Information and Computation **170**, 207–236 (2001)
doi:10.1006/inco.2001.3050, available online at http://www.idealibrary.com on IDEAL®

Functorial Models for Petri Nets[1]

Roberto Bruni

Dipartimento di Informatica, Università di Pisa, Italy
E-mail: bruni@di.unipi.it

José Meseguer

Computer Science Laboratory, SRI International, Menlo Park, California
E-mail: meseguer@csl.sri.com

Ugo Montanari

Dipartimento di Informatica, Università di Pisa, Italy
E-mail: ugo@di.unipi.it

and

Vladimiro Sassone

Dipartimento di Matematica e Informatica, Università di Catania, Italy
E-mail: vs@dmi.unict.it

Received March 1, 2000

We show that although the algebraic semantics of place/transition Petri nets under the collective token philosophy can be fully explained in terms of strictly symmetric 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-nets to overcome this, obtaining a fully satisfactory categorical treatment, where the operational semantics of nets yields an adjunction. This allows us to present a uniform logical description of net behaviors under both the collective and the individual token philosophies in terms of theories and theory morphisms in partial membership equational logic. Moreover, since the universal property of adjunctions guarantees that colimit constructions on nets are preserved in our algebraic models, the resulting semantic framework has good compositional properties.

Key Words: PT Petri nets; pre-nets; collective/individual token philosophy; monoidal categories; partial membership equational logic; configuration structures; concurrent transition systems.

INTRODUCTION

Petri nets, introduced by Petri in [29] (see also [32]), are one of the most widely used and evocative models for concurrency, because of the simple formal description of the net model and its natural characterization of concurrent and distributed systems. We will focus on place/transition Petri nets (PT nets). The extensive use of PT nets has given rise to different schools of thought regarding their semantical interpretation. In particular, there is an overall distinction between *collective* and *individual token philosophies* (see, e.g., [13]). According to the collective token philosophy (*CTph*), net semantics should not distinguish between different instances of the idealized resources which are positioned

[1] This research has been partly supported by the Office of Naval Research Contracts N00014-95-C-0225, N00014-96-C-0114 and N00014-99-C-0198, by a 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 the research was supported in part by CNR Integrated Project *Progettazione e Verifica di Sistemi Eterogenei Connessi mediante Reti di Comunicazione*; by Esprit Working Groups *CONFER2* and *COORDINA*; and by MURST project *TOSCa: Tipi, Ordine Superiore e Concorrenza*. The fourth author would like to thank **BRICS**, *Centre of the Danish National Research Foundation* for their support.

0890-5401/01 $35.00

at the same place, because any such instance (called a *token*) is operationally equivalent to all the others. This view chooses to ignore 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 being causally dependent on some previous event, and this may well be information which one is not ready to discard, which is the point of view of the individual token philosophy (*ITph*). Of course, causal dependencies may influence the degree of concurrency in the computations, and therefore the *CTph* and the *ITph* lead to different concurrent semantics.

The paper [21] proposed an algebraic approach to the analysis of concurrent net computations based on the observation that the monoidal structure of PT net states—multisets over the set of places form a free commutative monoid, where the monoidal operation is the multiset union and the unit is the empty multiset—can be lifted to transitions (and then to computations) in such a way that the suitably axiomatized terms of the new algebra yield an initial model for the concurrent semantics of the nets (according to the *CTph*). This construction respects the intuitive simulation morphisms between nets, when these are seen as graphs with structured nodes, as it can be expressed as a functor from the category **Petri** of PT nets (as objects) and simulation morphisms (as arrows) to the category **CMonCat** of strictly symmetric strict monoidal categories (as objects) and symmetric monoidal functors (as arrows), where computation models live. Moreover, when one considers the full subcategory of **CMonCat** consisting of categories whose objects are the elements of a free monoid, then the functorial construction of the concurrent model of computations is the left adjoint to an obvious forgetful functor. Thus, the universal properties of the adjunction guarantee the compositionality of the semantic framework: The semantics of all net compositions that can be expressed via colimit constructions in **Petri** (e.g., via pushouts) is given by taking the corresponding colimit in the category of models.

In this paper we investigate the operational, algebraic and logical aspects of PT nets under both the *CTph* and the *ITph*, exploiting the features of the algebraic approach to establish formal relationships between different proposals. As in [21], an important feature of our comparison and integration of different approaches to Petri net semantics in both the collective and individual token philosophies is that we emphasize the functorial character of the different semantic constructions. This is important for at least two reasons. First, by defining the appropriate categories, we make explicit the associated morphisms, which correspond to appropriate notions of *simulation* or *refinement* between nets. Second, by seeking functorial constructions, we ensure that they behave properly not only on the objects, but also on the simulation maps, and, furthermore, by making such constructions adjunctions we achieve a high degree of modularity because of the colimit-preserving nature of left adjoints. The abstract formulation of the algebraic semantics facilitates its comparison with other proposals, providing helpful mathematical tools for net analysis. Moreover, several efficient languages have been developed that support algebraic specifications and hence can be used to manipulate and reason about semantics models automatically.

Another important theme in this paper is the logical unification of the different semantics in the collective and individual token philosophies. That is, how can all these constructions be related within a unifying logical framework? We show in Section 4 that an appropriate partial equational logic can be a very good and economic logical framework in this regard, because (i) we can axiomatize the different categories of interest as categories of partial algebras for suitable equational specifications, and (ii) we can obtain each adjoint construction freely, in the sense that it follows—using general model-theoretic properties guaranteed by the logic—from a compact specification of the forgetful functor, which is in fact induced by a theory morphism between the underlying theories of the two categories of models under consideration. There are a number of such partial equational logics which are in some ways equivalent for the task, i.e., essentially all logics whose categories of models are the locally finitely presentable categories [12] (see [20, 25] for discussions of those logics). Among those equational logics we have chosen *partial membership equational logic* (**PMEqtl**) for the following reasons: (i) its support for subsorts, operator overloading, and membership axioms allows more compact specifications and often subsumes other formalisms as special cases; (ii) its theories have a tensor product construction which is very useful for our purposes; and (iii) there is a conservative extension to the framework of *total* membership equational logic [20] for which both a mechanized implementation [10] and well-developed theorem-proving techniques [4] are available. Hence, the proposed **PMEqtl** characterizations can be

used to support both execution and reasoning on the models they characterize. For example, there are mechanical tools for manipulating and simplifying expressions that represent concurrent computations and which can test for equality automatically. Moreover, working at a theory level leads to quite simple descriptions and allows one to use standard results to prove the existence of adjunctions, e.g., a theory morphism from a theory of programs (i.e., nets) to a theory of models induces an obvious forgetful functor (in the reverse direction) between the categories of algebras associated with the theories, which has a left adjoint providing the free construction of models for programs. It also allows one to compare different models.

The Collective Token Philosophy

The algebraic net theory developed under the *CTph* is well established [11, 21] and we are going to use it directly in order to present a detailed understanding of the relationships among the computational, algebraic, and logical interpretations of the *CTph*.

Starting with classical token-game semantics, many behavioral models for Petri nets have been proposed that follow the collective token philosophy. In fact, there are too many to be systematically reviewed here. Among these, however, there is a relatively recent study by van Glabbeek and Plotkin based on *configuration structures* [13]. Clearly inspired by the domains of configurations of *event structures* [36], configuration structures are simply collections of (multi)sets which, at the same time, represent the legitimate system states and the system dynamics, i.e., the transitions between these states.

One of the themes of this paper is a comparison of configuration structures with the algebraic model of [21], which adopts the *CTph* and provides a precise algebraic reinterpretation of yet another *CTph* model, namely, the *commutative processes* of Best and Devillers [3]. In particular, we observe that configuration structures are too abstract a model, i.e., that they make undesirable identifications of nets, and we conclude that strictly symmetric monoidal categories provide a superior model of net behavior.

To better illustrate the differences between the two semantic frameworks mentioned above, we adopt *concurrent transition systems* (CTS) as a bridge model. These provide a much more simplified version of higher dimensional transition systems [9]. In fact we choose these systems as the simplest bridge models to best convey our ideas.

Concurrent transition systems resemble configuration structures, but they are more expressive. They also draw on earlier very significant models, such as distributed transition systems [18], step and PN transition systems [26], and local event structures [15]. Moreover, the equivalence of the behavioral semantics of concurrent transition systems and the algebraic semantics of monoidal categories can be stated very concisely.

As a first result of our research, in this paper we show that Best–Devillers commutative processes, the algebraic model of [21] based on monoidal categories, and the concurrent transition system behavioral model all coincide, in the precise sense of being related by equivalences of categories. We also show how the behavioral model provided by configuration structures is too abstract, but it is related to all the above models by a natural transformation that characterizes the identification of inequivalent nets and behaviors caused by configuration structures.

We then observe that the construction of the concurrent operational behavior formulated via commutative processes exactly corresponds (1) at a semantic level, to the universal construction $\mathcal{T}(\_)$ of a strictly symmetric (strict) monoidal category[2] (the arrows of $\mathcal{T}(N)$ represent the commutative processes of the net N), and (2) at a logical level, to an adjunction induced by a suitable morphism between theories in **PMEqtl**.

Though the first observation is a well-known fact [11], the second point has only been outlined in [7] and is fully discussed in this paper. The features of **PMEqtl** (partiality, poset of sorts, membership assertions) offer a natural framework for the specification of categorical structures. For example, the sequential composition of arrows is a partial operation, and objects can be more easily modeled as a subsort of arrows, instead of using an injective embedding $id\_$. Moreover, a notion of tensor product for partial algebraic theories is used in [22] to obtain, among other things, an elegant definition of the theory of monoidal categories.

[2] Since we consider only strict monoidal categories, in the rest of the paper we omit the adjective *strict*.

The Individual Token Philosophy

When the *ITph* is assumed, it is more difficult to give a precise account of the relationships between the different interpretations. Building on the notion of *process* presented in [14], it has been shown that the semantics of a net can still be understood in terms of symmetric monoidal categories, but none of the proposed constructions 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. [11, 34]; see [24] for an overview). As recalled above, the functoriality (and hence the lifting of simulation morphisms on nets to the level of computation models) is, on the other hand, an essential property for guaranteeing the compositionality of the semantic framework.

More precisely, a simple variation on the Goltz–Reisig processes, called *concatenable processes*, is introduced in [11]. Concatenable processes admit sequential composition and yield a symmetric monoidal category $\mathcal{P}(N)$ for each net N, but their 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 may be the case that the concurrent behavior of N cannot be recovered from that of N', because equivalent computations in N should now be distinguished when they are simulated as computations in N' (see Example 1.1).

In [34] the situation is improved by introducing the notion of *strongly concatenable processes* as a slight refinement of concatenable processes, where a linear ordering is allocated to minimal and maximal places (whereas in concatenable processes, only instances of the same resource are ordered). Strongly concatenable processes can be expressed via a *pseudo-functor* $\mathcal{Q}(\_)$, i.e., a mapping between categories that preserves identities and composition up to a natural isomorphism. In particular, the pseudo-functor $\mathcal{Q}(\_)$ is a mapping of net morphisms to symmetric monoidal functors that strictly preserves identities. This construction is almost satisfactory, and indeed it extends to a functor from **Petri**, the category introduced in [21], to a quotient category of a suitable full subcategory of **SSMC**, the category of symmetric monoidal categories (as objects) and symmetric monoidal functors (as arrows); furthermore, it defines a left adjoint to a subcategory of such a quotient characterized by axiomatizing the role of transitions in $\mathcal{Q}(N)$.

The main difficulty in extending this nice algebraic framework to the *ITph* is that net morphisms in **Petri** allow one to replace two different resources a and b by two possibly nondisjoint multisets u and v in the target net in such a way that tokens in their union $u + v$ can be partitioned into u and v only up to a certain degree of ambiguity, whereas the *ITph* requires a precise correspondence between the instances of such resources. In [34] this is solved by including 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 $t_{\overline{u},\overline{v}} : \overline{u} \to \overline{v}$ is introduced (and suitably axiomatized) in the semantic model, for any linearizations $\overline{u}$ and $\overline{v}$ (i.e., strings of places) of multisets u and v. Although this settles the ambiguity problem, it gives a construction that, as mentioned above, is functorial only up to isomorphism, thus raising the need for a complex quotient operation.

We present an analogous construction centered on the notion of a *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 fixed (instead of all) linear implementation $t_{\overline{u},\overline{v}} : \overline{u} \to \overline{v}$ for some linearizations $\overline{u}$ and $\overline{v}$ 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 nonterminal symbols, and strings can be permuted before performing any step, i.e., ordinary grammars would only generate monoidal categories, without symmetries. Although abandoning multisets might at first appear unnatural to net enthusiasts, our formal approach to the *ITph* benefits from 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 for the implemented net given by the construction $\mathcal{Q}(\_)$ can be recovered from any pre-net implementation.

▷ The algebraic semantics of pre-nets can also be rephrased in the logical framework of **PMEqtl**.

This means that the investigation of the behavioral, algebraic, and logical aspects of PT nets already developed for the *CTph* can be successfully extended to the individual token approach, using pre-nets rather than PT nets to accomplish a better categorical construction which fully supports an algebraic viewpoint.

By using **PMEqtl** techniques, we are able to formalize the construction as the free construction associated with a straightforward theory morphism from the theory of pre-nets to the theory of symmetric monoidal categories. Finally, theory morphisms can be exploited to reconcile the *ITph* view of pre-nets and the *CTph* view of PT nets, in the sense that, starting from the category of pre-nets, one can either view them as PT nets and take the *CTph* semantics or take the *ITph* semantics and then forget about causality information, always obtaining the same result. Note that, from the property of theory morphisms, this result holds not only at the object level (as was shown in [11] for the constructions $\mathcal{T}(\_)$ and $\mathcal{P}(\_)$), but also at the arrow level, resulting in a commuting square of adjunctions (see Proposition 4.10).

Origin and structure of the paper. This paper improves and slightly extends our earlier research in [7, 8], regarding both collective and individual token philosophies, thus providing a structured and uniform presentation of all concepts and details of both philosophies.

In Section 1 we look at the basic definitions of PT nets, explain the distinction between their two computational interpretations (individual vs. collective), and summarize the approaches presented in the literature to accommodate these two views. The corresponding two semantics given in the literature are presented in Sections 1.1 and 1.2, respectively.

Section 2 compares configuration structures, monoidal categories, and concurrent transition systems. Section 3 introduces pre-nets, defining their algebraic semantics and the relationship with ordinary PT nets. In Section 4 we employ **PMEqtl** to formalize the logical aspects of both PT net and pre-net semantics, reconciling the two views. The basics of partial membership equational logic and the tensor product construction of theories are covered in the Appendix to make the paper practically self-contained.

1. PT NETS

Place/transition Petri nets, the most widespread flavor of Petri nets, are graphs with distributed states described by (finite) distributions of resources (tokens) in places. These are usually called *markings*, and are represented as multisets $u : S \to \mathbb{N}$, where $u(a)$ indicates the number of tokens that place a carries in u. To some extent, places can be seen as the different types of usable resources.

We shall use $\mu(S)$ to indicate the set of finite multisets on S, i.e., multisets that yield a zero on all but finitely many $a \in S$. Multiset union, which we denote by $+$ with $(u + v)(a) = u(a) + v(a)$ for any place a, makes $\mu(S)$ a free commutative monoid on S (whose unit is the empty multiset, denoted by $\emptyset$).

DEFINITION 1.1. A *place/transition Petri net* (*PT net*) is a tuple $N = (\partial_0, \partial_1, S, T)$, 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.

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 overall state. This is made explicit in the following definition, where we shall indicate multiset inclusion and difference by, respectively, $\subseteq$ and $-$, where $u \subseteq v$ if $u(a) \leq v(a)$ for any place a and $v - u$ is only defined when $u \subseteq v$ and returns the unique multiset u' such that $v = u + u'$.

DEFINITION 1.2. Let u and v be markings, and let $X : T \to \mathbb{N}$ be a finite multiset of transitions in 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., $\sum_{t \in T} X(t) \cdot \partial_0(t) \subseteq u$, and

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

A *step sequence* from u_0 to u_n is a sequence $u_0\, [X_1\rangle\, u_1\, [X_2\rangle\, u_2 \ldots u_{n-1}\, [X_n\rangle\, u_n$, with $n \geq 0$ (if $n = 0$ then the step sequence is *empty*).

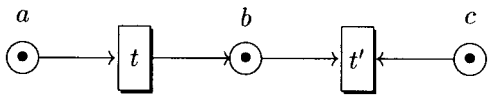

FIGURE 1

PT nets are often taken together with an initial state: a *marked* PT net N is a PT net $(\partial_0, \partial_1, S, T)$ together with an *initial marking* $u_0 \in \mu(S)$. 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 *net morphism* from $N = (\partial_0, \partial_1, S, T)$ to $N' = (\partial'_0, \partial'_1, S', T')$ 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 $f_p(u_0) = u'_0$.

To shorten the notation we will omit the subscripts from morphism components. We use **Petri** (respectively **Petri**$_*$) to indicate the category of (respectively marked) PT nets and their morphisms, with the obvious componentwise composition of arrows.

Several different proposals of net morphisms can be found in the literature that give rise to different categories of nets, e.g., [1, 5, 6, 26]. Comparing these categories to **Petri** is not so straightforward, and we cannot address here the question as to whether and to what extent our results would fit in with these choices. We limit ourselves to remarking that our morphisms follow the two main algebraic approaches to nets—namely Winskel's [35] and Meseguer-Montanari's [21]—and their strength lies in their naturality, which stems directly from viewing nets as algebras and graphs at the same time.

To compare the effects of collective and individual token philosophies on observing causal relations between fired transitions, let us consider the example in Fig. 1, adapted from [13]. (As usual, boxes stand for transitions, circles for places, dots for tokens, and weighted 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, whereas in the second case the two firings are independent. In the *CTph*, instead, the two firings are always taken as concurrent, because the firing of t does not affect the enabling condition of t'.

1.1. *Collective Token Semantics*

Several interesting aspects of Petri net theory can be profitably developed within category theory; see, e.g., [5, 21, 35]. We focus on the approach initiated in [21] (other relevant references are [11, 23, 24, 33, 34]) which reveals the monoidal structure of Petri nets under the operation of parallel composition. In [11, 21] it is shown how the sets of transitions can be endowed with appropriate algebraic structures in order to capture some basic constructions on nets. In particular, the commutative processes of Best and Devillers [3]—or equivalently, step and firing sequences up to diamond transformation equivalence—which represent the natural behavioral model for PT nets under the collective token philosophy, can be characterized by adding a functorial sequential composition on the monoid of steps, thus giving a strictly symmetric monoidal category $\mathcal{T}(N)$ (it is called strictly symmetric because the monoidal operation is commutative).

Denoting by **CMonCat** the category of strictly symmetric monoidal categories (as objects) and monoidal functors (as arrows), $\mathcal{T}(\_)$ extends to a functor from **Petri** to **CMonCat**.

Remark. All the functors that we present, from a category of nets to a category of models (viewed as suitable monoidal categories) are faithful but not full, as is common in the construction of free models. For example, the functor $\mathcal{T}(\_)$ is faithful but not full, since in **CMonCat** transitions can be mapped to computations (giving the so-called *implementation morphisms* [21]), whereas in **Petri** transitions can only be mapped to transitions.

For each net N, the category $\mathcal{T}(N)$ can be inductively defined by the inference rules in Table 1, modulo the axioms in Table 2, which state that $\mathcal{T}(N)$ is a strictly symmetric monoidal category, since

TABLE 1

Inference Rules for $\mathcal{T}(N)$

$\dfrac{u \in \mu(S_N)}{id_u : u \to u \in \mathcal{T}(N)}$	$\dfrac{t \in T_N,\ \partial_0(t) = u,\ \partial_1(t) = v}{t : u \to v \in \mathcal{T}(N)}$
$\dfrac{\alpha : u \to v,\ \beta : u' \to v' \in \mathcal{T}(N)}{\alpha \oplus \beta : u + u' \to v + v' \in \mathcal{T}(N)}$	$\dfrac{\alpha : u \to v,\ \beta : v \to w \in \mathcal{T}(N)}{\alpha ; \beta : u \to w \in \mathcal{T}(N)}$

they must be satisfied by all arrows α, α', β, β', δ (that can be composed correctly) and all multisets u and v.

The intuition here is that objects are multisets of places, arrows are step sequences, and arrow composition is their concatenation, whereas the monoidal operator $\_ \oplus \_$ allows such sequences to be composed in parallel. It turns out that this algebraic structure describes precisely the processes à la Best and Devillers.

PROPOSITION 1.1 (cf. [21]). *The presentation of $\mathcal{T}(N)$ given above precisely characterizes the algebra of commutative processes of the PT net N, i.e., the (equivalence classes of) arrows in $\mathcal{T}(N)$ are in bijective correspondence with the commutative processes of N, and the correspondence commutes with the operation of the algebra, defining a homomorphism.*

In other words, if we denote by $\mathcal{CP}(N)$ the monoidal category whose objects are multisets of places and whose arrows are the commutative processes of N, then $\mathcal{CP}(N)$ and $\mathcal{T}(N)$ are isomorphic in **CMonCat**. Moreover, if one considers the full subcategory of **CMonCat** consisting of categories whose monoids of objects are freely generated, then there is an obvious forgetful functor from this subcategory to **Petri**, which is a right adjoint to $\mathcal{T}(\_)$.

When we are interested in marked nets, by analogy with $\mathbf{Petri}_*$, we take a pointed category $(\mathbf{C}, c_0)$ to be a category $\mathbf{C}$ together with a distinguished object $c_0 \in \mathbf{C}$. Similarly, a pointed functor from $(\mathbf{C}, c_0)$ to $(\mathbf{D}, d_0)$ is a functor $F : \mathbf{C} \to \mathbf{D}$ which maps the distinguished object c_0 to the distinguished object d_0. Then, when $\mathbf{CMonCat}_*$ is used to denote the category of pointed strictly symmetric monoidal categories and their pointed functors, the previous construction extends immediately to a functor $\mathcal{T}_*(N) : \mathbf{Petri}_* \to \mathbf{CMonCat}_*$, such that for $N = (\partial_0, \partial_1, S, T, u_0)$ a marked PT net, then

$$\mathcal{T}_*(N) = (\mathcal{T}(\partial_0, \partial_1, S, T), u_0).$$

As an alternative proposal for representing the behavior of nets according to the *CTph*, in the same paper where they introduce the distinction between collective token and individual token philosophy, van Glabbeek and Plotkin define *configuration structures*. These are structures inspired by event structures [36] whose dynamics are uniquely determined by an explicitly given set of possible configurations of the system. However, the structures that they end up associating with nets are not exactly configuration structures. They enhance them in two ways: first, by considering multisets instead of sets of occurrences, and second, by using an explicit transition relation between configurations. While the first point can be handled easily, as we do below, the second one seems to compromise the basic ideas underlying the framework and to show that configuration structures do not offer a faithful representation of the concurrent behavior of nets.

DEFINITION 1.4. A *configuration structure* is given by a set E and a collection C of finite multisets over the set E. The elements of E are called *events* and the elements of C *configurations*.

TABLE 2

Axioms for $\mathcal{T}(N)$

Neutral	$id_\varnothing \oplus \alpha = \alpha$	
Commutativity	$\alpha \oplus \beta = \beta \oplus \alpha$	
Associativity	$(\alpha \oplus \beta) \oplus \delta = \alpha \oplus (\beta \oplus \delta)$	$(\alpha ; \beta) ; \delta = \alpha ; (\beta ; \delta)$
Identities	$\alpha ; id_v = \alpha = id_u ; \alpha$	$id_u \oplus id_v = id_{u+v}$
Functoriality	$(\alpha ; \beta) \oplus (\alpha' ; \beta') = (\alpha \oplus \alpha') ; (\beta \oplus \beta')$	

The idea is that an event is an occurrence of an action the system may perform and that a configuration X represents a state of the system, which is determined by the collection X of occurred events. The set C of admissible configurations yields a relation representing how the system can evolve from one state to another.

DEFINITION 1.5. Let (E, C) be a configuration structure. For X, Y in C we write $X \to Y$ if

(1) $X \subset Y$,

(2) $Y - X$ is finite,

(3) for any multiset Z such that $X \subset Z \subset Y$, we have $Z \in C$.

The relation $\to$ is called the *step transition relation*.

Intuitively, $X \to Y$ means that the system can evolve from state X to state Y by performing the events in $Y - X$ concurrently. To stress this we shall occasionally write $X \xrightarrow{L} Y$, with $L = Y - X$. Observe that the last condition essentially means that the events in $Y - X$ can be performed concurrently if and only if they can be performed in any order. In our opinion, this requirement embodies an interleaving-oriented view, as it reduces concurrency to nondeterminism. As we explain below, we view this as the main weakness of configuration structures.

In the following definition we slightly refine the notion of net configuration proposed in [13], since the original definition may improperly include multisets of transitions that cannot be fired from the initial marking.

DEFINITION 1.6 (From PT nets to configuration structures). Let $N = (\partial_0, \partial_1, S, T, u_0)$ be a marked PT net. A finite multiset X of transitions is called *fireable* if a partition $X_1, \ldots, X_n$ of X exists such that $u_0\ [X_1\rangle\ u_1 \ldots u_{n-1}\ [X_n\rangle\ u_n$ is a step sequence. A *configuration* of N is a fireable multiset X of transitions. The configuration structure associated with N is $cs(N) = (T, C_N)$, where C_N is the set of configurations of N.

It follows that for each configuration X the function $u_X : S \to \mathbb{Z}$ given by

$$u_X = u_0 + \sum_{t \in T} X(t) \cdot \partial_1(t) - \sum_{t \in T} X(t) \cdot \partial_0(t)$$

is a (reachable) marking; i.e., $0 \leq u_X(a)$ for all $a \in S$. Moreover, if X is a configuration and $u_X\ [U\rangle\ v$, then $X + U$ is also a configuration and $v = u_{(X+U)}$.

Generally speaking, if N is a *pure net*, i.e., a net with no self-loops, then $cs(N)$ can be considered a reasonable semantics for N. Otherwise, as observed also in [13], it is not a good idea to reduce N to $cs(N)$. Consider, for example, the marked nets N and M of Fig. 2. They have very different behaviors. In fact, in N the actions t_0 and t_1 are concurrent, whereas in M they are mutually exclusive. However, since in M any interleaving of t_0 and t_1 is possible, the diagonal $\emptyset \to \{t_0, t_1\}$ sneaks into the structure by definition. As a result, both N and M yield the configuration structure represented in Fig. 3, even though $\{t_0, t_1\}$ is *not* an admissible step for M. The limit case is the marked net consisting of a single self-loop: readers can check for themselves that, according to $cs(\_)$, it can fire arbitrarily large steps.

These problems have prompted us to look for a similar semantic framework that represents net behaviors more faithfully than configuration structures. The key observation is that there is nothing

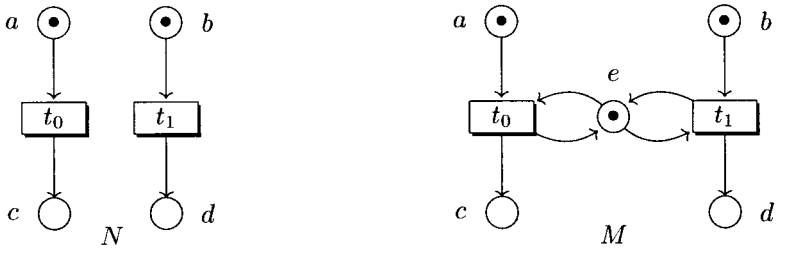

FIG. 2. The nets N and M of our running example.

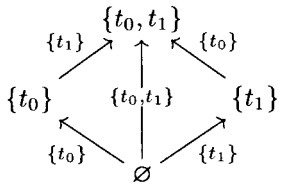

FIG. 3. The configuration structure $cs(N) = cs(M)$ for the nets N and M.

wrong with the assumption that if a step involving many parallel actions can occur in a certain state, then all the possible interleaving sequences of those actions can also occur from that state. The problematic bit is assuming the inverse implication, because it actually reduces concurrency to nondeterminism and makes the set of configurations uniquely determine the transition relation. Our proposed solution is the notion of concurrent transition systems, which will be discussed in Section 2.

1.2. *Individual Token Semantics*

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 that carry some explicit causal information between transition firings (called *events*). This corresponds to an abstract view of processes as posets whose elements are labeled by transitions of the net [27, 35, 36], i.e., as *pomsets*. Concretely, such computations are represented by suitable, structure-preserving maps from a special class of nets in 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.

DEFINITION 1.7. A *process net* (also called *deterministic occurrence 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 $t_0 \neq t_1 \in T$, $\partial_i(t_0) \cap \partial_i(t_1) = \emptyset$, for $i = 0, 1$.

DEFINITION 1.8. 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).

Two processes $\pi : P \to N$ and $\pi' : P' \to N$ are *isomorphic*, and thus identified, if a net isomorphism $\psi : P \to P'$ exists 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 we can similarly define $\pi(D(P))$ as those resources available in N when the execution of π is completed.

Two processes for the (marked) net of Fig. 1 are represented by the mappings π_0 and π_1 from the process nets P_0 and P_1 in Fig. 4, where dotted arrows show the images of places and transitions (for readability, we omit the names of the elements of P_0 and P_1).

Since processes represent computations, it is natural to seek a notion of a sequential composition of those processes π and π' with $\pi(D(P)) = \pi'(O(P'))$, that is, π' starts from the marking π terminates

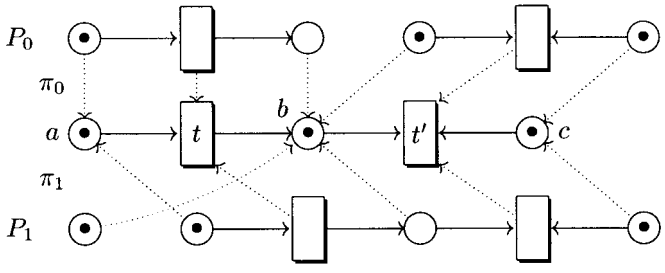

FIG. 4. Two processes P_0 and P_1 modeling the concurrent and the sequential execution of t and t', respectively.

in. There can be 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 [11] by imposing a total ordering of origins and destinations that are instances of the same place.

DEFINITION 1.9. Given a labeling 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 $\beta_y : l^{-1}(y) \to \{1, \ldots, |l^{-1}(y)|\}$.

The idea is that Y is the set of places of a given net N, while 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 1.10. 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 labeling 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 monoidal category whose symmetries are given by concatenable processes consisting only of places—see [11] for the formal definitions of such operations.

We saw in Section 1.1 that the definition of commutative processes can be nicely expressed via a categorical adjunction. Under the *ITph*, one might expect some analogous results to hold between symmetric monoidal categories and concatenable processes. We recall in fact that symmetric monoidal categories possess some auxiliary arrows called *symmetries* that can model the possible reorganization of minimal and maximal places of a process. It is worth noting that, in concatenable processes, the ordering of minimal and maximal places is only imposed on instances of the same place. We recall here the definition of the category $\mathcal{P}(N)$ introduced in [11] and finitely axiomatized in [33].

DEFINITION 1.11. Let N be a PT net. The category $\mathcal{P}(N)$ is the monoidal quotient of the free symmetric monoidal category $\mathcal{F}(N)$ generated by N, modulo the axioms

$$\begin{aligned} \gamma_{a,b} &= id_a \otimes id_b && \text{if } a, b \in S_N, \text{ and } a \neq b \\ s; t; s' &= t && \text{if } t \in T_N \text{ and } s, s' \text{are symmetries,} \end{aligned}$$

where $\gamma_{\_,\_}$, $id_\_$, $\_\otimes\_$, and $\_;\_$ are, respectively, the symmetry isomorphism, the identities, the tensor product, and the composition of $\mathcal{F}(N)$ (see Table 3).

We note that in $\mathcal{F}(N)$ the tensor product is not commutative and the symmetries satisfy the naturality axiom

$$(\alpha \otimes \alpha'); \gamma_{v,v'} = \gamma_{u,u'}; (\alpha' \otimes \alpha) \tag{1}$$

for all arrows $\alpha : u \to v$ and $\alpha' : u' \to v'$ and also the MacLane coherence axioms [19]

$$\gamma_{u,v}; \gamma_{v,u} = id_{u+v} \tag{2}$$

$$\gamma_{u,v+v'} = (\gamma_{u,v} \otimes id_{v'}); (id_v \otimes \gamma_{u,v'}) \tag{3}$$

TABLE 3

Inference Rules for $\mathcal{F}(N)$

$\dfrac{u \in \mu(S_N)}{id_u : u \to u \in \mathcal{F}(N)}$	$\dfrac{u, u' \in \mu(S_N)}{\gamma_{u,u'} : u + u' \to u' + u \in \mathcal{F}(N)}$	$\dfrac{t \in T_N,\ \partial_0(t) = u,\ \partial_1(t) = v}{t : u \to v \in \mathcal{F}(N)}$
$\dfrac{\alpha : u \to v,\ \beta : u' \to v' \in \mathcal{F}(N)}{\alpha \otimes \beta : u + u' \to v + v' \in \mathcal{F}(N)}$		$\dfrac{\alpha : u \to v,\ \beta : v \to v' \in \mathcal{F}(N)}{\alpha; \beta : u \to v' \in \mathcal{F}(N)}$

TABLE 4

Axioms for $\mathcal{F}(N)$

Neutral	$id_\varnothing \otimes \alpha = \alpha = \alpha \otimes id_\varnothing$	
Associativity	$(\alpha \otimes \beta) \otimes \delta = \alpha \otimes (\beta \otimes \delta)$	$(\alpha;\beta);\delta = \alpha;(\beta;\delta)$
Identities	$\alpha;\ id_v = \alpha =\ id_u;\alpha$	$id_u \otimes\ id_v =\ id_{u+v}$
Functoriality	$(\alpha;\beta) \otimes (\alpha';\beta') = (\alpha \otimes \alpha');(\beta \otimes \beta')$	
Naturality	$(\alpha \otimes \alpha');\gamma_{v,v'} = \gamma_{u,u'};(\alpha' \otimes \alpha)$	
Coherence	$\gamma_{u,v+v'} = (\gamma_{u,v} \otimes id_{v'});(id_v \otimes \gamma_{u,v'})$	$\gamma_{u,v};\gamma_{v,u} = id_{u+v}$

for all markings u, v, and v'. Notice that from these axioms the equation $\gamma_{u,\varnothing} = id_u$ can also be easily inferred, since the tensor product of $\mathcal{F}(N)$ is strict. For the reader's convenience, the axioms of $\mathcal{F}(N)$ are gathered in Table 4.

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

EXAMPLE 1.1. Consider the nets N and N' shown in Fig. 5 and the net morphism $f: N \to N'$ such that $f(t_i) = t_i'$, $f(a_i) = a'$, and $f(b_i) = b_i'$, for $i \in [0, 1]$. 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 1.11), 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 onto the same place a' via a net morphism, then it should be the case that $\gamma_{a,b} = id_a \otimes id_b$ is mapped onto $\gamma_{a',a'} \neq id_{a'} \otimes id_{a'}$ via a monoidal functor that extends such a net morphism, which is not possible. Therefore, concatenable processes are still not completely satisfactory, because several net morphisms cannot be lifted to a behavior level.

To improve such a situation, the notion of strongly concatenable processes was introduced in [34], where a total order is imposed on origins and destinations and not only on the instances of the same place. Strongly concatenable processes also admit a sequential and parallel composition, yielding a symmetric monoidal category.

DEFINITION 1.12. 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, while the labeling functions $\ell_O: O(P) \to \{1, \ldots, |O(P)|\}$ and $\ell_D: D(P) \to \{1, \ldots, |D(P)|\}$ are bijections.

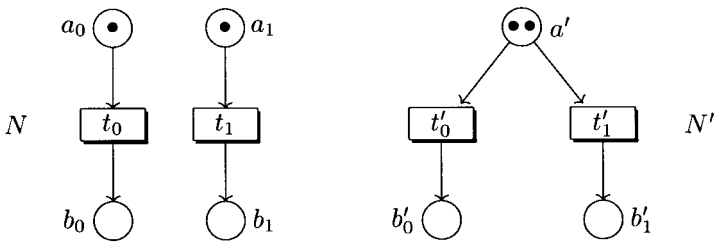

FIG. 5. The PT nets N and N' considered in Example 1.1.

The *pseudo-functorial* construction $\mathcal{Q}(\_) : \mathbf{Petri} \to \mathbf{SSMC}$ of [34], recovering strongly concatenable processes, improves the situation, in the sense that it strictly preserves identities and composition only up to a monoidal natural isomorphism (details in [34]).

DEFINITION 1.13. Let N be a PT net. The category $\mathcal{Q}(N)$ is obtained from the symmetric monoidal category freely generated from the places of N and, for each transition $t : u \to v$ of N, arrows $t_{\overline{u},\overline{v}} : \overline{u} \to \overline{v}$ for each pair of linearizations (as strings) of the pre- and post-sets of t, by quotienting modulo the axiom

$$s; t_{\overline{u},\overline{v}}; s' = t_{\overline{u}',\overline{v}'} \quad \text{for } s : \overline{u}' \to \overline{u} \text{ and } s' : \overline{v} \to \overline{v}' \text{ symmetries.} \tag{4}$$

Our point in this paper is that functoriality is lacking in these constructions because PT nets rely on a "state as multiset" paradigm, whereas the *ITph* imposes a distinction between different instances of the same resource. Hence, as a solution to this problem, we propose a refined view of nets, so that the associated notion of morphism behaves better w.r.t. the construction of the category of processes.

The paradigm we propose is called *pre-net* and is presented in Section 3.

2. A COMPARISON OF COLLECTIVE TOKEN SEMANTICS

The analysis of configuration structures suggests seeking a model that enforces the existence of all appropriate interleavings of steps, without allowing this to completely determine the set of transitions. Several such models appear in the literature. Among those that inspired us the most were distributed transition systems [18], step transition systems [26], PN transition systems [26], and higher dimensional transition systems [9]. Also closely related are the local event structures of [15], a model that extends event structures (rather than transition systems) by allowing the firing of sets (but not multisets) of events. Drawing on all of these, here we have chosen the simplest definition to suit our current aims.

DEFINITION 2.1. A *concurrent transition system* (CTS) is a structure $H = (S, L, \mathit{trans}, s_0)$, where S is a set of states, L is a set of actions, $s_0 \in S$ is the initial state, and $\mathit{trans} \subseteq S \times (\mu(L) - \{\emptyset\}) \times S$ is a set of transitions such that:

(1) if $(s, U, s_1), (s, U, s_2) \in \mathit{trans}$, then $s_1 = s_2$,

(2) if $(s, U, s') \in \mathit{trans}$ and U_1, U_2 is a partition of U, then $v_1, v_2 \in S$ exist such that (s, U_1, v_1), (s, U_2, v_2), (v_1, U_2, s'), $(v_2, U_1, s') \in \mathit{trans}$.

Condition (1) above states that the execution of a multiset of labels U in a state s deterministically leads to a different state, as this reflects our view of actions as transitions. The second condition guarantees that all the possible interleavings of the actions in U are possible paths from s to s' if $(s, U, s') \in \mathit{trans}$. Notice that, by (1), the states v_1 and v_2 of (2) are uniquely determined.

We formalize the idea that different paths which are different interleavings of the same concurrent step can be considered equivalent.

DEFINITION 2.2. A *path* in a CTS is a sequence of contiguous transitions

$$(s, U_1, s_1)(s_1, U_2, s_2) \cdots (s_{n-1}, U_n, s_n).$$

A *run* is a path that originates from the initial state.

DEFINITION 2.3. Given a CTS H, *adjacency* is the least reflexive, symmetric, binary relation $\leftrightarrow_H$ on the paths of H which is closed under path concatenation and such that $(s, U_1, s_1)(s_1, U_2, s_2) \leftrightarrow_H (s, U_1 + U_2, s_2)$. Then, the *homotopy* relation $\leftrightarrows_H$ on the paths of H is the transitive closure of $\leftrightarrow_H$. The equivalence classes of runs of H with respect to the homotopy relation are called *computations*. The computation associated with a generic run π is denoted by $[\pi]_{\leftrightarrows_H}$.

In order to simplify our exposition, we now refine the notion of concurrent transition system so as to be able to associate the same multiset of actions with each path between two states. As we shall see, such transition systems enjoy interesting properties.

DEFINITION 2.4. A CTS is *uniform* if all its states are reachable from the initial state and the unions of the actions along any two cofinal runs yield the same multiset, where cofinal means ending in the same state.

In a uniform CTS $H = (S, L, \mathit{trans}, s_0)$ each state s can be associated with the multiset of actions on any run to s. Precisely, we shall use ς_s to indicate $\sum_{i=1}^{n} U_i$, for

$$(s_0, U_1, s_1)(s_1, U_2, s_2) \cdots (s_{n-1}, U_n, s)$$

a run of H.

The *length* of any run π ending in s is denoted by $\mathit{len}(\pi)$ and is defined as equal to the cardinality of ς_s. By condition (2) in Definition 2.1, it follows that for each run π there always exists at least one homotopic run consisting of exactly $\mathit{len}(\pi)$ sequential transitions. Observe also that uniform CTSs must be acyclic, because any cycle $(s, U_0, s_1) \cdots (s_n, U_n, s)$ would imply the existence of runs to s carrying different actions. In the rest of the paper, we shall consider *only* uniform concurrent transition systems.

Introducing the natural notion of computation-preserving morphism for CTSs, we define a category of uniform CTSs. In the following, for functions $f: A \to B$, we denote by $f^{\mu}: \mu(A) \to \mu(B)$ the unique multiset homomorphism extending f; i.e., $f^{\mu}(X)(b) = \sum_{a \in f^{-1}(b)} X(a)$.

DEFINITION 2.5. For H_1 and H_2 CTS, a *morphism* from H_1 to H_2 consists of a map $f: S_1 \to S_2$ that preserves the initial state and a function $\alpha: L_1 \to L_2$ and such that $(s, U, s') \in \mathit{trans}_1$ implies $(f(s), \alpha^{\mu}(U), f(s')) \in \mathit{trans}_2$.

We denote by **CTS** the category of uniform CTSs (as objects) and their morphisms (as arrows).

DEFINITION 2.6 (From PT Nets to CTSs). Let $N = (\partial_0, \partial_1, S, T, u_0)$ be a marked PT net. The concurrent transition system associated with N is

$$\mathit{ct}(N) = (M_N, T, \mathit{trans}_N, \varnothing),$$

where M_N is the set of fireable multisets of transitions of N and

$$(X, U, X') \in \mathit{trans}_N \text{ if and only if } u_X \, [U\rangle \, u_{X'}.$$

(Recall that $u_X: S \to \mathbb{Z}$ is by definition a reachable marking.)

Although this construction is formally very close to the one proposed for configuration structures, the difference is that a CTS does not enforce diagonals to fill the squares: these are introduced if and only if the associated step is actually possible (see Fig. 6).

We give a precise categorical characterization of the representations of nets in the CTS framework in Section 2.1. For the time being, we note the following.

PROPOSITION 2.1. *$\mathit{ct}(\_)$ is a functor from* $\mathbf{Petri}_*$ *to* **CTS**.

Proof. The mapping $\mathit{ct}(\_)$ is already defined on objects. Let $f: N \to N'$ be a marked net morphism. Then $\mathit{ct}(f)$ is defined as f on the labels of $\mathit{ct}(N)$ and as f_{t}^{μ} on the states of $\mathit{ct}(N)$. In fact it is easy to

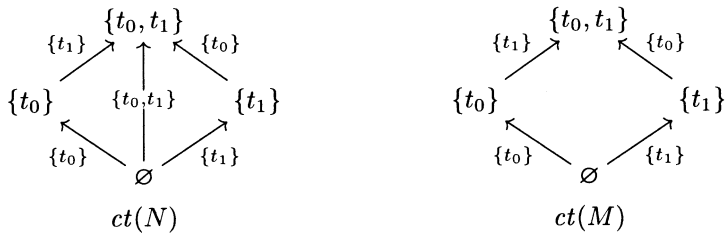

FIG. 6. The CTS $\mathit{ct}(N)$ and $\mathit{ct}(M)$ for the nets N and M in Fig. 2.

verify that the image through f_t^{μ} of each fireable multiset X in N is also a fireable multiset in N'. It is obvious that $ct(\_)$ preserves morphism composition and identities. ■

Although all cofinal runs of a CTS carry the same multiset of actions, not all such runs are homotopic, i.e., they do not necessarily represent the same computation. The enforcement of this condition is the purpose of the next definition.

DEFINITION 2.7. An *occurrence* concurrent transition system is a concurrent transition system H in which all pairs of *cofinal* transitions

$$(s_1, U_1, s), (s_2, U_2, s) \in trans_H$$

are the final steps of *homotopic* runs.

It can be shown that the previous definition implies the following property.

PROPOSITION 2.2. *All cofinal runs of an occurrence CTS are homotopic.*

Proof. We proceed by contradiction. Let π_1 and π_2 be two nonhomotopic cofinal runs of an occurrence CTS and let n be their minimum length, with π_1 and π_2 such that for any two nonhomotopic cofinal runs π_1' and π_2' whose minimum length is m we have $n \leq m$. We can assume, without loss of generality, that $len(\pi_1) = n$. If $n = 0$ then π_1 and π_2 are empty paths, thus contradicting the hypothesis. If $n > 0$ then $\pi_1 = \pi_1'(s_1, \{l_1\}, s)$ and $\pi_2 = \pi_2'(s_2, \{l_2\}, s)$ for some shorter runs π_1' and π_2'. If $s_1 = s_2$ then $l_1 = l_2$ and π_1' is homotopic to π_2', because the minimum of their lengths is $n - 1$. But this would imply that $\pi_1 \leftrightarrows \pi_2$, thus contradicting the hypothesis. Hence we must have $s_1 \neq s_2$. By the definition of occurrence concurrent transition systems, we know that two runs π_3 and π_4 exist leading to s_1 and s_2 respectively, such that $\pi_3(s_1, \{l_1\}, s)$ is homotopic to $\pi_4(s_2, \{l_2\}, s)$ and thus $len(\pi_4) = len(\pi_3)$. Moreover, it must be the case that $\pi_1' \not\leftrightarrows \pi_3$, because $len(\pi_1') < len(\pi_1)$. Therefore, $len(\pi_3) = len(\pi_1') = n - 1$ and $\pi_2' \leftrightarrows \pi_4$, otherwise we would contradict the hypothesis of minimality for n. From the closure properties of $\leftrightarrows$ we have of course that $\pi_1'(s_1, \{l_1\}, s) \leftrightarrows \pi_3(s_1, \{l_1\}, s)$ and $\pi_2'(s_2, \{l_2\}, s) \leftrightarrows \pi_4(s_2, \{l_1\}, s)$. By the transitivity of $\leftrightarrows$ we therefore have $\pi_1 \leftrightarrows \pi_2$, contradicting the hypothesis and concluding the proof. ■

Figure 7 illustrates the proof of Proposition 2.2.

We shall use **oCTS** to indicate the full subcategory of **CTS** consisting of occurrence CTSs. Clearly, a uniform CTS can be unfolded into an occurrence CTS.

DEFINITION 2.8 (From CTSs to occurrence CTSs). Let $H = (S, L, trans, s_0)$ be a concurrent transition system. Its *unfolding* is the occurrence concurrent transition system $\mathcal{O}(H) = (S', L, trans', \epsilon)$, where S' is the collection of computations of H and

$$trans' = \{([\pi]_{\leftrightarrows}, U, [\pi']_{\leftrightarrows}) \mid \exists s, s' \in S,\ [\pi']_{\leftrightarrows} \in S',\ \pi' \leftrightarrows_H \pi(s, U, s')\}.$$

This makes **oCTS** a *coreflective* subcategory of **CTS**.

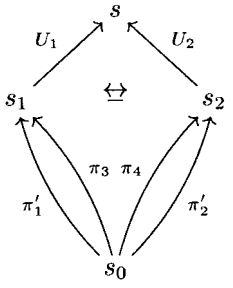

FIG. 7. Representation of the runs considered in the proof of Proposition 2.2.

PROPOSITION 2.3. *The construction $\mathcal{O}(\_)$ extends to a right adjoint to the inclusion of* **oCTS** *in* **CTS**.

Proof. For H a concurrent transition system, we take the CTS morphism $\varepsilon_H : \mathcal{O}(H) \to H$ that maps each $[\pi]_{\leftrightarrows} \in S_{\mathcal{O}(H)}$ to its final state $s \in S_H$. It is easy to verify that this forms the counit of the adjunction. ■

2.1. *Concurrent Transition Systems and Monoidal Categories*

In this section we look at the faithfulness of the CTS representation of nets, as given in Definition 2.6, with respect to the collective token philosophy. To accomplish this aim, we show that both the $ct(\_)$ and the $\mathcal{T}(\_)$ constructions yield two equivalent categories of net behaviors.

Regarding the monoidal approach, the obvious choice is to take the comma category of $\mathcal{T}(N)$ with respect to the initial marking, thus yielding a category whose objects are the commutative processes of N from its initial marking. An arrow from process p to process q is then the unique commutative process r such that $p; r = q$ in $\mathcal{T}(N)$. We denote the resulting category by $(u_0 \downarrow \mathcal{T}(N))$.

An analogous construction can be defined starting from $ct(N)$. The first step is to observe that the paths of a generic CTS under the homotopy relation define a category.

DEFINITION 2.9. For $H = (S, L, \mathit{trans}, s_0)$ a CTS, we define the *category of computations* of H to be the category $\mathcal{C}(H)$ whose

- ▷ *objects* are computations $[\pi]_{\leftrightarrows}$ of H,
- ▷ *arrows* are the homotopy equivalence classes of paths in H such that

$$[\psi]_{\leftrightarrows} : [\pi]_{\leftrightarrows} \to [\pi']_{\leftrightarrows} \quad \text{iff } \pi' \leftrightarrows_H \pi\psi,$$

- ▷ *composition* is defined as the homotopy class of path concatenation, i.e.,

$$[\psi]_{\leftrightarrows}; [\psi']_{\leftrightarrows} = [\psi\psi']_{\leftrightarrows},$$

- ▷ *identity arrow* at $[\pi]_{\leftrightarrows}$ is $\epsilon_{[\pi]_{\leftrightarrows}}$, the homotopy class of the empty path at the final state of π.

This construction extends easily to a functor $\mathcal{C}(\_)$ from **CTS** to **Cat**, the category of (small) categories and functors, yielding a functor $\mathcal{C}(ct(\_))$ from $\mathbf{Petri}_*$ to **Cat**. Observe also that $\mathcal{C}(\_)$ factors through $\mathcal{O} : \mathbf{CTS} \to \mathbf{oCTS}$ via the obvious path construction.

THEOREM 2.1. *Let N be a marked PT net with initial marking u_0. Then, the categories $\mathcal{C}(ct(N))$ and $(u_0 \downarrow \mathcal{T}(N))$ are isomorphic.*

Proof. We sketch the definition of functors

$$\mathsf{F} : (u_0 \downarrow \mathcal{T}(N)) \to \mathcal{C}(ct(N)) \quad \text{and} \quad \mathsf{G} : \mathcal{C}(ct(N)) \to (u_0 \downarrow \mathcal{T}(N))$$

that are inverses to each other. The functor F maps an object of the comma category to the homotopy class of any of the object's interleavings (which is well defined because of the diamond equivalence of [3]). Its action on morphisms is analogous.

On the other hand, for a computation $[\pi]_{\leftrightarrows}$ in $\mathcal{C}(ct(N))$, starting from the initial marking we can uniquely determine the corresponding arrow on $\mathcal{T}(N)$ and therefore define the action of G on both objects and arrows. ■

The categories of computations for the concurrent transition systems associated with nets N and M in Fig. 2 are shown in Fig. 8, where we use c_0 and c_1 to denote, respectively, the computations $[(\emptyset, \{t_0\}, \{t_0\})]_{\leftrightarrows}$, and $[(\emptyset, \{t_1\}, \{t_1\})]_{\leftrightarrows}$ in both $ct(N)$ and $ct(M)$. Analogously, p_1 and p_0 indicate respectively the homotopy classes of the paths $[(\{t_0\}, \{t_1\}, \{t_0, t_1\})]_{\leftrightarrows}$ and $[(\{t_1\}, \{t_0\}, \{t_0, t_1\})]_{\leftrightarrows}$. However, $c_0; p_1$ and $c_1; p_0$ yield the same result $c = [(\emptyset, \{t_0, t_1\}, \{t_0, t_1\})]_{\leftrightarrows}$ in $\mathcal{C}(ct(N))$, whereas in $\mathcal{C}(ct(M))$ they denote two different objects:

$$c' = [(\emptyset, \{t_0\}, \{t_0\})(\{t_0\}, \{t_1\}, \{t_0, t_1\})]_{\leftrightarrows}, \quad \text{and}$$
$$c'' = [(\emptyset, \{t_1\}, \{t_1\})(\{t_1\}, \{t_0\}, \{t_0, t_1\})]_{\leftrightarrows}.$$

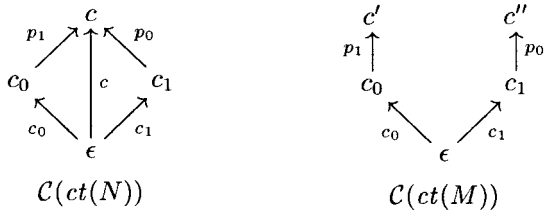

FIG. 8. The categories $\mathcal{C}(ct(N))$ and $\mathcal{C}(ct(M))$ for the nets in Fig. 2.

2.2. *Configuration Structures and Concurrent Transition Systems*

In this section we first give a categorical structure to the class of configuration structures and then show that the obvious injection of configuration structures in CTS yields a reflection.

DEFINITION 2.10. Given two configuration structures (E_1, C_1) and (E_2, C_2), a *cs-morphism* from (E_1, C_1) to (E_2, C_2) is a function $g : E_1 \to E_2$ such that for each configuration $X \in C_1$, then $g^{\mu}(X) \in C_2$. We denote by **CSCat** the category of configuration structures (as objects) and cs-morphisms (as arrows).

The obvious injection functor $\mathcal{J}(\_)$ from **CSCat** to **CTS** maps a configuration structure $CS = (E, C)$ onto the concurrent transition system

$$\mathcal{J}(CS) = (C, E, \ trans_{CS}, s_0),$$

where $trans_{CS} = \{(X, L, Y) \mid X \xrightarrow{L} Y\}$, and maps a cs-morphism $g: E_1 \to E_2$ onto the morphism (g', g), where $g': C_1 \to C_2$ is the obvious extension g^{μ} of g to multisets, with a domain restricted to C_1.

THEOREM 2.2. *The functor* $\mathcal{J}(\_)$: **CSCat** $\to$ **CTS** *has a left adjoint* $\mathcal{R}(\_)$: **CTS** $\to$ **CSCat**. *Moreover, since the counit of the adjunction is the identity,* $\mathcal{J}(\_)$ *and* $\mathcal{R}(\_)$ *define a reflection.*

Proof. We sketch the proof, giving the precise definition of the reflection functor. The reflection functor $\mathcal{R}(\_)$ maps a uniform CTS $H = (S, L, \ trans, s_0)$ onto the configuration structure $\mathcal{R}(H) = (L, C_S)$ such that $C_S = \{\varsigma_s \mid s \in S\}$ (recall that ς_s is the multiset union of the actions of any run leading to s). ■

We denote by $\rho_H: H \to \mathcal{J}(\mathcal{R}(H))$ the component at H of the unit of the adjunction.

THEOREM 2.3 (Configuration Structures via CTSs). *Let N be a marked PT net. Then $cs(N) = \mathcal{R}(ct(N))$.*

Proof. The events of $cs(N)$, the actions of $ct(N)$, and, therefore, the events of $\mathcal{R}(\ ct(N))$ are the transitions of N. The states S of the uniform CTS $ct(N)$ are exactly the configurations of $cs(N)$, and for each $s \in S$, we have $\varsigma_s = s$. This is sufficient, since a configuration structure is entirely determined by its set of configurations. ■

These results support our claim that configuration structures do not offer a faithful representation of net behaviors. In fact, $\mathcal{R}(\_)$ clearly collapses the structure excessively, since the natural transformation associated with the reflection map ρ can identify nonhomotopic runs (e.g., c' and c'' of Fig. 8).

3. PRE-NETS

Pre-nets are nets whose states are strings of tokens (as opposed to multisets). Such states 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 fetched and the results are produced, as if it were an elementary strongly concatenable process.

We 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, whose unit is the empty string ϵ. Moreover, for $w \in \lambda(S)$, we write $|w|$ to denote the length of w, w_i to denote the ith element of w, and $\mu(w)$ to denote the underlying multiset of w.

DEFINITION 3.1. 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 $\zeta_0, \zeta_1 : T \to \lambda(S)$ 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 [34], 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 is enough to choose one representative for each transition.

DEFINITION 3.2. 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 $g_t : T \to T'$ is function, and $g_p : \lambda(S) \to \lambda(S')$ 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.

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 3.1. *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 $\mathcal{A}(\_) : \mathbf{PreNet} \to \mathbf{Petri}$ 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 inverse left adjoint $\hat{\mathcal{A}} : \mathbf{Net} \to \mathbf{Petri}$ (with $\hat{\mathcal{A}}_{|\mathbf{PreNet}} = \mathcal{A}$ and $\hat{\mathcal{A}}_{|\mathbf{Petri}} = 1_{\mathbf{Petri}}$), yielding a *reflection*; 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 explains the rationale behind our proposed approach to the *ITph*.

Under the *ITph*, the natural algebraic models for representing concurrent computations on pre-nets belong to 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 a distributed state as a collection of atomic entities (tokens in places) which the net theory is based on. We denote such a category by **FSSMC**.

PROPOSITION 3.2. *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 the elements in $\lambda(S_R)$ as objects, and as arrows those generated by the rules in Table 5, modulo the axioms of monoidal categories (associativity, functoriality, identities, unit), including the coherence axioms that make $c_{\_,\_}$ the symmetry natural isomorphism. ■

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 [30]. In computer science similar constructions were given by Hotz's X-categories [16] and by Benson [2], with grammars as the primary area of application, and therefore for categories that are 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 3.1). Furthermore, what we really want is a quotient of the free construction, as explained in Theorem 3.2. The main result is that any two pre-nets representing isomorphic PT nets yield the same algebraic net semantics.

TABLE 5

Inference Rules for $\mathcal{Z}(R)$

$\dfrac{w \in \lambda(S_R)}{id_w : w \to w \in \mathcal{Z}(R)}$	$\dfrac{w, w' \in \lambda(S_R)}{c_{w,w'} : ww' \to w'w \in \mathcal{Z}(R)}$	$\dfrac{t \in T_R,\ \zeta_0(t) = \overline{u},\ \zeta_1(t) = \overline{v}}{t : \overline{u} \to \overline{v} \in \mathcal{Z}(R)}$
$\dfrac{\alpha : \overline{u} \to \overline{v},\ \beta : \overline{u'} \to \overline{v'} \in \mathcal{Z}(R)}{\alpha \otimes \beta : \overline{uu'} \to \overline{vv'} \in \mathcal{Z}(R)}$		$\dfrac{\alpha : \overline{u} \to \overline{v},\ \beta : \overline{v} \to \overline{v'} \in \mathcal{Z}(R)}{\alpha ; \beta : \overline{u} \to \overline{v'} \in \mathcal{Z}(R)}$

THEOREM 3.1. *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 onto 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 $\gamma(i, t) : \phi(\zeta_i(t)) \to \zeta_i(\phi(t))$. 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 monoidal categories. ■

THEOREM 3.2. *Let R be a pre-net. The category $\mathcal{Z}(R)$ quotiented out by the axiom $t = s_0; t; s_1$, for any transition $t : \overline{u} \to \overline{v}$ and symmetries $s_0 : \overline{u} \to \overline{u}$ and $s_1 : \overline{v} \to \overline{v}$, is equivalent to the category $\mathcal{Q}(\mathcal{A}(R))$ of strongly concatenable processes.*

Proof. This is straightforward according to the following argument. In $\mathcal{Q}(\mathcal{A}(R))$ we add a transition $t_{\overline{u},\overline{v}}$ for each transition of N and each pair of linearizations $\overline{u}$ and $\overline{v}$ of its pre- and post-set, and we quotient out by the axiom (4). On the other hand, in $\mathcal{Z}(R)$ we arbitrarily fix only *one* linearization of t, say $t_{\overline{u},\overline{v}}$, but we get all the others for free by composing $t_{\overline{u},\overline{v}}$ with symmetries (as s and s' in the axiom (4) of Definition 1.13). ■

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

One interesting question concerns relating morphisms of PT nets, rather than pre-nets, to the algebraic models obtained by pre-nets. For this purpose, for $f : N \to N'$ a morphism in **Petri**, consider the PT net N_f which has the same transitions as N and the same places as N' with a $t : f(u) \to f(v)$ in N_f corresponding to $t : u \to v$ in N. It can be easily verified that f is decomposable as $f = g; h$ with

$$g = \langle id_{T_N},\ f_{\mathrm{p}} \rangle : N \to N_f;$$
$$h = \langle f_{\mathrm{t}},\ id_{\mu(S_{N'})} \rangle : N_f \to N'.$$

It is straightforward that, for any pre-net R such that $\mathcal{A}(R) = N$ and for any linearization g'_{p} of $g_{\mathrm{p}} = f_{\mathrm{p}}$, we can always find a pre-net R_f such that $\mathcal{A}(R_f) = N_f$ and $\overline{g} = \langle id_t, g'_{\mathrm{p}} \rangle : R \to R_f$. Likewise, 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 $\overline{h} = \langle f_{\mathrm{t}}, id_{\lambda(S_{N'})} \rangle : R'_f \to R'$. 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 3.1 points us to a constructive way of relating $\mathcal{Z}(R_f)$ and $\mathcal{Z}(R'_f)$, thus yielding a lifting of f to a monoidal functor $F = \mathcal{Z}(\overline{g}); \Phi; \mathcal{Z}(\overline{h})$ between $\mathcal{Z}(R)$ and $\mathcal{Z}(R')$.

4. ALGEBRAIC SEMANTICS OF NETS VIA THEORY MORPHISMS

The algebraic semantics of PT nets and pre-nets can be conveniently expressed by means of morphisms between theories in **PMEqtl** [20], a logic of partial algebras with subsorts and subsort polymorphism whose sentences are Horn clauses on equations $\mathtt{t} = \mathtt{t}'$ and membership assertions $\mathtt{t} : \mathtt{s}$, and whose features (partiality, poset of sorts, membership assertions) offer a natural framework for the specification of categorical structures. Among the advantages mentioned in the Introduction, we recall that **PMEqtl** specifications can be made executable, e.g., by using the logic language Maude,[3] and therefore facilitate

[3] Maude [10] is a language of the OBJ-family, recently developed at SRI International; it is based on rewriting logic and supports partial membership equational logic specifications.

mechanical reasoning. This section and the Appendix provide a short introduction to the main ideas of (one-kinded) **PMEqtl**; see [20, 22] for self-contained presentations.

A *theory* in **PMEqtl** is a pair $\mathsf{T} = (\Omega, \Gamma)$, where $\Omega = (\mathsf{S}, \leq, \Sigma)$ is a signature over a poset of sorts (S is the set of sorts, ordered by $\leq$, and Σ is the set of partial operators) and Γ is a set of **PMEqtl**-sentences in the language of Ω. We denote by $\mathbf{PAlg}_\Omega$ the category of partial Ω-algebras and by $\mathbf{PAlg}_\mathsf{T}$ the full subcategory consisting of T-algebras, i.e., those partial Ω-algebras that satisfy all the sentences in Γ. A theory morphism H from T to T' is a mapping of the operators and sorts of T into T', preserving domain, codomain, and subsorting, and such that the translation of the axioms of T are entailed by those of T'. It induces a forgetful functor $\mathcal{U}_\mathsf{H} : \mathbf{PAlg}_{\mathsf{T}'} \to \mathbf{PAlg}_\mathsf{T}$ that—for T and T' theories without freeness constraints, as we will clarify later—admits a left adjoint $\mathcal{F}_\mathsf{H} : \mathbf{PAlg}_\mathsf{T} \to \mathbf{PAlg}_{\mathsf{T}'}$ whose effect is to lift H to a free model construction in $\mathbf{PAlg}_{\mathsf{T}'}$.

PROPOSITION 4.1 (cf. [20]). *The forgetful functor $\mathcal{U}_\mathsf{H} : \mathbf{PAlg}_{\mathsf{T}'} \to \mathbf{PAlg}_\mathsf{T}$ associated with a theory morphism $\mathsf{H} : \mathsf{T} \to \mathsf{T}'$ has a left adjoint $\mathcal{F}_\mathsf{H} : \mathbf{PAlg}_\mathsf{T} \to \mathbf{PAlg}_{\mathsf{T}'}$.*

A notion of *tensor product* for partial algebraic theories is used in [22] to obtain, among other things, a very elegant definition of the theory of monoidal categories (see Example A.2 in the Appendix). The tensor product (see for instance [17, 28]) is a well-known construction for ordinary algebraic (Lawvere) theories. Its importance can be understood by observing that the algebraic structures of a theory T can be defined not only on sets, the standard case denoted by $\mathbf{PAlg}_\mathsf{T}(\mathbf{Set})$—where **Set** is the category which has small sets as objects and functions as arrows—but also on any category **C** with suitable products or limits, to yield a category $\mathbf{PAlg}_\mathsf{T}(\mathbf{C})$. In particular, given two theories T and T', we can consider T'-algebras on the category of T-algebras or instead T-algebras on the category of T'-algebras. Regardless of the order, we obtain the same result up to isomorphism, namely, the category of algebras for the tensor product $\mathsf{T} \otimes \mathsf{T}'$ of both theories. If $\mathsf{T} = (\Omega, \Gamma)$ and $\mathsf{T}' = (\Omega', \Gamma')$ are theories in partial membership equational logic, where $\Omega = (\mathsf{S}, \leq, \Sigma)$ and $\Omega' = (\mathsf{S}', \leq', \Sigma')$, then their tensor product $\mathsf{T} \otimes \mathsf{T}'$ is the theory with signature $\Omega \otimes \Omega'$ which has the poset of sorts $(\mathsf{S}, \leq) \times (\mathsf{S}', \leq')$, 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 $g \in \Sigma'_m$ (n and m are the arities of the operators, and indices l and r stand for *left* and *right*, respectively, and witness whether the operator is inherited from the left or from the right component). The axioms of $\mathsf{T} \otimes \mathsf{T}'$ are determined from those of T and T' as explained in the Appendix. The essential property of the tensor product of theories is expressed by the following theorem, where $\mathbf{PAlg}_\mathsf{T}(\mathbf{C})$ indicates the category of T-algebras taken over a base category **C**, rather than over **Set**.

THEOREM 4.1 (cf. [22]). *Let T, T' be theories in partial membership equational logic. Then, we have the following isomorphisms of categories*:

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

We will use a self-explanatory Maude-like notation for presenting **PMEqtl** theories. As a compact *legenda* we say that the keyword sorts precedes the list of sorts of the signature, the keyword `subsorts` denotes the specification of subsorting, the keyword `ops` precedes the list of operators of the signature, keywords `eq` and `ceq` denote equations and conditional equations, respectively, while keywords mb and `cmb` denote membership assertions and conditional membership assertions, respectively. Theory morphisms are called *views* in Maude. Other tips will accompany the notation when needed. All the theories and views we discuss can be found in Appendix A.2.

4.1. *A Logical Characterization of the Two Philosophies and Their Relationship*

When considering the *CTph*, we are mainly concerned with defining the theories `PETRI` of PT nets and `CMONCAT` of strictly symmetric strict monoidal categories. However, to study the relationships between `PETRI` and `CMONCAT`, we also define an intermediate theory `CMON-AUT` of automata whose states form a commutative monoid. The main result of this section is then that the composition of the obvious inclusion functor of **Petri** into $\mathbf{PAlg}_{\texttt{CMON-AUT}}$ and the free functor $\mathcal{F}_\mathsf{V}$ from $\mathbf{PAlg}_{\texttt{CMON-AUT}}$ to $\mathbf{PAlg}_{\texttt{CMONCAT}}$ associated with the theory morphism `V` from `CMON-AUT` to `CMONCAT` corresponds exactly to the functor $\mathcal{T}(\_) : \mathbf{Petri} \to \mathbf{CMonCat}$.

PROPOSITION 4.2. *The functor* $\mathcal{T}(\_)$: **Petri** $\rightarrow$ **CMonCat** *is the composition*

$$\textbf{Petri} \hookrightarrow \textbf{PAlg}_{\texttt{CMON-AUT}} \stackrel{\mathcal{F}_{\mathsf{V}}}{\rightarrow} \textbf{PAlg}_{\texttt{CMONCAT}} = \textbf{CMonCat}.$$

Defining the theory CMONCAT is almost effortless thanks to the tensor product construction of theories. Essentially, we introduce a theory CMON of commutative monoids (see Table A3) and apply the tensor product construction with the theory CAT of categories (defined in the Example A.1, see Table A1). Here we exploit the possibility, given by Maude, of declaring the associativity, commutativity, and unit element as attributes `assoc`, `comm`, and `id:0` of the monoidal operator. We recall that, by writing the sorts of the arguments of an operator in its definition, we are assuming that it is a total operation on the elements of those sorts. The theory of strictly symmetric monoidal categories is then defined as in Table A3. Note the use of `left` and `right` corresponding to the indices l and r discussed in the informal explanation of the tensor product of theories.

In order to define a theory in **PMEqtl** that represents PT nets, we first introduce a theory whose models are automata and whose states form a commutative monoid (see Table A4).

PROPOSITION 4.3. *The category* **Petri** *is a full subcategory of* $\textbf{PAlg}_{\texttt{CMON-AUT}}$.

Proof. It is trivial to check that each PT net is just a model of CMON-AUT whose states are the object of the commutative monoid freely generated by the set of places and that morphisms between two such models are ordinary net morphisms. ■

Exploiting the modularity features of Maude, we can characterize **Petri** as a subcategory of $\textbf{PAlg}_{\texttt{CMON-AUT}}$, axiomatized by the theory `PETRI[S::TRIV]` in Table A4. In fact, it suffices to import a functional module `MSET[E::TRIV]` of multisets, parametrized by a functional theory TRIV, whose models are sets corresponding to the places of the net. Note that functional theories, enclosed within the keywords `fth` and `endfth`, have a *loose semantics*, in the sense that any algebra satisfying the sentences in the theory is an acceptable model, while functional modules, enclosed within the keywords `fmod` and `endfmod`, have an *initial semantics* imposing freeness constraints on acceptable models (i.e., any model of a parametrized module must be a free extension of a model of the parameter theory). Specifically, the module `MSET[E::TRIV]` imposes the freeness constraint associated with the theory inclusion $\texttt{TRIV} \stackrel{\mathsf{Y}}{\hookrightarrow} \texttt{MSET}$, so that the models have to be of the form $\mathcal{F}_{\mathsf{Y}}(X)$ for X a set. Then, when we import such a module in a `protecting` mode in PETRI, the freeness constraint that the monoid of states must be a free commutative monoid is imposed. The precise definitions are given in Table A4.

The inclusion functor from **Petri** to $\textbf{PAlg}_{\texttt{CMON-AUT}}$ is induced as the forgetful functor of the morphism I specified as a Maude `view` in Table A5.

Finally, the algebraic semantics of PT nets under the collective token philosophy, i.e., the construction $\mathcal{T}(\_)$, can easily be recovered via a simple theory morphism specified in a Maude-like notation in Table A5.

PROPOSITION 4.4. *The signature morphism* V *from* CMON-AUT *to* CMONCAT, *mapping sorts* `State` *and* `Transition` *to sorts* `Object` *and* `Arrow` *resp., relating homonym operators, and mapping operators* `origin(_)` *and* `destination(_)` *to operators* `d(_)` *and* `c(_)` *is a theory morphism.*

As stated in Proposition 4.2, the construction $\mathcal{T}(\_)$: **Petri** $\rightarrow$ **CMonCat** is given by the (functor) composition of the inclusion of **Petri** into $\textbf{PAlg}_{\texttt{CMON-AUT}}$ followed by $\mathcal{F}_{\mathsf{V}}$.

When dealing 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 *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 and whose states form a monoid (see Table A6). We then characterize the category **PreNet** of pre-nets as a subcategory of $\textbf{PAlg}_{\texttt{MON-AUT}}$, axiomatized by the theory `PRE-NETS[S::TRIV]` in Table A6. As for the parametrized module MSET, the module `LIST[E::TRIV]` of lists (parametrized on the functional theory TRIV) imposes the required freeness constraint associated with the theory inclusion $\texttt{TRIV} \stackrel{\mathsf{X}}{\hookrightarrow} \texttt{LIST}$. Then this constraint is imposed on PRE-NETS by the protecting importation of LIST (see Table A6). The inclusion functor from **PreNet** to $\textbf{PAlg}_{\texttt{MON-AUT}}$ is induced as the forgetful functor of the view L in Table A7.

PROPOSITION 4.5. *The category* **PreNet** *is a full subcategory of* $\mathbf{PAlg}_{\mathtt{MON\text{-}AUT}}$.

To define the theory SMONCAT of symmetric monoidal categories, we exploit the definition of the theory MONCAT of monoidal categories from [22], which for the reader's convenience is reported in the Appendix. The Maude-like specification of SMONCAT is given in Table A7. Finally, the algebraic semantics of pre-nets, i.e., the construction $\mathcal{Z}(\_)$, can easily be recovered via the view W defined in Table A7.

PROPOSITION 4.6. *The signature morphism* W *from* MON-AUT *to* SMONCAT *defined in Table* A7 *is a theory morphism.*

By Proposition 4.1, we know that W induces a free functor $\mathcal{F}_{\mathtt{W}}$ from $\mathbf{PAlg}_{\mathtt{MON\text{-}AUT}}$ to $\mathbf{PAlg}_{\mathtt{SMONCAT}}$. Then, we have the following result:

PROPOSITION 4.7. *The functor* $\mathcal{Z}(\_)$: **PreNet** $\rightarrow$ **SSMC** *is the composition*

$$\mathbf{PreNet} \hookrightarrow \mathbf{PAlg}_{\mathtt{MON\text{-}AUT}} \xrightarrow{\mathcal{F}_{\mathtt{W}}} \mathbf{PAlg}_{\mathtt{SMONCAT}} = \mathbf{SSMC}.$$

To conclude, we show that we can abstract from the constructions on pre-nets to those on PT nets by flattening the monoids of states to commutative monoids. In fact, we can define suitable theory morphisms U (from MON-AUT to CMON-AUT) and S (from SMONCAT to CMONCAT) and then compose them with V and W to build a commutative square.

PROPOSITION 4.8. *The signature morphisms* U *and* S *mapping operators and sorts as detailed in Table* A8 *are theory morphisms.*

Proof. The proof for U is immediate, since all the properties of monoids follow from those of commutative monoids. Similarly, the proof for S just entails verifying that all the axioms for symmetries can be derived from those of a commutative tensor product when symmetries are mapped to identities. ■

PROPOSITION 4.9. U; V = W; S.

Proof. By composition of morphisms it follows that both U; V and W; S yield the theory morphism H from MON-AUT to CMONCAT defined below:

```
view H from MON-AUT to CMONCAT is
  sort State to Object.   sort Transition to Arrow.
  op 1 to 0.              op _ ⊗ _ to _⊕_.
  op origin(_) to d(_).   op destination(_) to c(_).
endview
```
■

Note that, by working at a theory level, the corresponding proofs are straightforward. By Proposition 4.1 we have that the forgetful functors $\mathcal{U}_{\mathtt{U}}$ (from $\mathbf{PAlg}_{\mathtt{CMON\text{-}AUT}}$ to $\mathbf{PAlg}_{\mathtt{MON\text{-}AUT}}$) and $\mathcal{U}_{\mathtt{S}}$ (from $\mathbf{PAlg}_{\mathtt{CMONCAT}} = \mathbf{CMonCat}$ to $\mathbf{PAlg}_{\mathtt{SMONCAT}} = \mathbf{SSMC}$), induced by U and S, have left adjoints respectively $\mathcal{F}_{\mathtt{U}}$ and $\mathcal{F}_{\mathtt{S}}$, forming the commuting square of adjunctions in Fig. 9. The picture is completed by the injections of **PreNet** and **Petri** into $\mathbf{PAlg}_{\mathtt{MON\text{-}AUT}}$ and $\mathbf{PAlg}_{\mathtt{CMON\text{-}AUT}}$ respectively and by the functor $\mathcal{A}$: **PreNet** $\rightarrow$ **Petri** defined in Proposition 3.1, which enjoy the following properties.

PROPOSITION 4.10. $\mathcal{A}(\_); \mathcal{L}(\_) = \mathcal{I}(\_); \mathcal{F}_{\mathtt{U}}(\_)$.

COROLLARY 4.1. $\mathcal{A}(\_); \mathcal{T}(\_) = \mathcal{Z}(\_); \mathcal{F}_{\mathtt{S}}(\_)$.

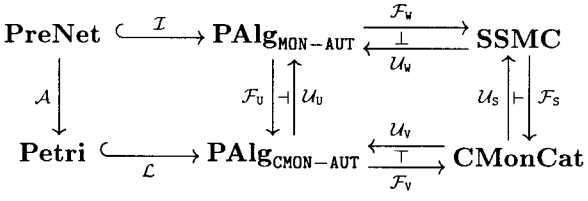

FIG. 9. The Complete scheme of adjunctions and injections.

TABLE 6

Conceptual Classification of the Approaches Considered

	Structures		
Computation model	Behavioral	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	$\mathcal{P}(N)$, $\mathcal{Q}(N)$, $\mathcal{Z}(R)$	$\mathtt{CAT} \otimes \mathtt{MON} + \mathtt{SYM}$

CONCLUDING REMARKS

The conceptual framework of this paper is summarized in Table 6, which illustrates the constructions and results of our research program on the behavioral, 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.

Regarding the first row of the table, we have first concentrated our attention on the expressiveness of some collective-token semantics for PT nets. In particular, to remedy the weakness of configuration structures, we have introduced concurrent transition systems—a version of higher dimensional transition systems [9] more suited to the collective token philosophy, because they do not assign individual identities to multiple action occurrences in a multiset—and have shown that they can provide a faithful description of net behaviors. The diagram of functors, equivalences, and natural transformations in Fig. 10 summarizes the relationships between all these models.

In the diagram, commutation on the nose (resp. natural equivalence) is represented by $=$ (resp. $\simeq$), and ρ denotes the unit of the reflection into the subcategory of configuration structures. The functor $\mathcal{CP}(\_)$ gives the category of Best–Devillers commutative processes. The functor $ct(\_)$ corresponds to the construction of the CTS for a given net, as defined in Section 2. The functor $\mathcal{C}(\_)$ yields the construction of the category of computations (i.e., homotopy equivalence classes of paths beginning in the initial state) of a CTS. The equivalence $\simeq$ between $\mathcal{C}(ct(\_))$ and $(u_{\text{in}} \downarrow \mathcal{T}(\_))$ is shown in Section 2.1, providing the faithfulness of the construction. The functor $cs(\_)$ represents the abstraction from nets to configuration structures. Unfortunately, **CSCat** is a reflective subcategory of **CTS**, as shown in Section 2.2 via the adjunction $\mathcal{R}(\_) \dashv \mathcal{J}(\_)$. The reflection functor $\mathcal{R}(\_)$ identifies too many things, so that the natural transformation associated with the reflection map ρ can identify nonhomotopic runs. Our running example shows that causality information can get lost when using configuration structures, because homotopic paths are mapped onto the same equivalence class. At a logical level, the presentation of the collective token semantics can easily be formulated as a theory morphism from the theory of nets to the theory of strictly symmetric monoidal categories.

On the other hand for the individual token interpretation, the best candidates for a suitable construction of the algebraic model, namely the symmetric monoidal categories $\mathcal{P}(N)$ of concatenable processes [11] and $\mathcal{Q}(N)$ of strongly concatenable processes [34], were both seen to be somehow unsatisfactory. In

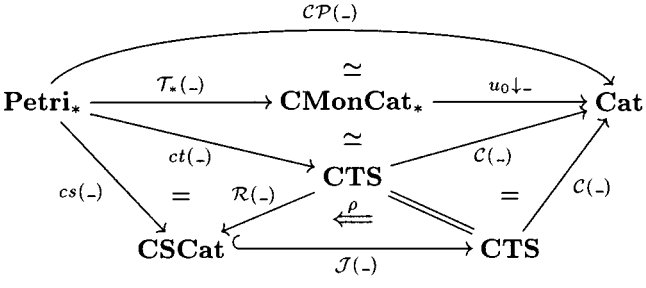

FIG. 10. Overall picture for the collective token philosophy.

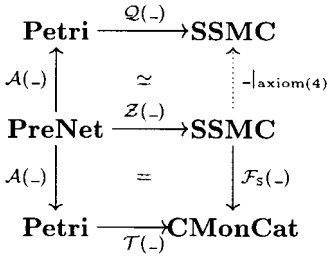

FIG. 11. Overall picture for the individual token philosophy.

fact $\mathcal{P}(\_)$ is a nonfunctorial construction, a drawback that inhibits many of the applications we have in mind, whilst $\mathcal{Q}(\_)$ solves the problem at the price of complicating the construction. Hence, we have proposed the categorical construction $\mathcal{Z}(R)$, based on pre-nets, as a suitable algebraic framework for nets in the *ITph*. 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, thanks to the preservation properties of adjoints, the semantic models of nets obtained as colimit constructions are found 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 by merging some of their places—is the pushout of their semantics. From a logical viewpoint, it is not difficult to formulate a theory SYM of permutations and symmetries (cf. [33]) bridging the gap between strictly symmetric monoidal categories and categories that are symmetric only up to coherent isomorphism. The results are summarized in Fig. 11, where the upper square in the diagram commutes up to a monoidal natural isomorphism (remember that $\mathcal{Q}$ is only pseudo-functorial) and the lower commuting square establishes the connections between the individual token philosophy for pre-nets and the collective token philosophy for Petri nets. We recall that $\mathcal{A}(\_)$ is the functor that forgets about the ordering of transition source and target, $\mathcal{F}_{\mathsf{S}}(\_)$ is the left-adjoint to the forgetful functor associated with the theory morphism S, whereas the dotted arrow labeled by $-|_{\text{axiom (4)}}$ denotes the monoidal quotient of the image of $\mathcal{Z}(\_)$ w.r.t. axiom (4) (via the obvious inclusion of **FSSMC** into **SSMC**).

The complete analysis and comparison of bisimulation-related issues in the various models considered in the paper (as in [13] for configuration structures) needs further work which we leave for future research.

APPENDIX

In this Appendix we recall the basic notions of partial membership equational logic (**PMEqtl**). Here we look at the one-kinded case, in which the poset of sorts has a single connected component. A more detailed exposition for the many-kinded case can be found in [20].

A.1. Partial Membership Equational Logic

DEFINITION A.1. A *signature* is a triple $\Omega = (\mathsf{S}, \leq, \Sigma)$, with $(\mathsf{S}, \leq)$ a poset with a top element $\top$, and $\Sigma = \{\Sigma_k\}_{k \in \mathbb{N}}$ a family of sets indexed by natural numbers. The poset $(\mathsf{S}, \leq)$ is called the poset of *sorts* of Ω.

DEFINITION A.2. Given a signature $\Omega = (\mathsf{S}, \leq, \Sigma)$, a *partial* Ω*-algebra* A assigns:

1. to each $\mathsf{s} \in \mathsf{S}$ a set A_{s}, in such a way that whenever $\mathsf{s} \leq \mathsf{s}'$, we have $A_{\mathsf{s}} \subseteq A_{\mathsf{s}'}$;
2. to each $f \in \Sigma_k$, $k \geq 0$, a partial function $A_f : A_\top^k \rightharpoonup A_\top$.

Given two partial Ω-algebras A and B, an Ω-*homomorphism* is a function $h : A_\top \to B_\top$ such that:

(i) for each $\mathsf{s} \in \mathsf{S}$, $h(A_{\mathsf{s}}) \subseteq B_{\mathsf{s}}$;

(ii) for each $f \in \Sigma_k$, $k \geq 0$, and $\vec{a} \in A_\top^k$, if $A_f(\vec{a})$ is defined, then $B_f(h^k(\vec{a}))$ is also defined and equal to $h(A_f(\vec{a}))$.

This determines a category $\mathbf{PAlg}_\Omega$.

Note that, by point (i), for each $\mathsf{s} \in \mathsf{S}$ the function h restricts to a function $h|_{\mathsf{s}} : A_{\mathsf{s}} \to B_{\mathsf{s}}$.

DEFINITION A.3. Let $\Omega = (\mathsf{S}, \leq, \Sigma)$ be a signature. Given a set of variables $X = \{x_1, \ldots, x_m\}$, a *variable declaration* $\tilde{X}$ is a sequence $x_1 : \mathsf{S}_1, \ldots, x_m : \mathsf{S}_m$, where for each $i \in [1, m]$, S_i is a set of sorts $\{\mathsf{s}_{i1}, \ldots, \mathsf{s}_{il_i}\}$. *Atomic Ω-formulas* are either equations $\mathtt{t} = \mathtt{t}'$, where $\mathtt{t}, \mathtt{t}' \in \mathbb{T}_\Sigma(X)$ (with $\mathbb{T}_\Sigma(X)$ the usual free Σ-algebra on variables X), or membership assertions of the form $\mathtt{t} : \mathsf{s}$, where $\mathtt{t} \in \mathbb{T}_\Sigma(X)$ and $\mathsf{s} \in \mathsf{S}$. *General Ω-sentences* are then Horn clauses of one of the two forms

$$(\forall \tilde{X})\ \mathtt{t} = \mathtt{t}' \Leftarrow \mathtt{t}_1 = \mathtt{t}'_1 \wedge \cdots \wedge \mathtt{t}_n = \mathtt{t}'_n \wedge \mathtt{t}''_1 : \mathsf{s}_1 \wedge \cdots \wedge \mathtt{t}''_m : \mathsf{s}_m$$
$$(\forall \tilde{X})\ \ \mathtt{t} : \mathsf{s} \ \Leftarrow \mathtt{t}_1 = \mathtt{t}'_1 \wedge \cdots \wedge \mathtt{t}_n = \mathtt{t}'_n \wedge \mathtt{t}''_1 : \mathsf{s}_1 \wedge \cdots \wedge \mathtt{t}''_m : \mathsf{s}_m,$$

where the $\mathtt{t}$, $\mathtt{t}'$, $\mathtt{t}_i$, $\mathtt{t}'_i$, and $\mathtt{t}''_j$ are terms in $\mathbb{T}_\Sigma(X)$ and the s_j are sorts in S.

Given a partial Ω-algebra A and a variable declaration $\tilde{X}$, we can define assignments $\sigma : \tilde{X} \to A$ in the obvious way (if $x_i : \mathsf{S}_i$, and $\mathsf{s} \in \mathsf{S}_i$, then we must have $\sigma(x) \in A_{\mathsf{s}}$) and then we can define a partial function $\overline{\sigma} : \mathbb{T}_\Sigma(X) \rightharpoonup A_\top$, extending σ in a unique way. For atomic sentences we then define satisfaction by $A, \sigma \models \mathtt{t} = \mathtt{t}'$ meaning that $\overline{\sigma}(\mathtt{t})$ and $\overline{\sigma}(\mathtt{t}')$ are both defined and $\overline{\sigma}(\mathtt{t}) = \overline{\sigma}(\mathtt{t})$ and by $A, \sigma \models \mathtt{t} : \mathsf{s}$ meaning that $\overline{\sigma}(\mathtt{t})$ is defined and $\overline{\sigma}(\mathtt{t}) \in A_{\mathsf{s}}$. Satisfaction of Horn clauses is then defined in the obvious way.

DEFINITION A.4. Given a set Γ of Ω-sentences, we let $\mathbf{PAlg}_{\Omega,\Gamma}$ be the full subcategory of $\mathbf{PAlg}_\Omega$ determined by those partial Ω-algebras that satisfy all the sentences in Γ. (The pair $\mathsf{T} = (\Omega, \Gamma)$ is a *theory*, and $\mathbf{PAlg}_{\mathsf{T}} = \mathbf{PAlg}_{\Omega,\Gamma}$ is the category of its models.)

As an example, we recall the definition of the theory of categories from [22].

EXAMPLE A.1. The theory of categories `CAT` is defined as follows. Its poset of sorts has sorts `Object` and `Arrow` with `Object` $\leq$ `Arrow`. There are two unary total 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. The detailed definition of the theory is given in Table A1. By convention, functions with given domain and codomain are total on that domain and codomain. It is easy to check that a model of `CAT` is exactly a category (in which objects coincide with identity arrows), and that a `CAT`-homomorphism is exactly a functor.

TABLE A1

The Theories `CAT`, `MON`, and `MONCAT`

```
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 if c(f) == d(g).
 ceq c(f) = d(g) if f;g : Arrow.
 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
```

DEFINITION A.5. Given two signatures $\Omega = (\mathsf{S}, \leq, \Sigma)$ and $\Omega' = (\mathsf{S}', \leq', \Sigma')$, a *signature morphism* $\mathsf{H} : \Omega \to \Omega'$ is given by (1) a monotonic function $\mathsf{H} : (\mathsf{S}, \leq) \to (\mathsf{S}', \leq')$, and (2) a family of functions $\{\mathsf{H}_k : \Sigma_k \to \Sigma'_k\}_{k \in \mathbb{N}}$.

A signature morphism H induces a forgetful functor $\mathcal{U}_\mathsf{H} : \mathbf{PAlg}_{\Omega'} \to \mathbf{PAlg}_\Omega$, where for each $A' \in \mathbf{PAlg}_{\Omega'}$ we have:

1. for each $\mathrm{s} \in \mathsf{S}$, $\mathcal{U}_\mathsf{H}(A')_\mathrm{s} = A'_{\mathsf{H}(\mathrm{s})}$;
2. for each $f \in \Sigma_k$,

$$\mathcal{U}_\mathsf{H}(A')_f = A'_{\mathsf{H}(f)} \cap \big(\underbrace{A'_{\mathsf{H}(\top)} \times \ldots \times A'_{\mathsf{H}(\top)}}_{k} \times A'_{\mathsf{H}(\top)} \big);$$

3. for each Ω'-homomorphism $h' : A' \to B'$, $\mathcal{U}_\mathsf{H}(h') = h'|_{\mathsf{H}(\top)} : A'_{\mathsf{H}(\top)} \to B'_{\mathsf{H}(\top)}$, which is well-defined as a restriction of h' because h' is sort-preserving.

DEFINITION A.6. Given two theories $\mathsf{T} = (\Omega, \Gamma)$ and $\mathsf{T}' = (\Omega', \Gamma')$, a *theory morphism* $\mathsf{H} : \mathsf{T} \to \mathsf{T}'$ is a signature morphism $\mathsf{H} : \Omega \to \Omega'$ such that $\mathcal{U}_\mathsf{H}(\mathbf{PAlg}_{\mathsf{T}'}) \subseteq \mathbf{PAlg}_\mathsf{T}$, so that $\mathcal{U}_\mathsf{H}$ restricts to a forgetful functor $\mathcal{U}_\mathsf{H} : \mathbf{PAlg}_{\mathsf{T}'} \to \mathbf{PAlg}_\mathsf{T}$.

A remarkable property of theory morphisms, namely that $\mathcal{U}_\mathsf{H}$ always has a left adjoint $\mathcal{F}_\mathsf{H}$, has been recalled in Proposition 4.1.

Following similar lines to those in [31], and also using the categorical axiomatization of canonical inclusions in a category $\mathbf{C}$ as a poset category of special monos $\mathbf{I} \subseteq \mathbf{C}$ satisfying suitable axioms, one can, given a signature $\Omega = (\mathsf{S}, \leq, \Sigma)$, define partial algebras in a category $\mathbf{C}$ with finite limits and a suitable poset of canonical inclusions $\mathbf{I}$. Each sort s has an associated object A_s, and if $\mathsf{s} \leq \mathsf{s}'$ then there is a canonical inclusion $A_\mathsf{s} \hookrightarrow A_{\mathsf{s}'}$ in $\mathbf{I}$. Given $f \in \Sigma_k$ we associate with it an arrow $\mathrm{Dom}(A_f) \to A_\top$ in $\mathbf{C}$, where $\mathrm{Dom}(A_f)$ is a subobject with a canonical inclusion $\mathrm{Dom}(A_f) \hookrightarrow A_\top^k$. In this way we can define categories $\mathbf{PAlg}_\Omega(\mathbf{C})$ and $\mathbf{PAlg}_{\Omega,\Gamma}(\mathbf{C})$ so that our categories $\mathbf{PAlg}_\Omega$ and $\mathbf{PAlg}_{\Omega,\Gamma}$ are the special case $\mathbf{PAlg}_\Omega(\mathbf{Set})$ and $\mathbf{PAlg}_{\Omega,\Gamma}(\mathbf{Set})$. It is not hard to check that $\mathbf{PAlg}_\Omega$ and $\mathbf{PAlg}_{\Omega,\Gamma}$ are categories with limits and that Ω-subalgebra inclusions $A \subseteq B$ constitute a poset category of canonical inclusions. Therefore, given two theories $\mathsf{T} = (\Omega, \Gamma)$ and $\mathsf{T}' = (\Omega', \Gamma')$, we can consider the category $\mathbf{PAlg}_\mathsf{T}(\mathbf{PAlg}_{\mathsf{T}'})$. For example, for T the theory of monoids and T' the theory of categories, $\mathbf{PAlg}_\mathsf{T}(\mathbf{PAlg}_{\mathsf{T}'})$ is the category of monoidal categories (see Example A.2). In a construction which is analogous to that for Lawvere algebraic theories, there is then a theory $\mathsf{T} \otimes \mathsf{T}'$ in partial membership equational logic such that

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

The explicit definition of $\mathsf{T} \otimes \mathsf{T}'$ is as follows.

DEFINITION A.7. Let $\mathsf{T} = (\Omega, \Gamma)$ and $\mathsf{T}' = (\Omega', \Gamma')$ be theories in **PMEqtl**, with $\Omega = (\mathsf{S}, \leq, \Sigma)$ and $\Omega' = (\mathsf{S}', \leq', \Sigma')$. Then their *tensor product* $\mathsf{T} \otimes \mathsf{T}'$ is the theory with signature $\Omega \otimes \Omega'$ having:

1. poset of sorts $(\mathsf{S}, \leq) \times (\mathsf{S}', \leq')$;
2. signature $\Sigma \otimes \Sigma'$, with an operator $f^l \in (\Sigma \otimes \Sigma')_n$ for each $f \in \Sigma_n$, and with an operator $g^r \in (\Sigma \otimes \Sigma')_m$ for each $g \in \Sigma'_m$. In particular, for f a constant in Σ_0 we get a constant f^l in $(\Sigma \otimes \Sigma')_0$.

The axioms of $\mathsf{T} \otimes \mathsf{T}'$ are the following:

Inherited axioms. For each axiom $\alpha = (\forall (x_1 : \mathsf{S}_1, \ldots, x_m : \mathsf{S}_m)\ \varphi(\vec{x}) \Leftarrow \varphi_1(\vec{x}) \wedge \cdots \wedge \varphi_n(\vec{x}))$ in Γ, with $\mathsf{S}_i = \{\mathsf{s}_{i1}, \ldots, \mathsf{s}_{il_i}\}$, for $1 \leq i \leq m$, we introduce an axiom

$$\alpha^l = \big(\forall (x_1 : \mathsf{S}_1^l, \ldots, x_m : \mathsf{S}_m^l)\ \varphi^l(\vec{x}) \Leftarrow \varphi_1^l(\vec{x}) \wedge \cdots \wedge \varphi_n^l(\vec{x})\big)$$

TABLE A2

Explicit Definition of the Theory MON ⊗ CAT

```
fth MON ⊗ CAT is
 sorts (Monoid,Object) (Monoid,Arrow).
 subsort (Monoid,Object) < (Monoid,Arrow).
*** [B] subalgebra axioms are implicit in op declarations
 op 1 left : -> (Monoid,Arrow).
 op _⊗_ left : (Monoid,Arrow) (Monoid,Arrow) -> (Monoid,Arrow).
 ops d(_) right c(_) right : (Monoid,Arrow) -> (Monoid,Object).
 op  _;_ right.
 var a : (Monoid,Object).
 vars f g h k : (Monoid,Arrow).
*** [A] inherited axioms
 eq  (f⊗g left)⊗h left = f⊗(g⊗h left) left.
 eq  f⊗(1 left) left = f.
 eq  (1 left)⊗f left = f.
 eq  d(a) right = a.
 eq  c(a) right = a.
 ceq a;f right = f if d(f) right == a.
 ceq f;a right = f if c(f) right == a.
 cmb f;g right : (Monoid,Arrow) if c(f) right == d(g) right.
 ceq c(f) right = d(g) right if f;g right : (Monoid,Arrow).
 ceq d(f;g right) right = d(f) right if c(f) right == d(g) right.
 ceq c(f;g right) right = c(g) right if c(f) right == d(g) right.
 ceq (f;g right);h right = f;(g;h right) right
     if c(f) right == d(g) right and c(g) right == d(h) right.
*** [C] homomorphism axioms
 eq  d(1 left) right = 1 left.
 eq  c(1 left) right = 1 left.
 eq  (1 left);(1 left) right = 1 left.
 eq  d(f⊗g left) right = (d(f) right)⊗(d(g) right) left.
 eq  c(f⊗g left) right = (c(f) right)⊗(c(g) right) left.
 ceq (f;g right)⊗(h;k right) left = (f⊗h left);(g⊗k left) right
     if f;g right : (Monoid,Arrow) and h;k right : (Monoid,Arrow).
endfth
```

with $\mathsf{S}_i^l = \{(\mathsf{s}_{i1}, \top'), \ldots, (\mathsf{s}_{il_i}, \top')\}$, $1 \leq i \leq m$, and with φ^l, φ_j^l the obvious translations of φ, φ_j obtained by replacing each $f \in \Sigma$ by its corresponding f^l. Similarly, we define for each axiom $\beta \in \Gamma'$ the axiom β^r and impose all these axioms.

Subalgebra axioms.

(i) For each $f \in \Sigma_n$ and each $\mathsf{s}' \in \mathsf{S}'$, $\mathsf{s}' \neq \top'$, we introduce the axiom:

$$\forall (x_1 : (\top, \mathsf{s}'), \ldots, x_n : (\top, \mathsf{s}'))\ \ f^l(\vec{x}) : (\top, \mathsf{s}') \Leftarrow f^l(\vec{x}) : (\top, \top').$$

(ii) For each $g \in \Sigma'_m$ and each $\mathsf{s} \in \mathsf{S}$, $\mathsf{s} \neq \top$, we introduce the axiom:

$$\forall (x_1 : (\mathsf{s}, \top'), \ldots, x_m : (\mathsf{s}, \top'))\ \ g^r(\vec{x}) : (\mathsf{s}, \top') \Leftarrow g^r(\vec{x}) : (\top, \top').$$

(iii) For each $(\mathsf{s}, \mathsf{s}') \in \mathsf{S} \times \mathsf{S}'$ with $\mathsf{s} \neq \top$ and $\mathsf{s}' \neq \top'$, we have the axiom:

$$\forall x : (\top, \top')\ \ x : (\mathsf{s}, \mathsf{s}') \Leftarrow x : (\top, \mathsf{s}') \wedge x : (\mathsf{s}, \top').$$

TABLE A3

The Theories CMON and CMONCAT

```
fth CMON is
 sort Monoid.
 op 0 : -> Monoid.
 op _⊕_ : Monoid Monoid -> Monoid
    [assoc comm id: 0].
endfth
```

```
fth CMONCAT is CMON ⊗ CAT renamed by (
 sort (Monoid,Object) to Object.
 sort (Monoid,Arrow) to Arrow.
 op  0 left to 0.
 op _⊕_ left to _⊕_.
 op _;_ right to _;_.
 op d(_) right to d(_).
 op c(_) right to c(_).).
endfth
```

TABLE A4

The Theories TRIV, MSET, CMON-AUT, and PETRI

```
fth CMON-AUT is
 sorts State Transition.
 op 0 : -> State.
 op _⊗_ : State State -> State
    [assoc comm id: 0].
 op origin(_) : Transition -> State.
 op destination(_) : Transition -> State.
endfth

fth PETRI[S :: TRIV] is
 protecting MSET[S] renamed by
            (sort MSet to Marking.).
 sort Transition.
 ops pre(_) post(_) : Transition -> Marking.
endfth
```

```
fth TRIV is sort Element.
endfth

fmod MSET[E :: TRIV] is
 sort MSet.
 subsort Element < MSet.
 op ∅ : -> MSet.
 op _+_ : MSet MSet -> MSet
    [assoc comm id: ∅].
endfm
```

TABLE A5

The Views I and V

```
view I from CMON-AUT to PETRI[S :: TRIV]
 is sort State to Marking.
 sort Transition to Transition.
 op origin(_) to pre(_).
 op destination(_) to post(_).
 op 0 to ∅.
 op _⊗_ to _+_.
endview
```

```
view V from CMON-AUT to CMONCAT is
 sort State to Object.
 sort Transition to Arrow.
 op 0 to 0.
 op _⊕_ to _⊕_.
 op origin(_) to d(_).
 op destination(_) to c(_).
endview
```

TABLE A6

The Theories MON-AUT, LIST, and PRE-NET

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

TABLE A7

The Theory SMONCAT and the Views L and W

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

view W from MON-AUT to SMONCAT is
 sort State to Object.
 sort Transition to Arrow.
 op 1 to 1.
 op _⊗_ to _⊗_.
 op origin(_) to d(_).
 op destination(_) to c(_).
endview
```

```
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
```

Homomorphism axioms. For each $f \in \Sigma_n$, $g \in \Sigma'_m$, $n + m \geq 0$, we introduce the axiom:

$$\forall \vec{x}\ \ f^l(g^r(\vec{x_1}\_), \ldots, g^r(\vec{x_n}\_)) = g^r(f^l(\vec{x}\_1), \ldots, f^l(\vec{x}\_m)) \Leftarrow \textstyle\bigwedge_{1\leq i\leq n} g^r(\vec{x_i}\_) : (\top, \top') \wedge \bigwedge_{1\leq j\leq m} f^l(\vec{x}\_j) : (\top, \top'),$$

where $\vec{x_i}\_ = \{x_{ij} : (\top, \top')\}_{1\leq j\leq m}$, for $i \in [1, n]$, $\vec{x}\_j = \{x_{ij} : (\top, \top')\}_{1\leq i\leq n}$, for $j \in [1, m]$, and $\vec{x} = \{x_{ij} : (\top, \top')\}_{1\leq j\leq m}^{1\leq i\leq n}$.

The tensor product of theories is *functorial* in the category of theories. Thus, if $\mathsf{H} : \mathsf{T}_1 \to \mathsf{T}_2$ and $\mathsf{G} : \mathsf{T}'_1 \to \mathsf{T}'_2$ are theory morphisms, we have an associated theory morphism $\mathsf{H} \otimes \mathsf{G} : \mathsf{T}_1 \otimes \mathsf{T}'_1 \to \mathsf{T}_2 \otimes \mathsf{T}'_2$. The essential property of $\mathsf{T} \otimes \mathsf{T}'$ has been recalled in Theorem 4.1. For example, the definition of the theory MONCAT of monoidal categories used in Section 4.1 is almost effortless thanks to the tensor product construction.

EXAMPLE A.2. The theory MONCAT is defined in Table A1 by applying the tensor product construction to CAT and to the theory MON of monoids (see Table A1). Notice the use of left and right corresponding to the indices l and r discussed above. The explicit axiomatization of MON ⊗ CAT is given in Table A2.

A.2. Maude Specifications of Theories and Views for Nets

For the reader's convenience, we have collected in this appendix complete specifications for all the theories and views addressed in Section 4. In particular, Tables A3–A5 concern the algebraic semantics under the *CTph*, Tables A6–A7 concern the algebraic semantics under the *ITph*, and Table A8 relates the two different interpretations.

TABLE A8

The Views U and S

```
view U from MON-AUT to CMON-AUT is
 sort State to State.
 sort Transition to Transition.
 op 1 to 0.
 op _⊗_ to _⊕_.
 op origin(_) to origin(_).
 op destination(_) to destination(_).
endview
```

```
view S from SMONCAT to CMONCAT is
 sort Object to Object.
 sort Arrow to Arrow.
 op d(_) to d(_).
 op c(_) to c(_).
 op _;_ to _;_.
 op 1 to 0.
 op _⊗_ to _⊕_.
 op γ(_,_) to _⊕_.
endview
```

ACKNOWLEDGMENTS

We thank the anonymous referees for their interesting comments and Paolo Baldan for several helpful discussions.

REFERENCES

1. Bednarczyk, M. A., and Borzyszkowski, A. M. (1999), General morphisms of Petri nets, *in* "Proceedings of ICALP'99, 26th International Colloquim on Automata, Languages and Programming" (J. Wiedermann, P. van Emde Boas, and M. Nielsen, Eds.), Lecture Notes in Computer Science, Vol. 1644, pp. 190–199, Springer-Verlag, Berlin.
2. Benson, D. B. (1975), The basic algebraic structures in categories of derivations, *Inform. and Control* **28**, 1–29.
3. Best, E., and Devillers, R. (1987), Sequential and concurrent behaviour in Petri net theory, *Theoret. Comput. Sci.* **55**, 87–136.
4. Bouhoula, A., Jouannaud, J.-P., and Meseguer, J. (2000), Specification and proof in membership equational logic, *Theoret. Comput. Sci.* **236**, 35–132.
5. Brown, C., and Gurr, D. (1990), A categorical linear framework for Petri nets, *in* "Proceedings of LICS'90, 5th Symposium on Logics in Computer Science," pp. 208–218, IEEE Computer Society, Los Alamitos, CA.
6. Brown, C., Gurr, D., and de Paiva, V. (1991), A Linear Specification Language for Petri Nets," Technical Report PB-363, DAIMI.
7. Bruni, R., Meseguer, J., Montanari, U., and Sassone, V. (1998), A comparison of Petri net semantics under the collective token philosophy, *in* "Proceedings of ASIAN'98, 4th Asian Computing Science Conference" (J. Hsiang and A. Ohori, Eds.), Lecture Notes in Computer Science, Vol. 1358, pp. 225–244, Springer-Verlag, Berlin.
8. Bruni, R., Meseguer, J., Montanari, U., and Sassone, V. (1999), Functorial semantics for Petri nets under the individual token philosophy, *in* "Proceedings of CTCS'99, 8th Conference on Category Theory and Computer Science" (M. Hofmann, G. Rosolini, and D. Pavlovic, Eds.), Electronic Notes in Theoretic Computer Science, Vol. 29, Elsevier, Amsterdam/New York.
9. Cattani, G. L., and Sassone, V. (1996), Higher dimensional transition systems, *in* "Proceedings of LICS'96, 11th Symposium on Logics in Computer Science," pp. 55–62, IEEE Computer Society, Los Alamitos, CA.
10. Clavel, M., Durán, F., Eker, S., Martí-Oliet, N., Meseguer, J., and Quesada, J. (1999), "Maude: Specification and Programming in Rewriting Logic," SRI International, available at `http://maude.csl.sri.com/manual`.
11. Degano, P., Meseguer, J., and Montanari, U. (1996), Axiomatizing the algebra of net computations and processes, *Acta Inform.* **33**, 641–667.
12. Gabriel, P., and Ulmer, F. (1971), "Lokal präsentierbare Kategorien," Lecture Notes in Mathematics, Vol. 221, Springer-Verlag, Berlin.
13. van Glabbeek, R. J., and Plotkin, G. D. (1995), Configuration structures, *in* "Proceedings of LICS'95, 10th Symposium on Logics in Computer Science" (D. Kozen, Ed.), pp. 199–209, IEEE Computer Society, Los Alamitos, CA.
14. Goltz, U., and Reisig, W. (1983), The non-sequential behaviour of Petri nets, *Inform. and Comput.* **57**, 125–147.
15. Hoogers, P. W., Kleijn, H. C. M., and Thiagarajan, P. S. (1996), An event structure semantics for general Petri nets, *Theoret. Comput. Sci.* **153**, 129–170.
16. Hotz, G. (1965), Eine algebraisierung des syntheseproblemen von schaltkreisen, I and II, Elektronische Informationverarbeitung und Kybernetik **1**, 185–205; 209–231.
17. Lawvere, F. W. (1968), Some algebraic problems in the context of functorial semantics of algebraic theories, *in* "Proceedings of the Midwest Category Seminar II," Lecture Notes in Mathematics, Vol. 61, pp. 41–61, Springer-Verlag, Berlin.
18. Lodaya, K., Ramanujam, R., and Thiagarajan, P. S. (1989), A logic for distributed transition systems, *in* "Linear Time, Branching Time, and Partial Order in Logics and Models for Concurrency" (J. W. de Bakker, W. P. de Roever, and G. Rozenberg, Eds.), Lecture Notes in Computer Science, Vol. 354, pp. 508–522, Springer-Verlag, Berlin.
19. MacLane, S. (1971), "Categories for the Working Mathematician," Springer-Verlag, Berlin.
20. Meseguer, J. (1998), Membership algebra as a logical framework for equational specification, *in* "Proceedings of WADT'97, 12th Workshop on Recent Trends in Algebraic Development Techniques" (F. Parisi-Presicce, Ed.), Lecture Notes in Computer Science, Vol. 1376, pp. 18–61, Springer-Verlag, Berlin.
21. Meseguer, J., and Montanari, U. (1990), Petri nets are monoids, *Inform. and Comput.* **88**, 105–155.
22. Meseguer, J., and Montanari, U. (1998), Mapping tile logic into rewriting logic, *in* "Proceedings of WADT'97, 12th Workshop on Recent Trends in Algebraic Development Techniques" (F. Parisi-Presicce, Ed.), Lecture Notes in Computer Science, Vol. 1376, pp. 62–91, Springer-Verlag, Berlin.
23. Meseguer, J., Montanari, U., and Sassone, V. (1996), Process versus unfolding semantics for place/transition Petri nets, *Theoret. Computer Science* **153**, 171–210.
24. Meseguer, J., Montanari, U., and Sassone, V. (1997), Representation theorems for Petri nets, *in* "Foundations of Computer Science: Potential–Theory–Cognition, to Wilfried Brauer on the Occasion of His Sixtieth Birthday" (Ch. Freksa, M. Jantzen, and R. Valk, Eds.), Lecture Notes in Computer Science, Vol. 1337, pp. 239–249, Springer-Verlag, Berlin.
25. Mossakowski, T. (1996), Equivalences among various logical frameworks of partial algebras, *in* "Proceedings of CSL'95, 9th International Workshop on Computer Science Logic" (H. Kleine Büning, Ed.), Lecture Notes in Computer Science, Vol. 1092, pp. 403–433, Springer-Verlag, Berlin.
26. Mukund, M. (1992), Petri nets and step transition systems, *Internat J. Found. Comput. Sci.* **3**, 443–478.
27. Nielsen, M., Plotkin, G., and Winskel, G. (1981), Petri Nets, Event Structures and Domains, Part 1, *Theoret. Comput. Sci.* **13**, 85–108.

28. Pareigis, B. (1970), "Categories and Functors," Academic Press, San Diego.
29. Petri, C. A. (1962), "Kommunikation mit Automaten," Ph.D. thesis, Institut für Instrumentelle Mathematik, Bonn.
30. Pfender, M. (1974), "Universal Algebra in s-Monoidal Categories," Algebra Berichte 20, Department of Mathematics, University of Munich.
31. Poigné, A. (1985), Algebra categorically, *in* "Category Theory and Computer Programming" (D. Pitt, S. Abramsky, A. Poigné, and D. Rydeheard, Eds.), Lecture Notes in Computer Science, Vol. 240, pp. 76–102, Springer-Verlag, Berlin.
32. Reisig, W. (1985), "Petri Nets: An Introduction," EATCS Monographs on Theoretical Computer Science, Springer-Verlag, Berlin.
33. Sassone, V. (1996), An axiomatization of the algebra of Petri net concatenable processes, *Theoret. Comput. Science* **170**, 277–296.
34. Sassone, V. (1998), An axiomatization of the category of Petri net computations, *Math. Struct. Comput. Sci.* **8**, 117–151.
35. Winskel, G. (1987), Petri nets, algebras, morphisms and compositionality, *Inform. and Comput.* **72**, 197–238.
36. Winskel, G. (1988), An introduction to event structures, *in* "Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency" (J. W. de Bakker, W.-P. de Roever, and G. Rozenberg, Eds.), Lecture Notes in Computer Science, Vol. 534, pp. 364–397, Springer-Verlag, Berlin.