, *all* the possible linearizations of the pre- and post-sets are considered in the alternative presentation of the net. We will show that to recover the process semantics of N it suffices to choose one representative for each transition.

Definition 2.2 *A morphism of pre-nets from (ζ_0, ζ_1, S, T) to $(\zeta'_0, \zeta'_1, S', T')$ is a pair $\langle g_t, g_p\rangle$ where $g_t \colon T \to T'$ is function, and $g_p \colon \lambda(S) \to \lambda(S')$ is a monoid homomorphism such that $\zeta'_i \circ g_t = g_p \circ \zeta_i$, for $i = 0, 1$. We denote by* **PreNet** *the category of pre-nets and their morphism with the obvious composition.*

Table 1

$$\frac{w \in \lambda(S_R)}{id_w\colon w \to w \in \mathcal{Z}(R)} \qquad \frac{t \in T_R,\ \zeta_0(t) = \bar{u},\ \zeta_1(t) = \bar{v}}{t\colon \bar{u} \to \bar{v} \in \mathcal{Z}(R)}$$

$$\frac{w, w' \in \lambda(S_R)}{c_{w,w'}\colon ww' \to w'w \in \mathcal{Z}(R)}$$

$$\frac{\alpha\colon \bar{u} \to \bar{v},\ \beta\colon \bar{u}' \to \bar{v}' \in \mathcal{Z}(R)}{\alpha \otimes \beta\colon \bar{u}\bar{u}' \to \bar{v}\bar{v}' \in \mathcal{Z}(R)} \qquad \frac{\alpha\colon \bar{u} \to \bar{v},\ \beta\colon \bar{v} \to \bar{v}' \in \mathcal{Z}(R)}{\alpha;\beta\colon \bar{u} \to \bar{v}' \in \mathcal{Z}(R)}$$

The notion of morphism for pre-nets is therefore tighter than that for PT nets, because mappings must preserve the ordering in which the tokens are produced and consumed by each transition. Within this view, there is a trivial forgetful functor from **PreNet** to **Petri** that forgets about such orderings.

Proposition 2.3 *The map $\mathcal{A}$, from pre-nets to PT nets, sending each pre-net $R = (\zeta_0, \zeta_1, S, T)$ to the net $\mathcal{A}(R) = (\partial_0, \partial_1, S, T)$ with $\partial_i(t) = \mu(\zeta_i(t))$ for each $t \in T$ and $i = 0, 1$, extends to a functor from* **PreNet** *to* **Petri***.*

The functor $\mathcal{A}(\_)\colon \mathbf{PreNet} \to \mathbf{Petri}$ is neither full, nor faithful. However, if we consider a category **Net** whose objects are either PT nets or pre-nets and whose morphisms are graph morphisms with monoid homomorphism for node components, then **Petri** and **PreNet** are full subcategories of **Net** and the inclusion of **Petri** into **Net** has a left adjoint $\widehat{\mathcal{A}}\colon \mathbf{Net} \to \mathbf{Petri}$ (with $\widehat{\mathcal{A}}_{|\mathbf{PreNet}} = \mathcal{A}$ and $\widehat{\mathcal{A}}_{|\mathbf{Petri}} = 1_{\mathbf{Petri}}$), yielding a *coreflection*, i.e., **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, which supports and further motivates our proposed approach to the *ITph*.

Under the *ITph*, the natural algebraic models for representing concurrent computations on pre-nets live in the category **SSMC**. More precisely, we are only interested in the full subcategory consisting of categories whose monoid of objects is freely generated. This is of course the most natural choice supporting the notion of distributed state as a collection of atomic entities, viz., tokens in places, which net theory is based on. We denote such category by **FSSMC**.

Proposition 2.4 *The obvious forgetful functor from the category* **FSSMC** *to the category* **PreNet** *admits a left adjoint $\mathcal{Z}$.*

Proof. *The category $\mathcal{Z}(R)$ has as objects the elements in $\lambda(S_R)$, and as arrows those generated by the rules in Table 1 modulo the axioms of strict monoidal categories (associativity, functoriality, identities, unit), including the coherence axioms that make of $c_{w,w'}$ the symmetry natural isomorphisms.*

The above construction is of course well-known; it can be traced back to work on coherence by MacLane and others, and even more closely to Pfender's construction of a free S-monoidal category [16]. In computer science similar constructions were given by Hotz's X-categories [8] and by Benson [1], with grammars as the primary area of application, and therefore for categories that were not necessarily symmetric.

In our case the symmetric structure is essential; in fact it means that the construction is independent of the choice of linearization (Theorem 2.5). Furthermore, what we really want is a quotient of the free construction, as explained in Theorem 2.6. The main result is that any two pre-nets representing isomorphic PT nets yield the same algebraic net semantics.

Theorem 2.5 *Let $R, R' \in$ **PreNet** with $\mathcal{A}(R) \simeq \mathcal{A}(R')$, then $\mathcal{Z}(R) \simeq \mathcal{Z}(R')$.*

Proof. *Let ϕ be a net isomorphism between $\mathcal{A}(R)$ and $\mathcal{A}(R')$ (e.g., $\phi = id_N$ if $\mathcal{A}(R) = \mathcal{A}(R') = N$). Thus, ϕ maps places into places. Then, for each transition t of R and $i = 0, 1$, $\phi(\zeta_i(t)) = \zeta_i(\phi(t))$ up to a permutation, say $\gamma(i,t)\colon \phi(\zeta_i(t)) \to \zeta_i(\phi(t))$. Therefore, the monoidal functor Φ from $\mathcal{Z}(R)$ to $\mathcal{Z}(R')$ defined as*

- ▷ $\Phi(a) = \phi(a)$ *for each place a of R,*
- ▷ $\Phi(c_{w,w'}) = c_{\phi(w),\phi(w')}$ *for any strings w, w' in $\lambda(S)$,*
- ▷ $\Phi(t) = \gamma(0,t); \phi(t); \gamma(1,t)^{-1}$ *for each transition t in R,*

is an isomorphism of symmetric strict monoidal categories.

Theorem 2.6 *Let R be a pre-net. The category $\mathcal{Z}(R)$ quotiented by the axiom*

$$t = s_0; t; s_1,$$

for any transition $t\colon \bar{u} \to \bar{v}$ and symmetries $s_0\colon \bar{u} \to \bar{u}$ and $s_1\colon \bar{v} \to \bar{v}$ is equivalent to the category $\mathcal{Q}(\mathcal{A}(R))$ of strongly concatenable processes.

Proof. *Easy by the following argument. In $\mathcal{Q}(\mathcal{A}(R))$ we add a transition $t_{\bar{u},\bar{v}}$ for each transition of N and each pair of linearizations $\bar{u}$ and $\bar{v}$ of its pre- and post-set, and we quotient by the axiom (1). On the other hand, in $\mathcal{Z}(R)$ we arbitrarily fix only one linearization of t, say $t_{\bar{u},\bar{v}}$, but we get all the others for free by composing $t_{\bar{u},\bar{v}}$ with symmetries (as s and s' in the axiom (1) of Definition 1.10).*

This result stresses an important point: *any pre-net representation of the net $\mathcal{A}(R)$ is as good as R.* More importantly, since left adjoints preserves colimits, it follows that the semantics of the (colimit) composition of pre-nets (e.g., seen as 'programs') can be studied just mimicking such a composition on their semantic interpretations.

An interesting question concerns relating morphisms of PT nets, rather than pre-nets, to the algebraic models obtained by pre-nets. To this purpose, for $f\colon N \to N'$ a morphism in **Petri**, consider the PT net N_f that has the same transitions as N and the same places as N' with a $t\colon f(u) \to f(v)$ in N_f corresponding to $t\colon u \to v$ in N. It can be easily verified that it is decomposable as $f = g; h$ with $g = \langle id_{T_N}, f_p\rangle\colon N \to N_f$ and $h = \langle f_t, id_{\mu(S_{N'})}\rangle\colon N_f \to N'$. Now, for any pre-net R such that $\mathcal{A}(R) = N$ and for any linearization g'_p of $g_p = f_p$, we can find a pre-net R_f such that $\mathcal{A}(R_f) = N_f$ and $\bar{g} = \langle id_t, g'_p \rangle : R \to R_f$. Analogously, for any pre-net R' such that $\mathcal{A}(R') = N'$ we can find a pre-net R'_f such that $\mathcal{A}(R'_f) = N_f$ with $\bar{h} = \langle f_t, id_{\lambda(S_{N'})} \rangle : R'_f \to R'$. It so happens that, in general, there might be no morphism in **PreNet** between R_f and R'_f for simulating f. However, resorting to the semantics models for pre-nets, the proof of Theorem 2.5 points us to a constructive way to relate $\mathcal{Z}(R_f)$ and $\mathcal{Z}(R'_f)$, thus yielding a lifting of f to a monoidal functor $F = \mathcal{Z}(\bar{g}); \Phi; \mathcal{Z}(\bar{h})$ between $\mathcal{Z}(R)$ and $\mathcal{Z}(R')$.

3 Recovering the Algebraic Semantics of Nets via Theory Morphisms

The algebraic semantics of PT Petri nets can be expressed very compactly by means of a morphism between theories in *partial membership equational logic* (**PMEqtl**) [9], a logic of partial algebras with subsorts and subsort polymorphism whose *sentences* are Horn clauses on equations $t = t'$ and membership assertions $t : s$, and whose features (partiality, poset of sorts, membership assertions) offer a natural framework for the specification of categorical structures. The Appendix provides an informal introduction to the main ideas of **PMEqtl**. We refer to [9,11] for self-contained presentations. The following proposition (cf. [9]) states an important result, relating the models of two theories that are themselves related by a theory morphism.

Proposition 3.1 *The forgetful functor $\mathcal{U}_H : \mathbf{PAlg}_{T'} \to \mathbf{PAlg}_T$ associated to a theory morphism $H : T \to T'$ has a left adjoint $\mathcal{F}_H : \mathbf{PAlg}_T \to \mathbf{PAlg}_{T'}$.*

The logical account of the *CTph* is discussed in [3], where, using a self-explanatory Maude-like notation,[2] we defined the theories `PETRI` of PT nets and `CMONCAT` of strictly symmetric strict monoidal categories together with an intermediate theory `CMON-AUT` of automata whose states form a commutative monoid. Then, the composition of the obvious inclusion functor from **Petri** into $\mathbf{PAlg}_{\texttt{CMON-AUT}}$ and the free functor $\mathcal{F}_{\texttt{V}}$ from $\mathbf{PAlg}_{\texttt{CMON-AUT}}$ to $\mathbf{PAlg}_{\texttt{CMONCAT}}$ associated to the theory morphism `V` from `CMON-AUT` to `CMONCAT` corresponds exactly to the functor $\mathcal{T}(\_) : \mathbf{Petri} \to \mathbf{CMonCat}$.

Analogously, we define here the theories of pre-nets, of symmetric strict monoidal categories, and of suitable intermediate models, so as to deal with the *ITph*. The main difference is that the monoidal theory we are interested in is not commutative, and that symmetries must be explicitly added to rearrange the object components in the symmetric category (in the *CTph*, symmetries are collapsed into identities).

In order to define a theory in **PMEqtl** that represents pre-nets and their morphisms, we first introduce a theory whose models are automata whose states form a monoid.

[2] Maude [4] is a language recently developed at SRI International; it is based on rewriting logic and supports execution of membership equational logic specifications.

```
fth MON-AUT is
 sorts State Transition.
 op 1 : -> State.
 op _⊗_ : State State -> State [assoc id: 1].
 ops origin(_) destination(_) : Transition -> State.
endfth
```

Note that the attributes `assoc` and `id: 1` state the monoidality of the operator $\_ \otimes \_$ on states of automata.

Exploiting the modularity features of Maude, we can characterise the category **PreNet** of pre-nets as a subcategory of $\mathbf{PAlg}_{\mathtt{MON-AUT}}$. We import a functional module `LIST[E :: TRIV]` of records, parametrised on a functional theory of `TRIV` of elements, whose models are sets corresponding to the places of the net.

```
fth TRIV is
 sort Element.
endfth
```

```
fmod LIST[E :: TRIV] is
 sort List.
 subsort Element < List.
 op ϵ : -> List.
 op _::_ : List List -> List [assoc id: ϵ].
endfm
```

```
fth PRE-NET[S :: TRIV] is
 protecting LIST[S]
           renamed by (sort List to Record.).
 sort Transition.
 ops pre(_) post(_) : Transition -> Record.
endfth
```

The inclusion functor from **PreNet** to $\mathbf{PAlg}_{\mathtt{MON-AUT}}$ is induced as the forgetful functor of a theory morphism `J` specified as a `view` in Maude as follows.

```
view J from MON-AUT to PRE-NET[S :: TRIV] is
 sort State to Record.
 op origin(_) to pre(_).
 op destination(_) to post(_).
 op 1 to ϵ.
 op _⊗_ to _::_.
endview
```

Proposition 3.2 *The category* **PreNet** *is a full subcategory of* $\mathbf{PAlg}_{\mathtt{MON-AUT}}$.

In order to define the theory `SMONCAT` of symmetric strict monoidal categories, we exploit the definition of the theory `MONCAT` of monoidal categories from [11], which for the reader's convenience is reported in the Appendix.

```
fth SMONCAT is including MONCAT.
 op γ(_,_) : Object Object -> Arrow.
 vars a,a',b,b',c : Object.
 vars f,f' : Arrow.
 eq d(γ(a,b)) = a⊗b.
 eq c(γ(a,b)) = b⊗a.
 eq γ(a,1) = a.
 eq γ(1,a) = a.
 eq γ(a⊗b,c) = (a⊗γ(b,c));(γ(a,c)⊗b).
 eq γ(a,b);γ(b,a) = a⊗b.
 ceq (f⊗f');γ(b,b') = γ(a,a');(f'⊗f)
     if d(f) == a and d(f') == a' and c(f) == b and c(f') == b'.
endfth
```

Finally, the algebraic semantics of pre-nets, i.e., the construction $\mathcal{Z}(\_)$, can be easily recovered via the theory morphism `W` defined as follows.

```
view W from MON-AUT to SMONCAT is
 sort State to Object.
 sort Transition to Arrow.
 op origin(_) to d(_).
 op destination(_) to c(_).
endview
```

Proposition 3.3 *The functor $\mathcal{Z}(\_)$:* $\mathbf{PreNet} \to \mathbf{SSMC}$ *is the composition*

$$\mathbf{PreNet} \hookrightarrow \mathbf{PAlg}_{\texttt{MON-AUT}} \xrightarrow{\mathcal{F}_{\texttt{W}}} \mathbf{PAlg}_{\texttt{SMONCAT}}.$$

Concluding Remarks

We have investigated the issue of a satisfactory categorical semantics of PT nets under the *ITph*. In particular, we have introduced *pre-nets* — a refined version of PT nets, where each transition is assigned an ordering for consuming (producing) resources — and have shown that they can provide a faithful description of net behaviours.

The conceptual framework of this paper is summarised in Table 2, which shows our research programme and results on the *behavioural*, *algebraic* and *logical* aspects of the two computational interpretations of PT nets, that is the *CTph* and the *ITph*, from the viewpoints of the structures suited to each of them and their mutual relationships.

The first row of Table 2 has been treated in [3]. As for the individual token interpretation, we have proposed here the categorical construction $\mathcal{Z}(R)$, based on pre-nets, as a suitable algebraic framework for nets. It offers some advantages w.r.t. previous constructions because it is functorial ($\mathcal{P}(N)$ is not), and very simple ($\mathcal{Q}(N)$ is not). Moreover, for the preservation properties of adjoints, the semantic models of nets obtained as colimit constructions is found

Table 2

	Structures		
Computation Model	**Behavioural**	**Algebraic**	**Logical**
Nets and Collective Token Philosophy	Conf. structures, CTS, Commutative processes	$\mathcal{T}(N)$	$\mathtt{CAT} \otimes \mathtt{CMON}$
Nets and Individual Token Philosophy	Processes, Concatenable Procs, Strongly Conc. Procs	$\mathcal{P}(N)$, $\mathcal{Q}(N)$, $\mathcal{Z}(R)$	$\mathtt{CAT} \otimes \mathtt{MON} + \mathtt{SYM}$

by applying the same constructions on models. For instance, the algebraic model of the *pushout* of two nets — which is often useful for combining two nets merging some of their places — is the pushout of their semantics.

From the logical viewpoint, it is not difficult to formulate a theory `SYM` of permutations and symmetries (cf. [18]) bridging the gap from strictly symmetric monoidal categories to categories symmetric only up to coherent isomorphism.

References

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

[2] E. Best and R. Devillers (1987), Sequential and Concurrent Behaviour in Petri Net Theory. *Theoretical Computer Science* **55**, 87–136, Elsevier.

[3] R. Bruni, J. Meseguer, U. Montanari, and V. Sassone (1998), A Comparison of Petri Net Semantics under the Collective Token Philosophy, in *Proceedings of the 4th Asian Computing Science Conference, ASIAN 98*, J. Hsiang and A. Ohori (Eds.), *LNCS* **1538**, 225–244, Springer-Verlag.

[4] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and J. Quesada (1999), Maude: Specification and programming in rewriting logic, SRI International, revised version. Available at `http://maude.csl.sri.com/manual`.

[5] P. Degano, J. Meseguer, and U. Montanari (1996), Axiomatizing the Algebra of Net Computations and Processes. *Acta Informatica* **33**(7), 641–667, Springer-Verlag.

[6] R.J. Van Glabbeek and G.D. Plotkin (1995), Configuration Structures, in *Proceedings of the 10th Symposium on Logics in Computer Science*, 199–209, IEEE Press.

[7] U. Goltz and W. Reisig (1983), The Non-Sequential Behaviour of Petri Nets. *Information and Computation* **57**, 125–147, Academic Press.

[8] G. Hotz (1965), Eine Algebraisierung des Syntheseproblemen von Schaltkreisen, I and II. *EIK* **1**, 185–206, 209–231.

[9] J. Meseguer (1998), Membership Equational Logic as a Logical Framework for Equational Specification, in *Proceedings of the 12th WADT Workshop on Algebraic Development Techniques*, F. Parisi-Presicce (Ed.), *Lecture Notes in Computer Science* **1376**, 18–61, Springer-Verlag.

[10] J. Meseguer and U. Montanari (1990), Petri Nets are Monoids. *Information and Computation* **88**(2), 105–155, Academic Press.

[11] J. Meseguer and U. Montanari (1998), Mapping Tile Logic into Rewriting Logic. in *Proceedings of the 12th WADT Workshop on Algebraic Development Techniques*, F. Parisi-Presicce (Ed.), *LNCS* **1376**, 62–91, Springer-Verlag.

[12] J. Meseguer, U. Montanari, and V. Sassone (1996), Process versus Unfolding Semantics for Place/Transition Petri Nets. *Theoretical Computer Science* **153**(1-2), 171–210, Elsevier.

[13] 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.

[14] M. Nielsen, G. Plotkin, and G. Winskel (1981), Petri Nets, Event Structures and Domains, Part 1. *Theoretical Computer Science* **13**, 85–108, Elsevier.

[15] C.A. Petri (1962), *Kommunikation mit Automaten.* PhD thesis, Institut für Instrumentelle Mathematik, Bonn.

[16] M. Pfender (1974), Universal Algebra in S-Monoidal Categories, Algebraberichte **20**, Department of Mathematics, University of Munich.

[17] W. Reisig (1985), *Petri Nets (an Introduction).* EATCS Monographs on Theoretical Computer Science **4**, Springer-Verlag.

[18] V. Sassone (1996), An Axiomatization of the Algebra of Petri Net Concatenable Processes. *Theoretical Computer Science* **170**, 277–296, Elsevier.

[19] V. Sassone (1998), An Axiomatization of the Category of Petri Net Computations. *Mathematical Structures in Computer Science* **8**, 117–151, Cambridge University Press.

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

[21] 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.), *LNCS* **354**, 365–397, Springer-Verlag.

Table A.1

```
fth CAT is
 sorts Object Arrow.
 subsort Object < Arrow.
 ops d(_) c(_) : Arrow -> Object.
 op  _;_.
 var a : Object.
 vars f g h : Arrow.
 eq  d(a) = a.
 eq  c(a) = a.
 ceq a;f = f if d(f) == a.
 ceq f;a = f if c(f) == a.
 cmb f;g : Arrow iff c(f) == d(g).
 ceq d(f;g) = d(f) if c(f) == d(g).
 ceq c(f;g) = c(g) if c(f) == d(g).
 ceq (f;g);h = f;(g;h) if c(f) == d(g) and c(g) == d(h).
endfth

fth MON is
 sort Monoid.
 op 1 : -> Monoid.
 op _⊗_ : Monoid Monoid -> Monoid [assoc id: 1].
endfth

fth MONCAT is MON ⊗ CAT
 renamed by (
             sort (Monoid,Object) to Object.
             sort (Monoid,Arrow) to Arrow.
             op  1 left to 1.
             op _⊗_ left to _⊗_.
             op _;_ right to _;_.
             op d(_) right to d(_).
             op c(_) right to c(_).
            ).
endfth
```

A Partial Membership Equational Logic

A *theory* in **PMEqtl** is a pair $T = (\Omega, \Gamma)$, where Ω is a signature over a *poset* of sorts and Γ is a set of Horn sentences in the language of Ω. We denote by $\mathbf{PAlg}_\Omega$ the category of partial Ω-algebras, and by $\mathbf{PAlg}_T$ the full subcategory consisting of T-algebras, i.e., those partial Ω-algebras that satisfy all the sentences in Γ.

For example, the poset of sorts of the **PMEqtl**-theory `CAT` of categories (see Table A.1) is `Object` $\leq$ `Arrow`. There are two unary operations `d(_)` and `c(_)`, for *domain* and *codomain*, and a binary composition operation `_;_` defined if and only if the codomain of the first argument is equal to the domain of the second argument. Functions with explicitly given domain and codomain are always *total*. The definition of the theory `MONCAT` of strict monoidal categories used in Section 3 is almost effortless thanks to the tensor product construction of theories, which is informally defined as follows.

Let $T = (\Omega, \Gamma)$ and $T' = (\Omega', \Gamma')$ be theories in partial membership equational logic, with $\Omega = (S, \leq, \Sigma)$ and $\Omega' = (S', \leq', \Sigma')$. Their *tensor product* $T \otimes T'$ is the theory with signature $\Omega \otimes \Omega'$ having: poset of sorts $(S, \leq) \times (S', \leq')$, and signature $\Sigma \otimes \Sigma'$, with operators $f_l \in (\Sigma \otimes \Sigma')_n$ and $g_r \in (\Sigma \otimes \Sigma')_m$ for each $f \in \Sigma_n$ and $g \in \Sigma'_m$ (indices l and r stand respectively for `left` and `right` and witness whether the operator is inherited from the left or from the right component). The axioms of $T \otimes T'$ are the determined from those of T and T' as explained in [11].

The essential property of the tensor product of theories is expressed by the following theorem, where $\mathbf{PAlg}_T(\mathbf{C})$ indicates the category of T-algebras taken over the base category $\mathbf{C}$ rather than over $\mathbf{Set}$.

Theorem A.1 *Let T, T' be theories in* $\mathbf{PMEqtl}$. *Then, we have the following isomorphisms of categories:*

$$\mathbf{PAlg}_T(\mathbf{PAlg}_{T'}) \simeq \mathbf{PAlg}_{T \otimes T'} \simeq \mathbf{PAlg}_{T'}(\mathbf{PAlg}_T).$$

Hence, the theory `MONCAT` is defined in Table A.1 by applying the tensor product construction to `CAT` and to the theory `MON` of (strict) monoids. Notice the use of `left` and `right` corresponding to the indices l and r discussed above.