Executions in (Semi-)Integer Petri Nets are Compact Closed Categories

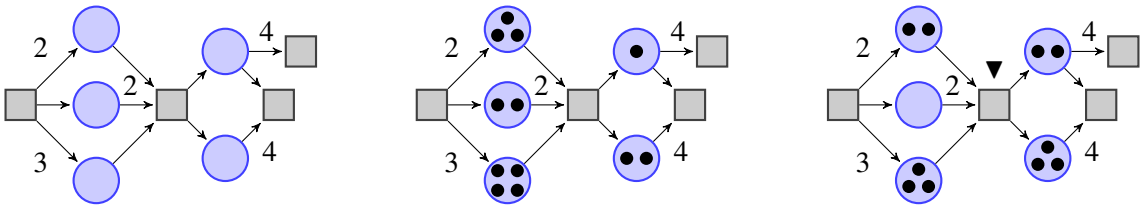
Fabrizio Genovese
Statebox Team
fabrizio@statebox.io
University of Oxford
fabrizio.genovese@cs.ox.ac.uk

Jelle Herold
Statebox Team
jelle@statebox.io

In this work, we analyse Petri nets where places are allowed to have a negative number of tokens. For each net we build its correspondent category of executions, which is compact closed, and prove that this procedure is functorial. We moreover exhibit a procedure to recover the original net from its category of executions, show that it is again functorial, and that this gives rise to an adjoint pair. Finally, we use compact closeness to infer that allowing negative tokens in a Petri net makes the causal relations between transition firings non-trivial, and we use this to model interesting phenomena in economics and computer science.

1 Introduction

Petri nets are a well known tool to study concurrent systems, and have been around for decades [14]. Intuitively, a Petri net consists of a set of *places*, pictorially depicted as circles, and a set of *transitions*, represented as grey squares, that are connected to places via directed edges, decorated with natural numbers (see Figure 1a). To avoid clutter, we omit the decoration when it is equal to 1. Places can contain *tokens*, that are represented as black dots. An assignment of tokens for a given net is called a *state* (Figure 1b). We interpret places as types, tokens as resources of the type corresponding to the place they



(a) A Petri net. (b) A state, before firing. (c) Marked transition has fired.

Figure 1

are in, and transitions as processes that convert resources into other resources. In more detail, when a transition converts resources into other resources we say that it *fires*. When firing, a transition consumes tokens in the places connected to it via an inbound edge, and produces tokens in places connected to it via an outbound edge. The number of tokens consumed/produced for each place is specified by the weighting on the edges. Firing is denoted with the symbol ▼ (see Figures 1b and 1c).

Petri nets were originally invented to study chemical reactions [14], but quickly found a place in computer science as models for concurrency [13, 15]. This is motivated by the idea that transitions sharing some input places have to *compete* for tokens to fire, and can then be thought of as *concurrent processes*.

Giulio Chiribella, Peter Selinger (Eds.): 15th International Conference on Quantum Physics and Logic (QPL 2018).
EPTCS 287, 2019, pp. 127–144, doi:10.4204/EPTCS.287.7

Petri nets, as presented above, are good models for concurrency but difficult to implement: Tokens represent resources, but the model itself offers no way of "tracking" their history (i.e. all the proccesses by which a given token was consumed/produced), which is fundamental to turning a Petri net into actual code. We have then to distinguish a Petri net, representing a process in the abstract, from its *executions* (sometimes also called *computations*), representing all the possible ways to run it.

In [2, 11, 17, 18] Petri nets have been characterized categorically. An outcome of this line of work has been linking Petri nets to their executions in terms of functorial relationships between categories [16]. In this work we carry on along the same lines, as follows: In Section 2 we generalize the notion of Petri net allowing for negative tokens, and explain why this is desirable; in Section 3 we reshape the categorical characterization of net executions, such that the functorial relationship is preserved; in Section 4, we will exploit some categorical properties of our model (mainly compact closedness) to show how our generalized nets have non-trivial behavior, and will provide examples of why this is useful; in Section 5 we will sketch future directions of research.

Interestingly, our approach to Petri nets will have striking similarities with the one used in categorical quantum mechanics, both from a structural point of view – our categories will be compact closed, and we will make great use of string diagrams throughout the paper – and from a conceptual one – causality flow in (semi-)integer nets will be non-trivial and similar in flavor to quantum teleportation.

2 Integer and semi-integer Petri nets

There are many definitions of Petri net, not always equivalent. We follow an approach similar to the one used in [16]. In the remainder of this work, we will adopt the following notations: Categorical composition $A \xrightarrow{f} B \xrightarrow{g} C$ will be denoted with $f;g$. Given a set S, $\mathscr{M}_S^{\mathbb{N}}$ will denote the set of all *finite multisets* on S, that is, the set of all functions $S \to \mathbb{N}$ that are non-zero only on a finite subset of S. Similarly, $\mathscr{M}_S^{\mathbb{Z}}$ will denote the set of all *finite signed multisets* on S, viz. the set of all functions $S \to \mathbb{Z}$ that are non-zero only on a finite subset of S. It is worth recalling the well-known fact that for each S, $\mathscr{M}_S^{\mathbb{N}}$ is the free commutative monoid generated by S under the operation of multiset union [16], while $\mathscr{M}_S^{\mathbb{Z}}$ is the free abelian group generated by S under multiset union and subtraction [9, 45-46].

Definition 2.1. *A* Petri net *N is a 4-tuple $(P_N, T_N, {}^\circ(-)_N, (-)^\circ{}_N)$, where:*

- *P_N is a set, called the* set of places*;*
- *T_N is a set, called the* set of transitions*;*
- *${}^\circ(-)_N, (-)^\circ{}_N$ are functions $T_N \to \mathscr{M}_{P_N}^{\mathbb{N}}$, called* input *and* output*, respectively.*

A state for the net *N is an element of $\mathscr{M}_{P_N}^{\mathbb{N}}$, representing how many tokens are in each place.*

Given nets $N := (P_N, T_N, {}^\circ(-)_N, (-)^\circ{}_N)$ and $M := (P_M, T_M, {}^\circ(-)_M, (-)^\circ{}_M)$, a morphism from *$N$* to *$M$ is a pair $\langle f, g\rangle$ where f is a function $T_N \to T_M$, g is a monoid homomorphism $\mathscr{M}_{P_N}^{\mathbb{N}} \to \mathscr{M}_{P_M}^{\mathbb{N}}$, and the following conditions hold:*

$$f;{}^\circ(-)_M = {}^\circ(-)_N;g \qquad f;(-)^\circ{}_M = (-)^\circ{}_N;g$$

It is straightforward to check that Petri nets and Petri net morphisms form a category, called ***Petri***$^{\mathbb{N}}$.

A morphism of nets $N \to M$ expresses the fact that *N can be simulated by M*, as thoroughly explained in [11]. We are now ready to define the main objects of our investigation, generalizing the previous definition:

Definition 2.2. *A* semi-integer Petri net *is a Petri net where states are elements of* $\mathscr{M}_{P_N}^{\mathbb{Z}}$. *An* integer Petri net *is a semi-integer Petri net where* ${}^{\circ}(-)_N, (-)^{\circ}{}_N$ *are functions* $T_N \to \mathscr{M}_{P_N}^{\mathbb{Z}}$. *A morphism of semi-integer nets is defined exactly as in 2.1, while a morphism* $N \to M$ *of integer nets is defined taking* g *in the pair* $\langle f, g \rangle$ *to be a group homomorphism* $\mathscr{M}_{P_N}^{\mathbb{Z}} \to \mathscr{M}_{P_M}^{\mathbb{Z}}$.
Semi-integer nets and their morphisms form again a category, and so do integer nets and their morphisms. We denote them as $\mathbf{Petri}^{\mathbb{Z}state}$ *and* $\mathbf{Petri}^{\mathbb{Z}}$, *respectively.*

All in all, semi-integer Petri nets are just ordinary nets where states are allowed to have negative tokens, but transitions can only produce/consume positive ones. Integer nets, instead, also allow for transitions to consume and produce negative tokens.

Furthermore, some readers may have noticed that the category $\mathbf{Petri}^{\mathbb{N}}$ is defined exactly as in [16], and that $\mathbf{Petri}^{\mathbb{N}}$ and $\mathbf{Petri}^{\mathbb{Z}\text{state}}$ are the same thing. This should not surprise, since the categories just defined do not capture any information about the net states, and only account for the underlying topology. This is part of a bigger problem, precisely that the role of states for a net has always been ambiguous. States represent the "dynamical part" of the net (resources produced/consumed) and are often not considered to be part of it. The impact that this ambiguity has on the definition of net executions will be evident and thoroughly discussed at the end of Section 3.

Now we recast our graphical formalism to deal with (semi-)integer nets. We represent a negative number of tokens in a place using red dots. For instance, the transition in Figure 2a has -2 tokens in its input place and 3 tokens in its output place. Note that $\mathbb{Z}$ being a group, we can always "produce" an equal number of positive and negative tokens in each place (see Figure 2b). This has dramatic consequences on the behaviour of our nets: Now each transition can fire at will "borrowing" tokens from a place, that is left with an equal number of tokens of the opposite sign (see Figures 2c and 2d).

(a) An integer Petri net. (b) Integer tokens. (c) Before firing. (d) After firing.

Figure 2

Semi-integer nets can be useful to model conflict resolution in concurrent behaviour. Petri nets are, in fact, good models for concurrent computation, but do not take into account what happens when the computation is shared by multiple agents over a non-ideal network. Consider, for instance, the net in Figure 3a: Transitions t_1 and t_2 have to compete for the token in p_1 and they cannot both fire. Now suppose that there are two users, say U_1 and U_2, that can operate on the net, deciding which transition to fire. When a user takes a decision, it is broadcast over the network to the other user, and the overall state of the net is updated. In a realistic scenario, though, broadcasting over the network takes time: User U_1 could decide to fire t_1 and user U_2 could decide to fire t_2 while the broadcast choice of U_1 has still to be received, putting the overall net into an illegal state (Figure 3b).
In such a situation we need a way to re-establish *consensus*, that is, decide unambiguously in which legal state the net is. There are multiple ways to do this, but our main concern here is that the usual Petri net formalism does not have a way to represent illegal states, which is fundamental to attacking the problem. With integer states we are able to easily represent such a situation using negative tokens, as in Figure 3c. Re-establishing consensus from an illegal state then amounts to getting back to a state where the number of tokens in each place is non-negative.

Integer nets can instead be useful to model economic phenomena: Places can be seen as actors (or

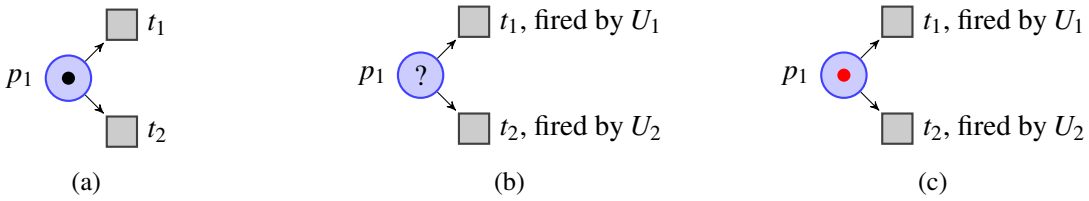


Figure 3

accounts) and tokens as entries in these accounts. Negative tokens then represent debit while positive tokens are credits, and transitions are bookkeeping events that convert between credits and debits. The characterization of economic phenomena in terms of process theories is object of a broader research that the Statebox team is carrying on along with multiple partners, and that also involves open games [5], macroeconomics [22] and open systems [20].

3 The category of executions of (semi-)integer Petri nets

Now we focus on defining executions for our nets. A suitable category of executions for $\mathbf{Petri}^{\mathbb{N}}$ has already been defined in [16], of which the work carried out in this section is a direct generalization. Most notably, the right categories of executions for $\mathbf{Petri}^{\mathbb{Z}^{\text{state}}}$ and $\mathbf{Petri}^{\mathbb{Z}}$ will turn out to be compact closed, while the category of executions for $\mathbf{Petri}^{\mathbb{N}}$ is not. This is striking considering that $\mathbf{Petri}^{\mathbb{N}}$ and $\mathbf{Petri}^{\mathbb{Z}^{\text{state}}}$ are the same category, and further highlights how the same category can be thought of in completely different ways.

The plan is as follows: We want to represent places as "basic types", states as monoidal products of these types (so, for instance, $A \otimes B \otimes A$ means "a state with two tokens in A and one in B") and transitions as morphisms between states. This tells us that our category has to be monoidal. Moreover, we want to represent negative tokens and the fact that they annihilate with positive ones, and to do so we resort to duals ($A^\star$ stands for "-1 tokens in A"), cups and caps (representing creation/annihilation of tokens of the opposite sign). Thus, we realize that a compact closed category would be a good model to represent (semi-)integer net computations.

Counter-intuitively, we cannot require the monoidal product to be commutative, because doing so would disrupt the functorial relationship between Petri nets and their executions, as proven in [16, Thm. 2.2]. This will force us to do quite a lot of bookkeeping, such as keeping track of permutations in a state.

Definition 3.1. *Let S be a set. S can be seen as a discrete category, and one can build the free strict*[1] *compact closed category generated by S, as shown in [10]. Denote this category with $\mathscr{S}$. We define the strict compact closed category $\mathscr{G}_S$ as $\mathscr{S}$ modulo the axioms*

$$\varepsilon_{u^{-1}} = \sigma_{u^{-1},u}; \varepsilon_u \qquad \eta_u; \sigma_{u^{-1},u}; \varepsilon_{u^{-1}} = id_I \qquad \forall u.(u \in \text{obj}\, \mathscr{G}_S) \tag{3.1}$$

Where σ denotes symmetries and η, ε units and co-units, respectively (also called cups and caps*).*

Given a set S, we will denote with $S^{\mathbb{Z}}$ the set of finite strings of elements of $S \cup S^{-1}$, where S^{-1} is the set of formal expressions $\left\{ s^{-1} \,\middle|\, s \in S \right\}$. It is easy to check that objects of $\mathscr{G}_S$ are just elements of $S^{\mathbb{Z}}$, that the monoidal product (denoted with $\otimes$) is just a concatenation of strings (with unit I being the empty string),

[1] Following the notation given in [10], by strict compact closed category we mean a category that is strict as a symmetric monoidal category, and for which the isomorphisms $(A \otimes B)^\star \simeq B^\star \otimes A^\star$, $I^\star \simeq I$ and $A^{\star\star} \simeq A$ are all identities.

and that the dual of a string is its inverse when $S^{\mathbb{Z}}$ is seen as the underlining set of the free group generated by S. Morphisms of $\mathscr{G}_S$ are obtained as finite compositions/monoidal products of identities, binary swaps, cups and caps on elements.

Remark 3.2. *Using the freeness of $\mathscr{S}$ one can moreover check that, for every strict compact closed category $\mathscr{C}$ for which the axioms in 3.1 hold, and for each function $f : S \to \mathrm{obj}\,\mathscr{C}$, there is a unique strict symmetric monoidal functor $\overline{f} : \mathscr{G}_S \to \mathscr{C}$, carrying the cups and caps of $\mathscr{G}_S$ to the cups and caps of $\mathscr{C}$, extending f.*

Definition 3.3. *An* integer Petri category *is a strict compact closed category whose monoid of objects is the monoid $S^{\mathbb{Z}}$ for some set S such that, for each object, $A^\star = A^{-1}$ and the axioms in 3.1 hold.*

Definition 3.4. *Given an integer Petri category $\mathscr{C}$, we call an arrow τ of $\mathscr{C}$* structural *if*

$$\tau = \bigotimes_{i_1=1}^{m_1} \alpha_{i_1}^1 ; \ldots ; \bigotimes_{i_n=1}^{m_n} \alpha_{i_n}^n$$

With each $\alpha_{i_j}^j$ being an identity, a symmetry, a cup or a cap.

Intuitively, structural arrows are needed for the above-mentioned bookkeeping in modelling a net execution. We want to represent the fact that transitions consume tokens produced by other transitions by composing processes, but to do this we need symmetries to reorder the way we present tokens if needed, and cups/caps to represent the creation/annihilation of tokens of the opposite sign when they are in the same place.

Definition 3.5. *Given an integer Petri category $\mathscr{C}$, an arrow τ of $\mathscr{C}$ is* primitive *if:*

- *τ is not structural;*
- *If $\tau = \alpha ; \beta$ then α is structural and β is primitive, or vice-versa;*
- *If $\tau = \alpha \otimes \beta$ then $\alpha = id_I$ and β is primitive, or vice-versa.*

We would like primitive arrows to represent "the actual transitions of the net". To accomplish this we think of an arrow τ in $\mathscr{C}$ as primitive when it's not in the image of the functor $\mathscr{G}_S \to \mathscr{C}$ obtained lifting the obvious inclusion $S \hookrightarrow \mathrm{obj}\,\mathscr{C}$. This is the best way to say that "a primitive arrow is the smallest arrow that is not structural", since structural arrows in $\mathscr{C}$ can be seen as "always coming from $\mathscr{G}_S$".

As we will see shortly, this is not enough to reliably identify what stands for a transition in a Petri category, and we will have to refine this idea further to make it work.

Lemma 3.6. *There is an obvious mapping $\mathfrak{M} : S^{\mathbb{Z}} \to \mathscr{M}_S^{\mathbb{Z}}$ that associates to each string $s \in S^{\mathbb{Z}}$ an integer multiset $S \to \mathbb{Z}$:*

$$\mathfrak{M}(s)(p) := \textit{ Occurrences of } p \textit{ in } s \; - \; \textit{Occurrences of } p^{-1} \textit{ in } s$$

Lemma 3.7. *Let $\mathscr{C}$ be an integer Petri category and let $S^{\mathbb{Z}}$ be its monoid of objects. Given an integer multiset $\nu : S \to \mathbb{Z}$, consider the set of objects of $\mathscr{C}$ such that their image through $\mathfrak{M}$ is ν, along with structural arrows between them. This gives a subcategory of $\mathscr{C}$, denoted with $\mathscr{G}_{\mathscr{C},\nu}$.*

If $\mathscr{M}_S^{\mathbb{Z}}$ can be seen as the free abelian group on S, $S^{\mathbb{Z}}$ clearly stands for the free group on S. $\mathfrak{M}$ acts identifying all the objects of $S^{\mathbb{Z}}$ that would end up being identified if we were to quotient it by introducing commutativity. $\mathscr{G}_{\mathscr{C},\nu}$ then is the category of all possible operations that we can make on an element in $S^{\mathbb{Z}}$ without changing the equivalence class it is sent to, viz. all the possible bookkeeping we can do on a object without altering the net state it corresponds to.

The following definitions are a direct generalization of the ones given in [16].

Definition 3.8. *Let $\mathscr{C}$ be an integer Petri category and let $S^{\mathbb{Z}}$ be its monoid of objects. Given integer multisets $v, v' : S \to \mathbb{Z}$, a* transition[2] *of $\mathscr{C}$ is a natural transformation $\tau : \pi_{\mathscr{C},v} \to \pi_{\mathscr{C},v'}$ whose components are all primitive, where $\pi_{\mathscr{C},v}$ and $\pi_{\mathscr{C},v'}$ are the obvious compositions of projection and inclusion functors:*

$$\pi_{\mathscr{C},v} : \mathscr{G}_{\mathscr{C},v'} \times \mathscr{G}_{\mathscr{C},v} \xrightarrow{\pi_1} \mathscr{G}_{\mathscr{C},v} \hookrightarrow \mathscr{C} \qquad \pi_{\mathscr{C},v'} : \mathscr{G}_{\mathscr{C},v'} \times \mathscr{G}_{\mathscr{C},v} \xrightarrow{\pi_2} \mathscr{G}_{\mathscr{C},v'} \hookrightarrow \mathscr{C}$$

We say that a strong monoidal functor between Petri categories $F : \mathscr{C} \to \mathscr{D}$ preserves transitions *if F carries structural arrows to structural arrows and for each transition τ of $\mathscr{C}$ there is a transition θ of $\mathscr{D}$ such that $F\tau_{u,v} = \theta_{Fu,Fv}$, where by $\tau_{u,v}$ we denote the components of τ (similarly for θ).*

As one can imagine, transitions in the previous definition represent the actual transitions of a given Petri net in its category of executions. To see this note that if we want to represent a transition as a process, then clearly we do not want to consider it as a different one if we permute the objects in its input or output. This is because in a Petri net a transition only cares about the number of tokens it consumes (produces) from (in) a place, and not about the order in which these tokens are received (sent). Similarly, we want to "ignore" all the pairs of type s, s^{-1} showing up in the process input and output, since these couples correspond to a 0 when translated to multisets, meaning that again in the Petri net formalism these pairs are "not seen" by any transition.

Clearly in the world of categories things are different, since morphisms of a category are sensitive to object order and/or presence of object+dual pairs. Requiring a morphism to be indifferent to this amounts exactly to asking that it commutes with swaps, cups and caps. In categorical terms it means requiring that the morphism is the component of a natural transformation between functors expressing the action of commuting and introducing/removing such pairs.

Finally, we see that if transitions in a net become natural transformations in the corresponding category of executions, then we want a notion of morphism between these categories that corresponds, functorially, to the notion of morphism we have between nets. Since a morphism between nets sends transitions to transitions, the natural requirement in the category of executions is that morphisms (viz. functors) preserve the natural transformations that are transitions.

Definition 3.9. *Given two integer Petri categories $\mathscr{C}, \mathscr{D}$ we define the relation $\mathfrak{R}_{\mathscr{C},\mathscr{D}}$ on transition-preserving functors $F, G : \mathscr{C} \to \mathscr{D}$ saying that $F, G \in \mathfrak{R}_{\mathscr{C},\mathscr{D}}$ if there are natural transformations $\tau : F \to G$ and $\tau' : G \to F$ such that their components are all structural arrows. For each couple $\mathscr{C}, \mathscr{D}$, $\mathfrak{R}_{\mathscr{C},\mathscr{D}}$ is an equivalence relation. Moreover,*

$$(F, G) \in \mathfrak{R}_{\mathscr{C},\mathscr{D}} \wedge (F', G') \in \mathfrak{R}_{\mathscr{D},\mathscr{E}} \implies (F; F', G; G') \in \mathfrak{R}_{\mathscr{C},\mathscr{E}}$$

This definition is, again, in line with the idea that we want to characterize functors only by looking at what they do to transitions. If they differ only in the way they handle the bookkeeping morphisms, then they should be regarded as the same. It is easy to check that identity functors preserve transitions, as does composition of transition-preserving functors. We moreover have:

Lemma 3.10. *$\mathfrak{R}$ is a congruence. If $F : \mathscr{C} \to \mathscr{D}$ preserves transitions and $\theta_{Fu,Fv} = F\tau_{u,v} = \theta'_{Fu,Fv}$, then $\theta' = \theta'$. Moreover, if $(F, G) \in \mathfrak{R}_{\mathscr{C},\mathscr{D}}$ and F, G preserve transitions, $F\tau_{u,v} = \theta_{Fu,Fv}$ iff $G\tau_{u,v} = \theta_{Gu,Gv}$.*

This is enough to ensure that the following definition is correct:

Definition 3.11. *We define the category of generalized Petri executions, **GExPetri**, as having integer Petri categories as objects and transition-preserving functors modulo $\mathfrak{R}$ as morphisms. Note that objects of **GExPetri** are compact closed categories, but **GExPetri** is not compact closed itself.*

[2] To be unambiguous, we will distinguish transitions in the net context from transitions in the categorical context by underlining.

Now we are finally ready to prove the main theorem of this work, that functorially associates, to each integer Petri net, a semantics representing its computations.

Theorem 3.12. *Consider a net $N := (P_N, T_N, {}^\circ(-)_N, (-)^\circ{}_N) \in \mathrm{obj}\ \mathbf{Petri}^{\mathbb{Z}}$. We can associate to N the strict compact closed category including $\mathscr{G}_{P_N}$ as a subcategory, plus the arrows defined by the following inference rules:*

$$\frac{t \in T_N}{t_{u,v} \in \mathrm{Hom}_{\mathfrak{F}(N)}[u,v]} \quad \forall u,v.(\mathfrak{M}(u) = {}^\circ(t) \wedge \mathfrak{M}(v) = (t)^\circ)$$

$$\frac{\alpha \in \mathrm{Hom}_{\mathfrak{F}(N)}[u,v],\ \beta \in \mathrm{Hom}_{\mathfrak{F}(N)}[u',v']}{\alpha \otimes \beta \in \mathrm{Hom}_{\mathfrak{F}(N)}[u \otimes u', v \otimes v']} \qquad \frac{\alpha \in \mathrm{Hom}_{\mathfrak{F}(N)}[u,v],\ \beta \in \mathrm{Hom}_{\mathfrak{F}(N)}[v,w]}{\alpha;\beta \in \mathrm{Hom}_{\mathfrak{F}(N)}[u,w]}$$

in which the following family of axioms holds:

$$p;t_{u',v'} = t_{u,v};q \qquad \forall p,q.(p \in \mathrm{Hom}_{\mathscr{G}_{P_N}}[u,u'] \wedge q \in \mathrm{Hom}_{\mathscr{G}_{P_N}}[v,v']) \tag{3.2}$$

This correspondence can be extended to a functor $\mathfrak{F}(-) : \mathbf{Petri}^{\mathbb{Z}} \to \mathbf{GExPetri}$.

Similarly, to each category $\mathscr{C} \in \mathrm{obj}\ \mathbf{Petri}^{\mathbb{Z}}$ we can associate the net $(P_N, T_N, {}^\circ(-)_N, (-)^\circ{}_N)$, where:

- *P_N is the generating set of* obj $\mathscr{C}$.
- $T_N := \bigcup_{v,v' \in \mathscr{M}_{P_N}^{\mathbb{Z}}} \{\tau \in \mathrm{Nat}[\pi_{\mathscr{C},v}, \pi_{\mathscr{C},v'}] \mid \tau \text{ is a } \underline{\text{transition}}\}$
- ${}^\circ(\tau : \pi_{\mathscr{C},v} \to \pi_{\mathscr{C},v'}) = v$
- $(\tau : \pi_{\mathscr{C},v} \to \pi_{\mathscr{C},v'})^\circ = v'$

This correspondence can again be extended to a functor $\mathfrak{U}(-) : \mathbf{GExPetri} \to \mathbf{Petri}^{\mathbb{Z}}$, producing an adjunction $\mathfrak{F}(-) \vdash \mathfrak{U}(-)$.

Finally, we can restrict the category **GExPetri** and the functor $\mathfrak{U}(-)$ to get the following:

Corollary 3.13. *Call $\mathbf{ExPetri}^{\mathbb{Z}}$ the full subcategory of **GExPetri** consisting of integer Petri categories whose morphisms are generated only from compositions and monoidal products of symmetries, cups, caps and transition components, modulo the compact closed categories axioms, the axioms in 3.1 and the axioms in 3.2. Then $\mathbf{Petri}^{\mathbb{Z}} \approx \mathbf{ExPetri}^{\mathbb{Z}}$.*

We found a suitable category to represent executions of integer Petri nets. Now we want to find a subcategory of $\mathbf{ExPetri}^{\mathbb{Z}}$ that can represent executions of semi-integer nets. This is indeed easy, and amounts to putting an obvious requirement on what transitions look like:

Definition 3.14. *We call an object $\mathscr{C}$ of $\mathbf{ExPetri}^{\mathbb{Z}}$* positive *if each transition in $\mathscr{C}$ is of type $\tau : \pi_{\mathscr{C},v} \to \pi_{\mathscr{C},v'}$ with all the elements in the images of v, v' being ≥ 0. Positive objects of $\mathbf{ExPetri}^{\mathbb{Z}}$ and morphisms between them form a full subcategory, denoted with $\mathbf{ExPetri}^{\mathbb{Z}^{state}}$.*

Theorem 3.15. *There is an equivalence $\mathbf{Petri}^{\mathbb{Z}^{state}} \approx \mathbf{ExPetri}^{\mathbb{Z}^{state}}$, and thus an equivalence*

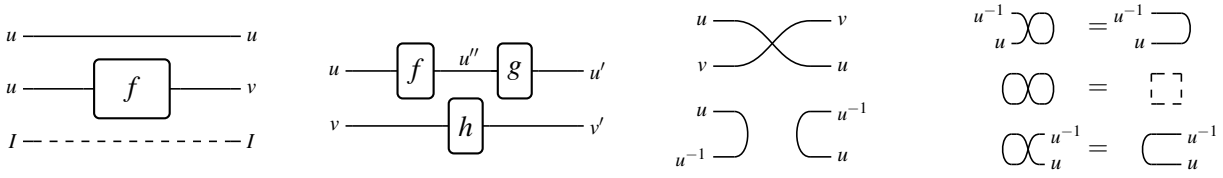
$$\mathbf{ExPetri}^{\mathbb{N}} \approx \mathbf{Petri}^{\mathbb{N}} = \mathbf{Petri}^{\mathbb{Z}^{state}} \approx \mathbf{ExPetri}^{\mathbb{Z}^{state}} \tag{3.3}$$

*Where $\mathbf{ExPetri}^{\mathbb{N}}$ and $\mathbf{Petri}^{\mathbb{N}}$ are the categories denoted as **PSSMC** and **Petri**, respectively, in [16].*

This result says that the category of executions for Petri nets defined in [16], whose objects are monoidal but not compact closed categories, is equivalent to $\mathbf{ExPetri}^{\mathbb{Z}^{\text{state}}}$, that is a category of compact closed categories. This should not surprise: $\mathbf{Petri}^{\mathbb{N}}$ and $\mathbf{Petri}^{\mathbb{Z}^{\text{state}}}$ are the same category, but represent different things: The fact is that they are different in the definition of what a state is, which is not accounted for in their categorical structure. Different definitions of state are then embedded in the structure of the objects of $\mathbf{ExPetri}^{\mathbb{N}}$ and $\mathbf{ExPetri}^{\mathbb{Z}^{\text{state}}}$, while the morphism structure of $\mathbf{ExPetri}^{\mathbb{N}}$ and $\mathbf{ExPetri}^{\mathbb{Z}^{\text{state}}}$ models how different nets interact with each other. This result then says that semi-integer nets simulate and interact with each other exactly as normal nets do: Different types of objects, representing different types of executions, are connected to each other in the same way, that only depends on the underlying topology.

4 The causal structure of net computations

Now we want to use the properties of compact closed categories to gain a better insight into how the executions of (semi-)integer Petri nets work, that is, we want to look inside the objects of $\mathbf{ExPetri}^{\mathbb{Z}}$ and $\mathbf{ExPetri}^{\mathbb{Z}^{\text{state}}}$. Compact closed categories admit a well-known graphical calculus [4, 19], that we will extensively use to represent what occurs. This calculus has been extensively used in the study of categorical quantum mechanics and the reader familiar with this topic should consider carefully such conceptual links while reading the following. First of all, a quick recap: In the graphical calculus for compact closed categories we express categorical facts as diagrams, that are read left to right. Objects are drawn as wires and arrows as boxes. The wire standing for the monoidal unit I is not drawn, or it is drawn as dashed when its presence needs to be emphasized (Figure 4a). Composition of arrows $f;g$ is just wiring the outputs of f into the inputs of g, while monoidal products are depicted putting boxes and wires next to each other (Figure 4b). Swaps, cups and caps are represented as in Figure 4c.



(a) Object, morphism, monoidal unit.

(b) $u \otimes v \xrightarrow{(f;g)\otimes h} u' \otimes v'$

(c) Swap (top), cap (b. left), cup (b. right).

(d) Axioms 3.1 and a consequence (bottom).

Figure 4: Graphical calculus for compact closed categories.

In a integer Petri category, moreover, the axioms in 3.1 hold, and can be represented graphically as in Figure 4d (note that the third line in Figure 4d is a consequence of the first, and has been added explicitly to give a sense of symmetry). The axioms, now that they are shown graphically, have a clear interpretation: The first and the third line in Figure 4d are interpreted as "it does not matter how you deform and twist them, in the end they are still a cap and a cup". The axiom in the second line represents the fact that we are not interested in registering events when "nothing happens": The symbol on the left represents the creation of a pair of type $u \otimes u^{-1}$ followed by its annihilation, while the one on the right is the monoidal unit. The axiom just says that if no element of the created couple undergoes any sort of process/transformation before the annihilation, then we may as well forget that the event happened.

The first thing that we want to do is to see what a transition looks like graphically. Given a category in

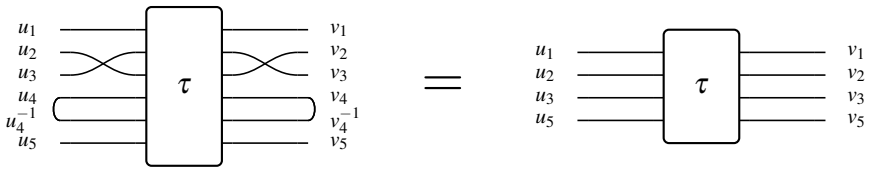
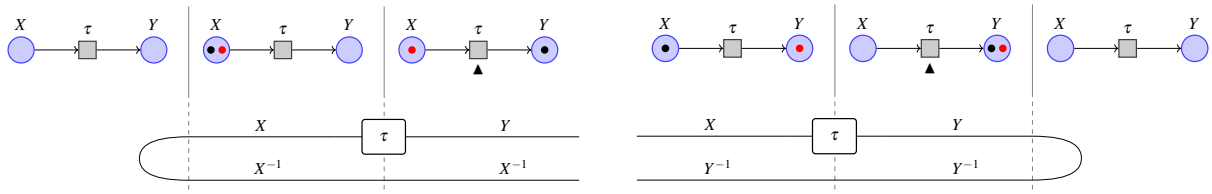


Figure 5: Transitions can be depicted as boxes with the property above, for suitable u_i and v_i.

ExPetri$^{\mathbb{Z}}$, we know that the only generating morphisms are transition components. The morphisms $\tau_{u,v}$ are all "avatars" of the same transition τ, to be used in different contexts depending on the bookkeeping we have to do. Since these components all stand for the same transition in the net $\mathfrak{U}(N)$, the only data that matters is the transition they are part of, and in diagrams we can safely omit the component indexes, as in Figure 5: Here the two boxes are clearly representing different components of the transition τ, and the equality, holding for any suitable choice of domain/co-domain (viz. objects that differ only by structural arrows, and get sent to the same multiset) is the exact graphical embodiment of τ being a natural transformation commuting with structural arrows.



(a) Turning a legal state into an illegal one.

(b) Turning an illegal state into a legal one.

Using cups and caps, we can see how we are now able to represent executions of Petri nets that weren't representable before. Look, for instance, at Figure 6a: Above, we are representing what happens from the point of view of Petri nets, while below we depict how the execution is built as the transition fires and couples of positive/negative tokens are produced/deleted. Vertical dashed lines separate consecutive instants in time. In the context of semi-integer nets, the net in Figure [6a] depicts an execution that turns a legal state into an illegal one. This represents well the situation detailed in Section 2, where we deduced that a net could always fire "borrowing" some tokens from a place. In the context of integer nets, the same figure may represent the idea of moving money from a given account (the first place) to another (the second place) without having it. Negative tokens then represent the necessary debt one has to make in order to take that money out of the account.

On the contrary, again in the context of semi-integer nets, the net in Figure 6b turns an illegal state into a legal one, literally "deleting" a negative token. From the point of view of integer Petri nets, this can be interpreted as the act of extinguishing a debt.

Combining the concepts considered above, we can take the *transpose* of any transition, as shown in Figure 7. This is demonstrates a duality that is ubiquitous when working with compact closed categories: Each transition can naturally be seen as a "forward-acting" process or as a "backward-acting" one on the dual entities. If a transition in a semi-integer net is thought of as "taking positive tokens from some place X to some place Y", Figure 7 shows how the same transition can be thought of as "taking negative tokens from Y to X". This means that if we are in an illegal state represented by a negative token in some place, we can shift that token backwards by firing transitions pointing at that place. This makes indeed sense: Firing a transition t from X to Y, with a negative token in Y, amounts to saying "If I have any way to produce a positive token in X, then I can automatically fix my problem in Y by firing t. So I might as well

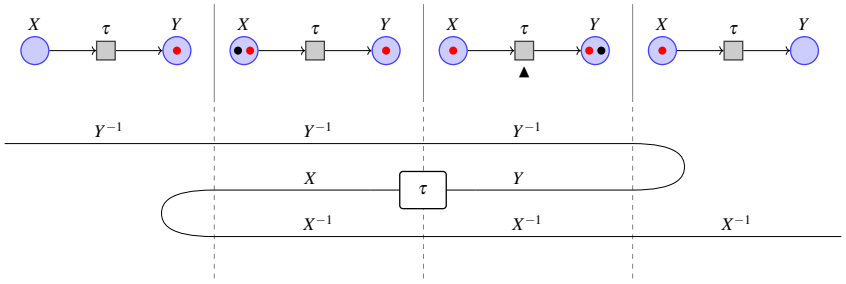


Figure 7: The transpose of a transition acts in the opposite direction on tokens of opposite sign.

say that t shifts my problem in Y to a problem in X".
In integer Petri nets, the transpose can be seen as switching from thinking in terms of credit to thinking in terms of debit: Paying someone could either mean "giving money" or "acquiring debt". In integer nets we can also have mixed transitions, like the *inversion transition* depicted in Figure 8. We think of it as a process consuming money and producing debt. The transpose, in this case, is again a process that consumes money and produces debt, but flowing in the other direction.

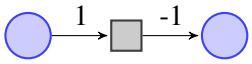


Figure 8: The Inversion transition

Now we can use this machinery to tackle the conflict resolution problem exposed in Section 2. The chain of events is represented in Figure 9: First, a user U_1 fires τ, consuming the only token present in X. Then user U_2, unaware of U_1's action, fires ν, putting the net into an illegal state. Finally, U_1 puts the net back to a legal state, "giving back" to X the resource used before, by firing μ. We clearly see how cups and caps *already give us the solution for our conflict problem*: If we apply the yanking equations and straighten the line, *we get a sequence of legal states*, namely the sequence of firings "τ, then μ, then ν". We witness how the graphical formalism for executions makes the solution to our problem easy to understand: If the vertical bars separate instants in time as it flows in the real word, the wire in the string diagram represents the flow of time *according to the net itself*: Eliminating negative tokens previously produced amounts to reshuffling the order of events in the net.
One downside of this approach is that U_1 is acting on a token that "was already there", while U_2 is acting on a token produced on the fly by a cup. Clearly these two tokens are different, since the first may have a complex history (e.g. it was consumed/produced by other transitions before) while the second one is created "on the fly". The whole point is obviously that U_2 *does not know* this: He is unaware of U_1's action, and believes to be using U_1's token!

Clearly, to solve this issue we need a way to decide who acted first between U_1 and U_2. This is a typical *consensus problem*, meaning that we need a way to establish an objective case (namely who acted first between U_1 and U_2) among a group of agents that may have different points of view. The advantage of our approach is that *the amount of consensus required is small*: Instead of having to converge on what is the best way to solve the conflict (i.e. by merging transitions in some strange way, or dropping the execution of one of the two transitions altogether etc.), the agents have to agree on a very simple fact that for instance could be solved, implementation-wise, just using timestamps. The execution structure of the net will take care of the rest.

As usual, Figure 9 has an interpretation also in the context of integer nets: In this case we again embrace the economics-oriented perspective. Places X, Y, Z represent accounts (more specifically portfolios), belonging to different agents, that will be denoted with the same names X, Y, Z to avoid clutter. Imagine

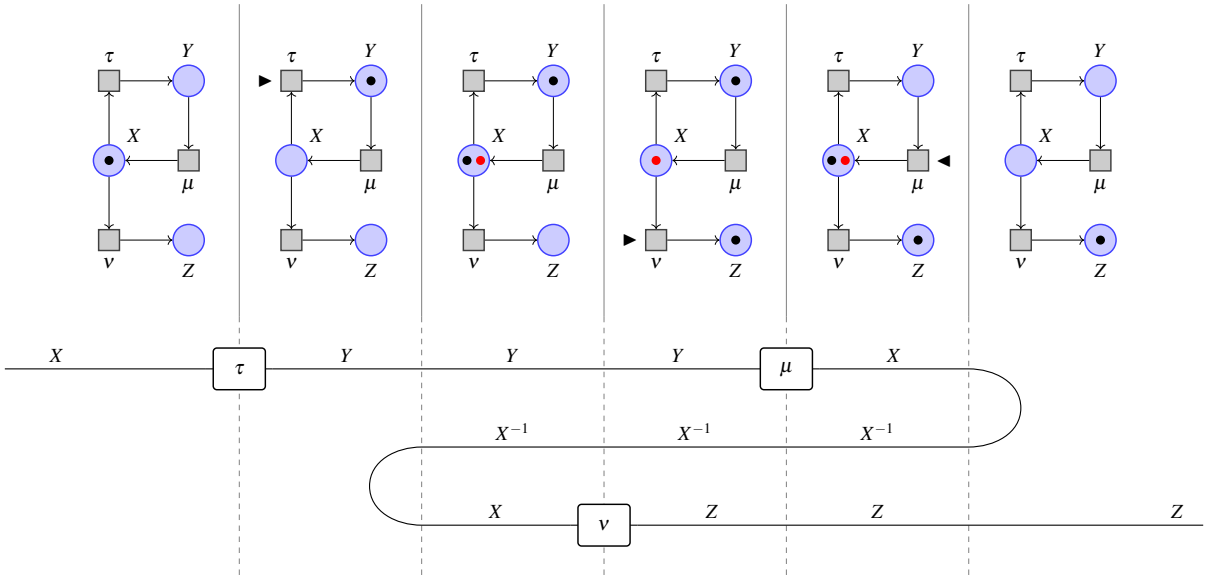


Figure 9: In **ExPetri**$^{\mathbb{Z}^{\text{state}}}$, Conflict resolution. In **ExPetri**$^{\mathbb{Z}}$, business strategy involving short-selling.

that X holds some financial instrument, and X predicts that prices will fall sharply in the future: Figure 9 may represent a possible business strategy. First, X sells the instrument to Y. To capitalize further on his prediction, he also *short sells* [3] the financial instrument to Z, for instance via going long [4] on a put option [5]. When prices go down, X buys back the financial instrument from Y and exercises the put option (effectively selling to Z), making a double profit.

In this example cups and caps are helpful to represent the idea that a financial operation can be executed at a different time from its purchase. Note also how we can infer, from this diagram, that Y's strategy consists in going long on the financial instrument owned by X: Y hopes to sell the instrument back to X at a profit, which is possible only if it acquires value in the short future.

5 Conclusion and Future work

In this work we generalized the notion of Petri nets in two different ways: We built their categories of executions and demonstrated the power of our formalism graphically. We were driven by practical applications, namely a way to represent conflict resolution in Petri nets and a way to represent economic phenomena and accounting. Both areas of research are deeply entangled in the Statebox project. Statebox [21] is a programming language for complex infrastructure entirely based on Petri nets, and runs in a decentralized way using Blockchain-based solutions. The Blockchain [12] is needed exactly to establish consensus when the net is run by multiple users.

[3] In finance, *going short* means buying a financial instrument with the expectation that it will decrease in value [8]. *Short selling* is the act of selling an asset that the seller does not own, and includes borrowing the instrument from a broker.

[4] *Going long* is the opposite of going short, and means buying a financial instrument with the expectation that it will acquire value [6].

[5] A *put option* is a contract giving the owner the right, but not the obligation, to sell a specified amount of the instrument at a specified price and time [7]. Note that in this case X is going long on the put option, because the more prices go down with regard to the selling value specified in the contract, the more the contract will acquire value.

Blockchain-based consensus deals with conflicts by accepting only one state and discarding conflicting others. Semi-integer nets provide an alternative way to preserve consensus by merging conflicting states in a net. This is achieved reshuffling the causality flow of the executed transitions. The practical use and hopefully implementation of the concepts presented here will surely be object of future work. We will also work to expand the categorical machinery to deal with more sophisticated conflicting scenarios.

On the other hand, integer nets are useful to model economic flows. This is very interesting from a Blockchain perspective, where the usual way of representing resources (computations included) is by monetizing them [3]. An operative account of economics is then very useful to design the best way to represent a given asset on the Blockchain, and to create high-level tools to solve long-standing open problems, such as smart contract analysis [1]. Future work will include investigating applications of integer Petri nets to real economic phenomena, and how they relate to other established tools in the field, such as Open Games [5].

Finally, from a genuinely academic point of view, the structural similarities with the categorical framework for quantum mechanics – namely compact closed categories – are worth studying in depth, since the possibility of describing quantum phenomena by means of Petri nets cannot be excluded a priori. If our intellectual resources allow it, this will surely be another future direction of research.

Acknowledgements

The authors would like to thank John Baez and Pawel Sobocinski, for having convinced them that the topic was worthy of being turned into a paper. They also thank David Spivak for having shared with them his invaluable knowledge and his TikZ macros, boosting their productivity by several orders of magnitude. Finally, they thank Emilia Gheorghe for having edited and proofread this document.

References

[1] Nicola Atzei, Massimo Bartoletti & Tiziana Cimoli (2017): *A survey of attacks on Ethereum smart contracts (SoK)*. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10204 LNCS, pp. 164–186, doi:10.1007/978-3-662-54455-6_8. (Cited on page 138.)

[2] Roberto Bruni, José Meseguer, Ugo Montanari & Vladimiro Sassone (2001): *Functorial Models for Petri Nets*. *Information and Computation* 170(2), pp. 207–236, doi:10.1006/inco.2001.3050. (Cited on page 128.)

[3] Vitalik Buterin (2014): *A Next-generation Smart Contract and Decentralized Application Platform*. *Ethereum* (January), pp. 1–36. Available at `http://buyxpr.com/build/pdfs/EthereumWhitePaper.pdf`. (Cited on page 138.)

[4] Bob Coecke & Aleks Kissinger (2017): *Picturing Quantum Processes. A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, doi:10.1017/9781316219317. (Cited on page 134.)

[5] Neil Ghani, Jules Hedges, Viktor Winschel & Philipp Zahn (2016): *Compositional Game Theory*. Available at `http://arxiv.org/abs/1603.04641`. (Cited on pages 130 and 138.)

[6] Investopedia: *Long Position*. Available at `https://www.investopedia.com/terms/l/long.asp`. (Cited on page 137.)

[7] Investopedia: *Put Option*. Available at `https://www.investopedia.com/terms/p/putoption.asp`. (Cited on page 137.)

[8] Investopedia: *Short Position*. Available at `https://www.investopedia.com/terms/s/short.asp`. (Cited on page 137.)

[9] K.D. Joshi (2003): *Applied Discrete Structures*. New Age International. (Cited on page 128.)

[10] Gregory Maxwell Kelly & Maria L. Laplaza (1980): *Coherence for Compact Closed Categories*. *Journal of Pure and Applied Algebra* 19, pp. 193–213, doi:10.1016/0022-4049(80)90101-2. (Cited on page 130.)

[11] José Meseguer & Ugo Montanari (1990): *Petri Nets are Monoids*. *Information and Computation* 88(2), pp. 105–155, doi:10.1016/0890-5401(90)90013-8. (Cited on page 128.)

[12] Satoshi Nakamoto (2008): *Bitcoin: A Peer-to-Peer Electronic Cash System*. *www.bitcoin.org*, pp. 1–9. Available at `https://bitcoin.org/bitcoin.pdf`. (Cited on page 137.)

[13] Mogens Nielsen (1991): *Models for Concurrency*. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 520 LNCS, pp. 43–46, doi:10.1007/3-540-54345-7_47. (Cited on page 127.)

[14] Carl Petri & Wolfgang Reisig (2008): *Petri Net*. *Scholarpedia* 3(4), p. 6477, doi:10.4249/scholarpedia.6477. (Cited on page 127.)

[15] Robert-Christoph Riemann (1999): *Modelling of Concurrent Systems: Structural and Semantical Methods in the High Level Petri Net Calculus*. Herbert Utz Verlag. (Cited on page 127.)

[16] Vladimiro Sassone (1995): *On the Category of Petri Net Computations*. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 915, pp. 334–348, doi:10.1007/3-540-59293-8_205. (Cited on pages 128, 129, 130, 131, 133, 134, and 144.)

[17] Vladimiro Sassone (1996): *An axiomatization of the algebra of Petri net concatenable processes*. *Theoretical Computer Science* 170(1-2), pp. 277–296, doi:10.1016/S0304-3975(96)00009-6. (Cited on page 128.)

[18] Vladimiro Sassone (2000): *On the Algebraic Structure of Petri Nets*. *Bulletin of the EATCS* 72, pp. 133–148. Available at `http://eprints.ecs.soton.ac.uk/11825/`. (Cited on page 128.)

[19] Peter Selinger (2010): *A Survey of Graphical Languages for Monoidal Categories*. In: *New structures for physics*, 813, Springer, pp. 289–355, doi:10.1007/978-3-642-12821-9_4. (Cited on page 134.)

[20] Pawel Sobociński (2010): *Representations of Petri Net Interactions*. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6269 LNCS, pp. 554–568, doi:10.1007/978-3-642-15375-4_38. (Cited on page 130.)

[21] Statebox Foundation (2017): *Statebox*. Available at `https://statebox.org`. (Cited on page 137.)

[22] Viktor Winschel & Markus Krätzig (2010): *Solving, Estimating, and Selecting Nonlinear Dynamic Models Without the Curse of Dimensionality*. *Econometrica* 78(2), pp. 803–821, doi:10.3982/ECTA6297. (Cited on page 130.)

A Proofs

Lemma 3.7. *Let $\mathscr{C}$ be an integer Petri category and let $S^{\mathbb{Z}}$ be its monoid of objects. Given an integer multiset $\nu : S \to \mathbb{Z}$, consider the set of objects of $\mathscr{C}$ such that their image through $\mathfrak{M}$ is ν, along with structural arrows between them. This gives a subcategory of $\mathscr{C}$, denoted with $\mathscr{G}_{\mathscr{C},\nu}$.*

Proof. Suppose that u is an object of $\mathscr{C}$ such that $\mathfrak{M}(u) = \nu$. Moreover, recall that by definition u has the form $\bigotimes_{i=1}^{n} u_i$ with each $u_i \in S$. The claim is obvious from the following considerations:

- Identities do not change anything, so clearly $id_u u = u$;
- $\mathfrak{M}$ is insensitive to ordering, all it does is counting. Hence applying a symmetry to u doesn't change its image through $\mathfrak{M}$, meaning: $\mathfrak{M}(\sigma(u)) = \mathfrak{M}(u)$;
- Given an element u_i of S, couples of the form (u_i^{-1}, u_i) do not change $\mathfrak{M}(u)$, since $\mathfrak{M}$ is evaluated as 1 on u_i and as -1 on u_i^{-1}. This means that such couples can be added and subtracted at will anywhere in/from u;
- All the cups/caps in $\mathscr{C}$ can be obtained composing symmetries and cups/caps on the elements of S, and their inverses. This means that cups/caps always add/remove elements from u in pairs as in the previous point, leaving $\mathfrak{M}(u)$ unaltered;
- A structural arrow is a composition of monoidal products of identities, symmetries and cups/caps. Since all these things do not alter $\mathfrak{M}(u)$, applying a structural arrow to $u \in \mathscr{G}_{\mathscr{C},\nu}$ gives us an object $v \in \mathscr{G}_{\mathscr{C},\nu}$.

Being u a generic element of $\mathscr{G}_{\mathscr{C},\nu}$ this proves that composition is well defined in $\mathscr{G}_{\mathscr{C},\nu}$. Associativity of morphisms is inherited from $\mathscr{C}$ and the existence of identities from the fact that identities are trivially structural arrows. □

Lemma 3.10. *$\mathfrak{R}$ is a congruence. If $F : \mathscr{C} \to \mathscr{D}$ preserves transitions and $\theta_{Fu,Fv} = F\tau_{u,v} = \theta'_{Fu,Fv}$, then $\theta' = \theta'$. Moreover, if $(F,G) \in \mathfrak{R}_{\mathscr{C},\mathscr{D}}$ and F,G preserve transitions, $F\tau_{u,v} = \theta_{Fu,Fv}$ iff $G\tau_{u,v} = \theta_{Gu,Gv}$.*

Proof. We start proving that $\mathfrak{R}_{\mathscr{C},\mathscr{D}}$ is an equivalence relation for each couple of integer Petri categories $\mathscr{C},\mathscr{D}$. Clearly for any functor $F : \mathscr{C} \to \mathscr{D}$ it is $(F,F) \in \mathfrak{R}_{\mathscr{C},\mathscr{D}}$ since there is always an identity natural transformation $id_F : F \to F$. Moreover, if $(F,G) \in \mathfrak{R}_{\mathscr{C},\mathscr{D}}$, then $(G,F) \in \mathfrak{R}_{\mathscr{C},\mathscr{D}}$ since the definition of $\mathfrak{R}_{\mathscr{C},\mathscr{D}}$ is totally symmetric (just invert the roles of τ and τ' in the definition). Now suppose that (F,G) and (G,H) are in $\mathfrak{R}_{\mathscr{C},\mathscr{D}}$. Then there are natural transformations

$$\tau : F \to G \qquad \tau' : G \to F \qquad \bar{\tau} : G \to H \qquad \bar{\tau}' : H \to G$$

With components all structural arrows. But then the componentwise compositions $\tau;\bar{\tau} : F \to H$ and $\bar{\tau}'\tau' : H \to F$ have all components structural arrows (since composition of structural arrows is a structural arrow), proving that $(F,H) \in \mathfrak{R}_{\mathscr{C},\mathscr{D}}$.

Now let's prove that $\mathfrak{R}$ defines a congruence. This means that it respects compositions. Consider $(F,G) \in \mathfrak{R}_{\mathscr{C},\mathscr{D}}$ and $(F',G') \in \mathfrak{R}_{\mathscr{D},\mathscr{E}}$. As usual we have:

$$\tau : F \to G \qquad \tau' : G \to F \qquad \bar{\tau} : F' \to G' \qquad \bar{\tau}' : G' \to F$$

So, the naturality conditions guarantee that, for each couple of objects $u, v \in \mathscr{C}$ and morphism $f : u \to v$,

$$\begin{array}{ccc}
(F;F')u & \xrightarrow{(F;F')f} & (F;F')v \\
\downarrow{\scriptstyle F'\tau_u} & & \downarrow{\scriptstyle F'\tau_v} \\
(G;F')u & \xrightarrow{(G;F')f} & (G;F')v \\
\downarrow{\scriptstyle \bar{\tau}_{Gu}} & & \downarrow{\scriptstyle \bar{\tau}_{Gv}} \\
(G;G')u & \xrightarrow{(G;G')f} & (G;G')v
\end{array}
\qquad
\begin{array}{ccc}
(G;G')u & \xrightarrow{(G;G')f} & (G;G')v \\
\downarrow{\scriptstyle G'\tau'_u} & & \downarrow{\scriptstyle G'\tau'_v} \\
(F;G')u & \xrightarrow{(F;G')f} & (F;G')v \\
\downarrow{\scriptstyle \bar{\tau}'_{Fu}} & & \downarrow{\scriptstyle \bar{\tau}'_{Fv}} \\
(F;F')u & \xrightarrow{(F;F')f} & (F;F')v
\end{array}$$

Commute. Since both F' and G' are transition-preserving they carry structural arrows to structural arrows, and hence the diagram on the left (resp. on the right) defines a natural transformation $F;F' \to G;G'$ (resp. $G;G' \to F;F'$) whose components are all structural arrows, proving $(F;F', G;G') \in \mathfrak{R}_{\mathscr{C},\mathscr{E}}$.

Now we prove the second claim. Consider a transition component $\theta_{x,y}$, and structural arrows $\gamma : x' \to x$, $\gamma' : y \to y'$, with $x, x' \in \mathscr{G}_{\mathscr{C},v}$ and $y, y' \in \mathscr{G}_{\mathscr{C},v'}$. Then, being θ a natural transformation, the following diagrams commute:

$$\begin{array}{ccc}
x' & \xrightarrow{\theta_{x',y}} & y \\
{\scriptstyle \gamma}\downarrow & & \\
x & \xrightarrow{\theta_{x,y}} & y
\end{array}
\qquad
\begin{array}{ccc}
x' & \xrightarrow{\theta_{x',y}} & y \\
 & & \downarrow{\scriptstyle \gamma'} \\
x' & \xrightarrow{\theta_{x',y'}} & y'
\end{array}$$

Proving $\gamma;\theta_{x,y};\gamma' = \theta_{x',y};\gamma' = \theta_{x',y'}$. From this it is easy to see that if $\theta_{Fu,Fv} = \theta'_{Fu,Fv}$, then $\theta = \theta'$. This is because given any two elements in $\mathscr{G}_{\mathscr{C},v}$ (in $\mathscr{G}_{\mathscr{C},v'}$, respectively) there is always a structural morphism in $\mathscr{G}_{\mathscr{C},v}$ (in $\mathscr{G}_{\mathscr{C},v'}$, respectively) connecting them, since each element can be obtained from another permuting it and adding/erasing pairs of elements u and u^{-1} using cups and caps.

The last claim is obvious. □

Theorem 3.12. *Consider a net $N := (P_N, T_N, {}^\circ(-)_N, (-)^\circ{}_N) \in \mathrm{obj}\,\textbf{\textit{Petri}}^{\mathbb{Z}}$. We can associate to N the strict compact closed category including $\mathscr{G}_{P_N}$ as a subcategory, plus the arrows defined by the following inference rules:*

$$\frac{t \in T_N}{t_{u,v} \in \mathrm{Hom}_{\mathfrak{F}(N)}[u,v]} \quad \forall u, v.(\mathfrak{M}(u) = {}^\circ(t) \wedge \mathfrak{M}(v) = (t)^\circ)$$

$$\frac{\alpha \in \mathrm{Hom}_{\mathfrak{F}(N)}[u,v],\ \beta \in \mathrm{Hom}_{\mathfrak{F}(N)}[u',v']}{\alpha \otimes \beta \in \mathrm{Hom}_{\mathfrak{F}(N)}[u \otimes u', v \otimes v']} \qquad \frac{\alpha \in \mathrm{Hom}_{\mathfrak{F}(N)}[u,v],\ \beta \in \mathrm{Hom}_{\mathfrak{F}(N)}[v,w]}{\alpha;\beta \in \mathrm{Hom}_{\mathfrak{F}(N)}[u,w]}$$

in which the following family of axioms holds:

$$p;t_{u',v'} = t_{u,v};q \qquad \forall p,q.(p \in \mathrm{Hom}_{\mathscr{G}_{P_N}}\left[u,u'\right] \wedge q \in \mathrm{Hom}_{\mathscr{G}_{P_N}}\left[v,v'\right]) \tag{3.2}$$

This correspondence can be extended to a functor $\mathfrak{F}(-) : \textbf{\textit{Petri}}^{\mathbb{Z}} \to \textbf{\textit{GExPetri}}$.

Similarly, to each category $\mathscr{C} \in \mathrm{obj}\ \textbf{\textit{Petri}}^{\mathbb{Z}}$ *we can associate the net* $(P_N, T_N, {}^{\circ}(-)_N, (-)^{\circ}{}_N)$*, where:*

- P_N *is the generating set of* $\mathrm{obj}\ \mathscr{C}$.
- $T_N := \bigcup_{v,v' \in \mathscr{M}_{P_N}^{\mathbb{Z}}} \left\{ \tau \in \mathrm{Nat}[\pi_{\mathscr{C},v}, \pi_{\mathscr{C},v'}] \;\middle|\; \tau \text{ is a } \underline{\textit{transition}} \right\}$
- ${}^{\circ}(\tau : \pi_{\mathscr{C},v} \to \pi_{\mathscr{C},v'}) = v$
- $(\tau : \pi_{\mathscr{C},v} \to \pi_{\mathscr{C},v'})^{\circ} = v'$

This correspondence can again be extended to a functor $\mathfrak{U}(-) : \textbf{\textit{GExPetri}} \to \textbf{\textit{Petri}}^{\mathbb{Z}}$*, producing an adjunction* $\mathfrak{F}(-) \vdash \mathfrak{U}(-)$.

Proof. First of all, note that the category $\mathfrak{F}(N)$ is isomorphic to the category $\mathscr{C}$ having $P_N^{\mathbb{Z}}$ as objects, and whose arrows are generated by the rules:

$$\frac{u \in P_N^{\mathbb{Z}}}{id_u \in \mathrm{Hom}_{\mathscr{C}}[u,u]} \quad \frac{u \in P_N^{\mathbb{Z}}}{\varepsilon_u \in \mathrm{Hom}_{\mathscr{C}}[u \otimes u^{-1}, I]} \quad \frac{u \in P_N^{\mathbb{Z}}}{\eta_u \in \mathrm{Hom}_{\mathscr{C}}[I, u \otimes u^{-1}]} \quad \frac{u,v \in P_N^{\mathbb{Z}}}{\sigma_{u,v} \in \mathrm{Hom}_{\mathscr{C}}[u \otimes v, v \otimes u]}$$

$$\frac{t \in T_N}{t_{u,v} \in \mathrm{Hom}_{\mathscr{C}}[u,v]} \quad \forall u,v.(\mathfrak{M}(u) = {}^{\circ}(t) \wedge \mathfrak{M}(v) = (t)^{\circ})$$

$$\frac{\alpha \in \mathrm{Hom}_{\mathscr{C}}[u,v],\ \beta \in \mathrm{Hom}_{\mathscr{C}}[u',v']}{\alpha \otimes \beta \in \mathrm{Hom}_{\mathscr{C}}[u \otimes u', v \otimes v']} \qquad \frac{\alpha \in \mathrm{Hom}_{\mathscr{C}}[u,v],\ \beta \in \mathrm{Hom}_{\mathscr{C}}[v,w]}{\alpha;\beta \in \mathrm{Hom}_{\mathscr{C}}[u,w]}$$

Modulo the axioms that make it into a strict compact closed category:

$$\begin{array}{cc}
\alpha;id_v = \alpha = id_u;\alpha & (\alpha;\beta);\gamma = \alpha;(\beta;\gamma) \\
I \otimes \alpha = \alpha = \alpha \otimes I & (\alpha \otimes \beta) \otimes \gamma = \alpha \otimes (\beta \otimes \gamma) \\
id_u \otimes id_v = id_{u \otimes v} & (\alpha \otimes \alpha');(\beta \otimes \beta') = (\alpha;\beta) \otimes (\alpha';\beta') \\
\sigma_{u,v \otimes w} = (\sigma_{u,v} \otimes id_w);(id_v \otimes \sigma_{u,w}) & \sigma_{u,v};\sigma_{v,u} = id_{u \otimes v} \\
\sigma_{u,u'};(\beta \otimes \alpha) = (\alpha \otimes \beta);\sigma_{v,v'} & \forall \alpha,\beta.(\alpha \in \mathrm{Hom}_{\mathscr{C}}[u,v] \wedge \beta \in \mathrm{Hom}_{\mathscr{C}}\left[u',v'\right] \\
(id_v \otimes \eta_v);(\varepsilon_v \otimes id_v) = id_v & (\eta_v \otimes id_{v^{-1}});(id_{v^{-1}} \otimes \varepsilon_v) = id_{v^{-1}} \\
\varepsilon_{u^{-1}} = \sigma_{u^{-1},u};\varepsilon_u & \eta_{u^{-1}} = \eta_u;\sigma_{u^{-1},u}
\end{array}$$

And the axioms:

$$\begin{array}{cc}
\varepsilon_{u^{-1}} = \sigma_{u^{-1},u};\varepsilon_u & \eta_u;\sigma_{u^{-1},u};\varepsilon_{u^{-1}} = id_I \\
p;t_{u',v'} = t_{u,v};q & \forall p,q.(p \in \mathrm{Hom}_{\mathscr{G}_{P_N}}\left[u,u'\right] \wedge q \in \mathrm{Hom}_{\mathscr{G}_{P_N}}\left[v,v'\right])
\end{array}$$

This is not difficult to prove using the freeness of $\mathscr{G}_{P_N}$.

We now prove that $\mathfrak{F}(-)$ is a functor. Let $N := (P_N, T_N, {}^{\circ}(-)_N, (-)^{\circ}{}_N)$ and $M := (P_M, T_M, {}^{\circ}(-)_M, (-)^{\circ}{}_M)$ be nets, and $\langle f,g \rangle$ a morphism $N \to M$. We want to use the information given by f and g to build a functor between the categories $\mathfrak{F}(N)$ and $\mathfrak{F}(M)$. We sketch this procedure as follows:

- We note that g is an homomorphism $\mathscr{M}^{\mathbb{Z}}_{P_N} \to \mathscr{M}^{\mathbb{Z}}_{P_M}$ of free abelian groups, and hence corresponds uniquely to a function $g' : P_N \to \mathscr{M}^{\mathbb{Z}}_{P_M}$ because of the freeness of $\mathscr{M}^{\mathbb{Z}}_{P_N}$.
- We need a way to lift g to an equivalence class of transition-preserving functors $\mathfrak{F}(N) \to \mathfrak{F}(M)$. As an intermediate step, we start lifting g' to a functor $\mathscr{G}_{P_N} \to \mathfrak{F}(M)$.
- To do this, note that for each right inverse of $\mathfrak{M}$, viz. some $\alpha : \mathscr{M}^{\mathbb{Z}}_{P_M} \to P^{\mathbb{Z}}_M$ such that $\mathfrak{M}(\alpha(u)) = u$, the composition $g'; \alpha : P_N \to P^{\mathbb{Z}}_M$ is a function from P_N to obj $\mathfrak{F}(M)$.
- $\mathfrak{F}(M)$ is clearly strict compact closed and by definition respects the axiom $\eta_u; \sigma_{u^{-1},u}; \varepsilon_{u^{-1}} = id_I$, hence we can use Remark 3.2 to obtain a functor $F : \mathscr{G}_{P_N} \to \mathfrak{F}(M)$.
- We extend the functor $F : \mathscr{G}_{P_N} \to \mathfrak{F}(M)$ to a functor $\mathfrak{F}(\langle f, g\rangle) : \mathfrak{F}(N) \to \mathfrak{F}(M)$, mapping $t_{u,v}$ to $f(t)_{F(u),F(v)}$ and coinciding with F on cups, caps and symmetries.
- Using the equivalent axiomatization for $\mathfrak{F}(N)$ presented above, we note that the $t_{u,v}$ are the components of a transition in $\mathfrak{F}(N)$, and that all transitions in $\mathfrak{F}(N)$ have components of type $t_{u,v}$ for some $t \in T_N$ and objects u, v. On the other hand, $f(t)_{Fu,Fv}$ are the components of a transition in $\mathfrak{F}(M)$, and hence $\mathfrak{F}(\langle f, g\rangle)$ preserves transitions.
- For each suitable choice of α, we get functors that are identified by the relation $\mathfrak{R}_{\mathfrak{F}(N),\mathfrak{F}(M)}$, hence this construction is independent from the choice of α. Moreover, if F, G are two functors $\mathfrak{F}(N) \to \mathfrak{F}(M)$ and $(F, G) \in \mathfrak{R}_{\mathfrak{F}(N),\mathfrak{F}(M)}$ then both F, G are obtained using the procedure sketched above, with different choices of α.

This ensures that what we are doing is well defined , and that to each $\langle f, g\rangle$ corresponds a morphism $\mathfrak{F}(\langle f, g\rangle)$ in the category **GExPetri**. Proving functoriality of $\mathfrak{F}(-)$ is easy noting that the identity morphism $\langle id_{T_N}, id_{\mathscr{M}^{\mathbb{Z}}_{P_N}}\rangle$ is mapped in the equivalence class of the identity functor $\mathfrak{F}(N) \to \mathfrak{F}(N)$, and hence to the identity functor on $\mathfrak{F}(N)$ in **GExPetri**. Composition is preserved noting that $\mathfrak{F}(\langle f; f', g; g'\rangle)$ sends a transition t to $(f; f')(t)$, that is a transition since composition of transition-preserving functors is transition-preserving, and $\mathfrak{R}$ is a congruence with respect to this.

Next step is to prove functoriality of $\mathfrak{U}(-)$. First of all we have to determine how $\mathfrak{U}(-)$ acts on morphisms. Let $\mathscr{C}$ and $\mathscr{D}$ be integer Petri categories. We denote their monoids of objects as $C^{\mathbb{Z}}$ and $D^{\mathbb{Z}}$, respectively. If $F : \mathscr{C} \to \mathscr{D}$ is a representative of a morphism in **GExPetri**, then by definition it preserves structural arrows, meaning that if $u, v \in \mathscr{G}_{\mathscr{C},\nu}$ then $\mathfrak{M}(Fu) = \mathfrak{M}(Fv)$. But then the action of F on objects induces an obvious homomorphism of free abelian groups $F_{pl} : \mathscr{M}^{\mathbb{Z}}_C \to \mathscr{M}^{\mathbb{Z}}_D$, which is clearly independent from the choice of representative F.

Since F is also transition-preserving, then it induces an obvious function $F_{tr} : T_{\mathfrak{U}(\mathscr{C})} \to T_{\mathfrak{U}(\mathscr{D})}$: Each transition t of the net $\mathfrak{U}(\mathscr{C})$ comes by definition from a transition τ in $\mathscr{C}$. F_{tr} then maps each t to the transition of $\mathfrak{U}(\mathscr{D})$ coming from the transition in $\mathscr{D}$ having components $F\tau_{Fu,Fv}$. This is again clearly independent from the choice of F. We then set $\mathfrak{U}(F) := \langle F_{tr}, F_{pl}\rangle$. Functoriality at this point follows trivially from the definitions.

Now we have to prove the adjunction. We define the co-unit $\varepsilon : \mathfrak{F}(\mathfrak{U}(-)) \to (-)$ specifying its components. For each $\mathscr{C} \in$ obj **GExPetri**, the functor $\varepsilon_{\mathscr{C}} : \mathfrak{F}(\mathfrak{U}(\mathscr{C})) \to \mathscr{C}$ is identity on objects and structural arrows. Given a transition $\tau : \pi_{\mathscr{C},\nu} \to \pi_{\mathscr{C},\nu'}$ in the category $\mathscr{C}$, this will correspond to a transition $[\tau]$ in the net $\mathfrak{U}(\mathscr{C})$, and this transition will be again mapped to a family of morphisms $[[\tau]]_{u,v}$ in $\mathfrak{F}(\mathfrak{U}(\mathscr{C}))$, with

$$\mathfrak{M}(u) = {}^{\circ}([\tau]) = \nu \qquad \mathfrak{M}(v) = ([\tau])^{\circ} = \nu'$$

Thanks to the axioms in 3.2, the $[[\tau]]_{u,v}$ define a transition in the category $\mathfrak{F}(\mathfrak{U}(\mathscr{C}))$, and we can define $\varepsilon_{\mathscr{C}}$ as mapping each morphism $[[\tau]]_{u,v}$ to $\tau_{u,v}$. This obviously makes $\varepsilon_{\mathscr{C}}$ transition-preserving, and thus a representative of a morphism in **GExPetri**. Then each $[\varepsilon_{\mathscr{C}}]_{\mathfrak{R}_{\mathscr{C},\mathscr{C}}}$ (here $[-]$ denotes an equivalence class) defines the components of a natrual transformation $\varepsilon : \mathfrak{F}(\mathfrak{U}(-)) \to (-)$, and proving that ε has the co-universal property is just a straightforward check.

The unit $\eta : (-) \to \mathfrak{U}(\mathfrak{F}(-))$ is much easier to define: For each net $N := (P_N, T_N, {}^\circ(-)_N, (-)^\circ{}_N)$, η_N it is the isomorphism $\langle \varphi, id_{\mathscr{M}_{P_N}^{\mathbb{Z}}} \rangle$, where φ is the bijection sending each transition t of N to the transition $[t]$ of $\mathfrak{U}(\mathfrak{F}(N))$ coming from the transition in $\mathfrak{F}(N)$ having components $t_{u,v}$. □

Corollary 3.13. *Call* ***ExPetri***$^{\mathbb{Z}}$ *the full subcategory of* ***GExPetri*** *consisting of integer Petri categories whose morphisms are generated only from compositions and monoidal products of symmetries, cups, caps and transition components, modulo the compact closed categories axioms, the axioms in 3.1 and the axioms in 3.2. Then* ***Petri***$^{\mathbb{Z}} \approx$ ***ExPetri***$^{\mathbb{Z}}$.

Proof. We just have to show that the counit $\varepsilon_{\mathscr{C}}$ of the adjunction $\mathfrak{F}(-) \vdash \mathfrak{U}(-)$ is an iso if and only if $\mathscr{C} \in$ **Petri**$^{\mathbb{Z}}$, which is obvious from the definitions. □

Theorem 3.15. *There is an equivalence* ***Petri***$^{\mathbb{Z}state} \approx$ ***ExPetri***$^{\mathbb{Z}state}$*, and thus an equivalence*

$$\textbf{\textit{ExPetri}}^{\mathbb{N}} \approx \textbf{\textit{Petri}}^{\mathbb{N}} = \textbf{\textit{Petri}}^{\mathbb{Z}state} \approx \textbf{\textit{ExPetri}}^{\mathbb{Z}state} \tag{3.3}$$

Where ***ExPetri***$^{\mathbb{N}}$ *and* ***Petri***$^{\mathbb{N}}$ *are the categories denoted as* ***PSSMC*** *and* ***Petri****, respectively, in [16].*

Proof. This is quite easy. First of all we note that **Petri**$^{\mathbb{Z}\text{state}}$ can obviously be considered a full subcategory of **Petri**$^{\mathbb{Z}}$. This is clear since any multiset can also be considered as a signed multiset, and homomorphisms of free commutative monoids $\mathscr{M}_S^{\mathbb{N}} \to \mathscr{M}_{S'}^{\mathbb{N}}$ can be lifted to a homomorphisms of free abelian groups $\mathscr{M}_S^{\mathbb{Z}} \to \mathscr{M}_{S'}^{\mathbb{Z}}$ via the usual free properties. The equivalence then follows trivially that:

- The image through $\mathfrak{F}(-)$ of a net in **Petri**$^{\mathbb{Z}\text{state}}$ is a positive object in **ExPetri**$^{\mathbb{Z}}$, which is obvious from the definition;
- The image through $\mathfrak{U}(-)$ of a positive object in **ExPetri**$^{\mathbb{Z}}$ is a net in **Petri**$^{\mathbb{Z}\text{state}}$, which is again obvious from the definition.

□