# A compositional framework for Petri nets

Serge Lechenne[1,2], Clovis Eberhart[2,3], and Ichiro Hasuo[2,4]

[1] École normale supérieure Paris-Saclay, France
[2] National Institute of Informatics (NII), Tokyo, Japan
[3] Japanese-French Laboratory for Informatics, IRL 3527, Tokyo, Japan
[4] SOKENDAI (The Graduate University for Advanced Studies), Hayama, Japan

**Abstract.** We define a bidirectional compositional framework for Petri nets based on a line of work about compositionally defining games and computation models. This relies on defining structures with open ends that form interfaces they can be composed along. Together with this syntactic construction, we give a graphical language of morphisms in a PROP and a semantic category that describes the evolution of markings in a Petri net. Compared to previous work, the novelty is that computations in a Petri net are stateful, requiring specific care. This framework allows us to solve reachability compositionally.

**Keywords:** Petri Nets · Category Theory · Compositionality

## 1 Introduction

Petri nets [13,15,16] are a graph theoretical formalisation of concurrency and parallel programming. Agents are represented by *places* and communication between agents by *transitions*. The main question in this formalism is how information circulates between agents. This information is represented by *tokens* that are contained in places and transmitted to other places by transitions.

A crucial problem on Petri nets is *reachability* [16]: Given an initial *marking* (a number of tokens in each place), is there a sequence of transitions that reaches a given final marking? In this paper, we develop a compositional framework for Petri nets that we call *open Petri nets*. The idea is that we equip Petri nets with *open ends*, which are interfaces along which two Petri nets can be composed. The reachability problem naturally becomes an *open-reachability* problem [19], where the question is to know whether, starting from a given marking on the entry interface of the open Petri net, a given marking on its exit interface is reachable. We then define a graphic language for Petri nets based on PROPs, a categorical framework that has been successfully used to model many different graphical structures [1,5,6,7]. It abstracts away all the complexity of the definition of open Petri nets by defining a syntax based on generators and equations. Finally, to solve the open-reachability problem, we define a semantic category into which open Petri nets can be interpreted and where open-reachability can be solved. The base idea is that this semantic category is morally a category of relations between entry markings (markings on the entry interface) and exit markings (markings on the exit interface).

This development follows previous work on different open structures: open parity games [20], open Markov decision processes [21], and open mean-payoff games [22]. However, there are several fundamental differences between previous work and the current article. The first one is that the nature of the structure studied here is different from those of the games studied before. All the games previously studied are sequential in the sense that they are games that consist of passing around a single token. Therefore, when studying a game composed of many sub-games, only one sub-game may play: the game that currently contains the token. In Petri nets, there are many tokens passed around, and the nature of the structure is, therefore, completely concurrent. This means that the semantic composition of Petri nets should be different from those of previous structures.

Another difference is that all the structures we have previously studied are memoryless, in the sense that solving problems on those structures only require the knowledge of which place the token is in. This means that most of the structure can be abstracted away in the semantics. In an open Petri net, a possible exit marking is dependent on the entry marking, but also on the internal state of the Petri net, which is therefore relevant to our approach. In this sense, we may say that the properties studied in the previous papers in this line of work were static, while we study a dynamic property here. For these reasons, the semantic category of open Petri nets differs from those of the previous papers.

One advantage of our approach is that we derive a complex bidirectional framework, expressed in the language of compact closed categories (CompCCs), from a simpler unidirectional framework, expressed in the language of traced symmetric monoidal categories (TSMCs). This is done for free using the well-known Int-construction [10]. On the level of Petri nets, this means that we derive a category **oPN** of open Petri nets from a category **roPN** of *rightward open Petri nets* where all open ends point towards the right. We do the same at the level of semantics, the semantic category $\mathbb{S}_r$ for the unidirectional framework is lifted to $\mathbb{S}$ in the bidirectional framework using the same construction. Moreover, the Int-construction also lifts the interpretation of open Petri nets into the semantic category from the unidirectional level ($\mathcal{S}_r$) to the bidirectional one ($\mathcal{S}$). This is the result of Lemma 6, which is illustrated in Figure 1.

$$\mathbf{roPN} \xrightarrow{\quad\mathcal{S}_r\quad} \mathbb{S}_r \qquad \mathbf{oPN} = \mathrm{Int}(\mathbf{roPN}) \xrightarrow{\quad\mathcal{S} = \mathrm{Int}(\mathcal{S}_r)\quad} \mathbb{S} = \mathrm{Int}(\mathbb{S}_r)$$

**Fig. 1.** Lifting from the unidirectional framework to the bidirectional framework

*Related Work* As previously mentioned, this work is a continuation of a line of work on compositional structures [20,21,22], which aims at using compositionality to design faster algorithms to compute properties of structures.

Our graphical language is based on PROPs, which are a specific type of traced symmetric monoidal categories. Many lines of work have used TSMCs as

graphical languages. This dates back to [12], see [17] for a survey. For graph-like structures, [8] considers acyclic graphs. In recent years, PROPs have been used to study many graphical structures such as networks [1], signal flow diagrams [5], and quantum graphic calculi [6,7].

Many other works describe compositional approaches to Petri nets. In [3,2], open Petri nets are defined as cospans and composed by pushout. While this is close to our presentation, the fact that open Petri nets are cospans means that they are not exactly graphical objects. In [4], the authors define a graphical language for linear systems. By interpreting this language in the right category, they can encode Petri nets. While their approach is also based on string diagrams, it is a bit farther from graphical intuition than ours, as places, transitions, and their interactions, are encoded as non-trivial string diagrams.

In [19,18], the authors develop a compositional approach to Petri nets, called "Petri nets with boundaries". One advantage is that the categorical structure can be used to derive efficient algorithms. Petri nets with boundaries, in the fashion of a compositional approach, simplify model checking by applying a "divide and conquer" approach, as explained in [14]. We here present a systematic derivation of a compositional Petri net framework, separating the essence of Petri nets with boundaries from specific details. Our framework, called "open Petri nets", is derived systematically, while Petri nets with boundaries were derived in a somewhat ad hoc manner. This approach is the closest to ours, and we offer a more thorough comparison in the article.

*Plan* In Section 2, we give some reminders on Petri nets and introduce rightward open Petri nets. Then, in Section 3, we study the structure of rightward open Petri nets and show that they form a traced symmetric monoidal category (TSMC), and compare them to Petri nets with boundaries. Section 4 is dedicated to defining the semantic category and proving that it is also a TSMC, as well as the interpretation of rightward open Petri nets into the semantic category and prove that it respects the TSMC structure. Finally, we derive the bidirectional framework in Section 5 as well as a syntax to inductively compute on open Petri nets, and use this to solve the open-reachability problem compositionally.

*Notations and Prerequisites* We assume that the reader is familiar with some basic notions of category theory, especially monoidal categories. For all $k \in \mathbb{N}$, we write $[k]$ for the set $\{1, 2, \dots, k\}$. Given a proposition $P$, we write $\delta_P$ for the Kronecker symbol, i.e., $\delta_P = 1$ if $P$ holds, and $\delta_P = 0$ otherwise. Given two functions $f\colon X \to Y$ and $f'\colon X' \to Y'$, we name $f + f'\colon X + X' \to Y + Y'$ the function obtained by universal property of coproduct. Given two pairs of natural numbers $n = (a, b)$ and $m = (c, d)$, we write $n + m$ for the pair $(a + c, b + d)$.

## 2 Rightward Open Petri Nets

After some brief reminders on Petri nets, this section describes our first contribution, *rightward open Petri nets*, which are a compositional approach to Petri nets. For Petri nets, we follow the notations and definitions from [19].

### 2.1 Petri nets

**Definition 1 (Petri nets).** *A* Petri net *is a tuple* $(P, T, {}^\bullet(\_), (\_)^\bullet)$ *where*

- $P$ *(resp.* $T$*) is a set whose elements are called* places *(resp.* transitions*),*
- ${}^\bullet(\_)$ *and* $(\_)^\bullet$ *are functions* $T \to \mathcal{P}(P)$*.*

Petri nets admit a graph-theoretical representation [13,15,16] where the places are drawn as circles and the transitions as rectangles, sometimes with their labels written around them. The function ${}^\bullet(\_)$ is represented by arrows from place to transitions, and the function $(\_)^\bullet$ by arrows from transitions to places. For example, the Petri net with $P = \{p_1, p_2, p_3\}$, $T = \{t\}$, ${}^\bullet t = \{p_1\}$, and $t^\bullet = \{p_2, p_3\}$ can be represented as on the left of Figure 2.

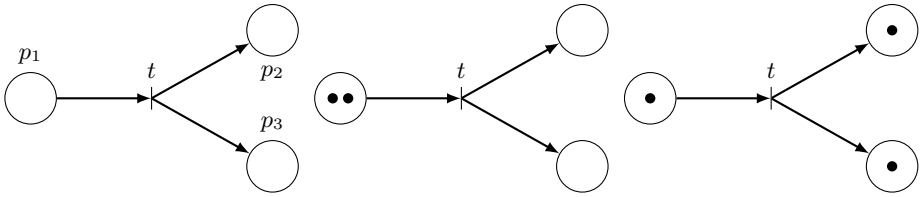


**Fig. 2.** A Petri net, a marking of it, and the marking reached after firing transition $t$

The data, named tokens, is exchanged through the transitions from place to place. Intuitively, the data does not represent any physical object but information that agents can freely exchange, create or destroy. This information circulating in a network is modelled in the form of a marking function.

**Definition 2 (marking).** *A* marking function *(or* marking*) on a Petri net is a function* $\mu : P \to \mathbb{N}$*.*

Information is transmitted between agents by selecting a transition $t$, consuming one token from every place leading to $t$ (i.e., every place in ${}^\bullet t$) and placing one token in every place reached by $t$ (i.e., every place in $t^\bullet$).

**Definition 3 (enabling, firing).** *Given a Petri net, a transition* $t$*, and a marking* $\mu$*, we say that* $t$ *is* enabled *by* $\mu$ *if* $\forall x \in {}^\bullet t, \mu(x) \geq 1$*.*

*A transition* $t$ *enabled by a marking* $\mu$ *can modify it into a new marking* $\mu'$ *defined by* $\mu'(x) = \mu(x) + \delta_{x \in t^\bullet} - \delta_{x \in {}^\bullet t}$*. This operation is called* firing $t$*, and is denoted* $\mu \xrightarrow{t} \mu'$*.*

In the definition above, $\mu'$ removes one token from $\mu$ in all places in $t^\bullet$, and adds one to all places in ${}^\bullet t$.

We extend the notation of firing to sequences $s = t_1, \ldots, t_n \in T^*$ and write $\mu \xrightarrow{s} \mu'$ to mean that each $t_i$ is enabled on the marking reached from $\mu$ after firing all the previous $t_j$'s and that $\mu'$ is reached after firing $t_n$. We will also write $t(\mu)$ the marking $\mu'$ (or $t_N(\mu)$ when the Petri net $N$ is not explicit).

We now illustrate how the graphical representation accommodates markings and firings of transitions. Tokens are represented as black dots, and a marking $\mu$ is represented by placing $\mu(x)$ tokens in each place $x$. For example, remembering the example Petri net on the left of Figure 2, the drawing in the middle of Figure 2 represents the marking $\mu$ defined by $\mu(p_1) = 2$ and $\mu(p_2) = \mu(p_3) = 0$. The transition $t$ is enabled by $\mu$ in this net, and firing it outputs the marking on the right of Figure 2.

### 2.2 Rightward Open Petri Nets

We now define rightward open Petri nets, which have open ends along which they can be composed. We then adapt all the definitions of Petri nets to rightward open Petri nets.

**Definition 4 (rightward open Petri net).** *A* rightward open Petri net *is a tuple* $N = (m, n, P, T, {}^\circ(\_), (\_)^\circ, {}^\bullet(\_), (\_)^\bullet)$ *where*

- $m = (m_b, m_r)$ *and* $n = (n_b, n_r)$ *are pairs of natural numbers called the (*left *and* right*)* interfaces *of* $N$*,*
- $P$ *(resp.* $T$*) is a set whose elements are called* places *(resp.* transitions*),*
- ${}^\circ(\_) : m_r \to T + n_r, m_b \to P + n_b$ *is a function such that* $\forall j \in [n_b] + [n_r], i_1, i_2 \in [m_b] + [m_r], {}^\circ(i_1) = j = {}^\circ(i_2) \Rightarrow i_1 = i_2$*,*
- $(\_)^\circ : P \to \mathcal{P}([n_r]), T \to \mathcal{P}([n_b])$ *is a function such that* $\forall t, t' \in T \cup P, (t)^\circ \cap (t')^\circ \neq \emptyset \Rightarrow t = t'$*,*
- ${}^\bullet(\_)$ *and* $(\_)^\bullet$ *are functions* $T \to \mathcal{P}(P)$*.*

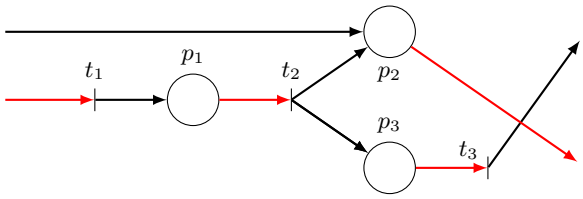


**Fig. 3.** A rightward open Petri net with interfaces $(1, 1)$ and $(1, 1)$

We now explain in detail what each element of the tuple represents. First of all, $P$, $T$, ${}^\bullet(\_)$, and $(\_)^\bullet$ represent the same elements as for regular Petri nets. The left and right interfaces, $m$ and $n$, respectively, encode the number of open ends on the left-hand and right-hand side of the rightward open Petri net. They are pairs of numbers because open ends that point to places are different from open ends that point to transitions; $m_b$ and $n_b$ represent open ends that point to places, while $m_r$ and $n_r$ represent open ends that point to transitions. The subscripts $b$ and $r$ stand for "black" and "red" respectively, corresponding to the

graphical representation we show below. The constraints on ${}^\circ(\_)$ and $(\_)^\circ$ mean that there is at most one arrow pointing to each open end on the right.

The function ${}^\circ(\_)$ (which is formally two functions, but for which we use the same symbol) encodes how each open end on the left (in $[m_b]$ or $[m_r]$) is connected to a place, a transition, or an open end on the right (i.e., in $[n_b]$ or $[n_r]$). The type of the function ensures that black open ends only point to either places or black open ends on the right, and red open ends only to either transitions or red open ends on the right.

The function $(\_)^\circ$ (which is again formally two functions) encodes the connection of places and transitions to open ends on the right.

We also adapt the graphical representation of Petri nets to rightward open Petri nets by graphically drawing ${}^\circ(\_)$ as arrows coming from the left open ends and leading to places, transitions, or to the right open ends. We draw $(\_)^\circ$ similarly from places and transitions to right open ends. Figure 3 shows an example of the graphic representation of a rightward open Petri net. It describes the rightward open Petri net with $m = n = (1, 1)$, $P = \{p_1, p_2, p_3\}$, $T = \{t_1, t_2, t_3\}$, ${}^\circ 1_b = p_2$, ${}^\circ 1_r = t_1$, $p_1^\circ = p_2^\circ = t_1^\circ = t_3^\circ = \emptyset$, $p_2^\circ = 1_r$, $t_3^\circ = 1_b$, ${}^\bullet t_1 = \emptyset$, ${}^\bullet t_2 = \{p_1\}$, ${}^\bullet t_3 = \{p_3\}$, $t_1^\bullet = \{p_1\}$, $t_2^\bullet = \{p_2, p_3\}$, and $t_3^\bullet = \emptyset$, where $1_b$, $1_r$ are used to distinguish between black and red open ends (even though it is unambiguous even without this distinction because of the typing).

Open ends are numbered, and verticality in the graphical representation encodes the order between open ends. Note that we always draw black open ends above red ones for interfaces, but that is just a graphical convention: there is no order between open ends of different colours.

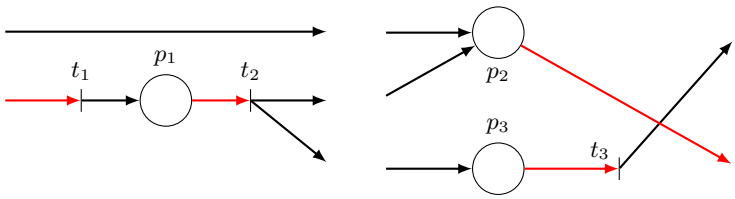


**Fig. 4.** Two roPNs with interfaces $(1, 1) \to (3, 0)$ and $(3, 0) \to (1, 1)$ respectively

We can now define configurations, which are the counterparts of markings for traditional Petri nets. We can then define the way roPN configurations evolve. Like Petri nets, they evolve through firing transitions. The main difference with traditional Petri nets is that they can also evolve by tokens "sliding" from the entry interface to the exit interface.

**Definition 5 (Markings, configurations).** *Given a rightward open Petri net with interfaces $m$ and $n$ and a set $P$ of places, a* marking *is a function $\mu \colon P \to \mathbb{N}$, an* entry marking *is a function $\mu_\mathrm{i} \colon [m_b] + [m_r] \to \mathbb{N}$, and an* exit marking *is a function $\mu_\mathrm{o} \colon [n_b] + [n_r] \to \mathbb{N}$. A* configuration *is an element $(\mu_\mathrm{i}, \mu, \mu_\mathrm{o})$ of $\mathbb{N}^{m_b+m_r} \times \mathbb{N}^P \times \mathbb{N}^{m_b+m_r}$.*

**Definition 6 (firing, sliding).** *Given a rightward open Petri net $N$, a configuration $c = (\mu_{\mathrm{i}}, \mu, \mu_{\mathrm{o}})$, and a transition $t$, we say that $t$ is* enabled *on $c$ if for all places $p \in {}^\bullet t$, $\mu(p) > 0$, and for all entries $i \in [m_r]$ such that ${}^\circ i = t$, $\mu_{\mathrm{i}}(i) > 0$. Similarly, an entry $i \in [n_b]$ is enabled on $c$ if $\mu(i) > 0$, and an exit $j \in [m_r]$, for which there exists $p \in P$ with $j \in p^\circ$, is enabled if $\mu(p) > 0$.*

*The* firing *of $t$ enabled on $c$ turns it into $(\mu'_{\mathrm{i}}, \mu', \mu'_{\mathrm{o}})$ such that $\mu'_{\mathrm{i}}(i) = \mu_{\mathrm{i}}(i) - \delta_{{}^\circ i = t}$, $\mu'(p) = \mu(p) - \delta_{p \in {}^\bullet t} + \delta_{p \in t^\bullet}$, and $\mu'_{\mathrm{o}}(j) = \mu_{\mathrm{o}}(j) + \delta_{j \in t^\circ}$.*

*The* sliding *of entry $i$ enabled on $c$ turns it into $(\mu'_{\mathrm{i}}, \mu', \mu'_{\mathrm{o}})$ with $\mu'_{\mathrm{i}}(i') = \mu_{\mathrm{i}}(i') - \delta_{i'=i}$, $\mu'(p) = \mu(p) + \delta_{{}^\circ i = p}$, $\mu'_{\mathrm{o}}(j) = \mu_{\mathrm{o}}(j) + \delta_{{}^\circ i = j}$.*

*The* sliding *of exit $j$ enabled on $c$ turns it $(\mu'_{\mathrm{i}}, \mu', \mu'_{\mathrm{o}})$ with $\mu'_{\mathrm{i}} = \mu_{\mathrm{i}}$, $\mu'(p) = \mu(p) - \delta p^\circ = j$, and $\mu'_{\mathrm{o}}(j') = \mu_{\mathrm{o}}(j') + \delta_{j=j'}$.*

In this definition, sliding is rather straightforward: tokens may slide freely between open ends and places (as these correspond to transitions being fired outside of the open Petri net). To fire a transition, all arrows pointing to it must contain at least one token: both from places and from open ends.

We denote by $t(c)$ the configuration obtained by firing transition $t$ from $c$. Similarly, we denote by $l_i(c)$ (resp. $r_j(c)$) the configuration obtained by sliding the $i$th entry (resp. $j$th exit) from $c$.

## 3 The Categorical Structure of roPN

We now move on to explicitating the categorical structure of rightward open Petri nets, viewed as arrows between interfaces. We explicitate the categorical structure of rightward open Petri nets, using the language of monoidal categories [11,9]. We will then compare our approach and the work done in [19,18].

### 3.1 The Category of Rightward Open Petri Nets

We start by establishing their structure as a category by defining the identities and composition. When a Petri net $N$ has interfaces $m$ and $n$, we write $N \colon m \to n$. Indeed, in this section we define the category **roPN** or rightward open Petri nets whose objects are interfaces are morphisms are roPNs.

**Definition 7 (composition).** *Given $N \colon m \to n$ and $N' \colon n \to k$, we name $N ; N' \colon m \to k$ the rightward open Petri net made from the following data: $P_{N;N'} = P \cup P'$, $T_{N;N'} = T \cup T'$,*

$$
{}^\bullet t_{N;N'} = \begin{cases} {}^\bullet t_N & \text{if } t \in T \\ {}^\bullet t_{N'} \cup \{p \in P \mid \exists j \in [m_r] \,.\, j \in p^\circ_N \wedge {}^\circ j_{N'} = t\} & \text{otherwise,} \end{cases}
$$

$$
t^\bullet_{N;N'} = \begin{cases} t^\bullet_N \cup \{{}^\circ j_{N'} \in P' \mid j \in t^\circ_N\} & \text{if } t \in T \\ t^\bullet_{N'} & \text{otherwise,} \end{cases}
$$

$$
{}^{\circ}i_{N;N'} = \begin{cases} {}^{\circ}({}^{\circ}i_N)_{N'} & \text{if } {}^{\circ}i_N \in [j_b] \\ {}^{\circ}i_N & \text{otherwise,} \end{cases}
$$

$$
x^{\circ}_{N;N'} = \begin{cases} x^{\circ}_{N'} & \text{if } {}^{\circ}(x^{\circ}_N)_{N'} \in P \cup T \\ \{{}^{\circ}j_{N'} \in [k_b] + [k_r] \mid j \in x^{\circ}_N\} & \text{otherwise.} \end{cases}
$$

Checking that ${}^{\circ}(\_)_{N;N}$ and $(\_)^{\circ}_{N;N}$ satisfy the condition is easy.

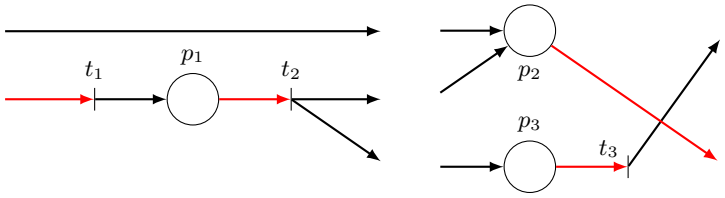


**Fig. 5.** Two roPNs with interfaces $(1,1) \to (3,0)$ and $(3,0) \to (1,1)$ respectively

The definitions of $(\_)^{\circ}$, ${}^{\circ}(\_)$, ${}^{\bullet}(\_)$, and $(\_)^{\bullet}$ encode the fact that edges that point to and from the middle interface get merged in the composition $N; N'$. Figure 5 shows two roPNs whose composition is the roPN in Figure 3.

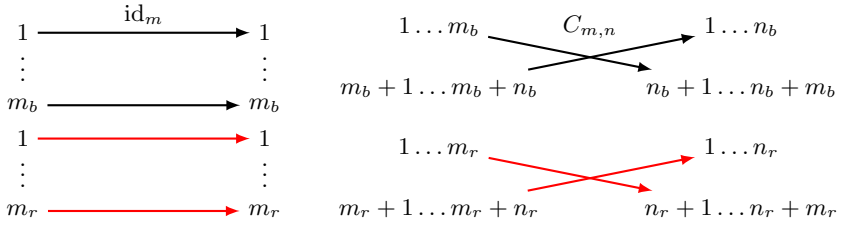


**Fig. 6.** Identities and swaps

**Definition 8 (identities, swaps).** *Given an interface $m$, the* identity *roPN* $\mathrm{id}_m \colon m \to m$ *is given by the following data: $P = T = \emptyset$, ${}^{\circ}i = i$ (the rest of the data being trivial since $P = T = \emptyset$).*

*Given two interfaces $m$ and $n$, the* swap $C_{m,n} \colon m + n \to n + m$ *is given by the following data: $P = T = \emptyset$, ${}^{\circ}i$ is $i + m_b$ if $i \leq m_b$ and $i - m_b$ otherwise for $i \in [m_b + n_b]$, and ${}^{\circ}i$ is $i + m_r$ if $i \leq m_r$ and $i - m_r$ otherwise for $i \in [m_r + n_r]$.*

The identities and swaps are illustrated in Figure 6. We can now organise rightward open Petri nets into a category **roPN**.

**Definition 9 (roPN).** *We name* **roPN** *the category whose objects are pairs of elements $(m_r, m_b) \in \mathbb{N} \times \mathbb{N}$ and arrows $N : (m_r, m_b) \to (n_r, n_b)$ are rightward*

*open Petri nets with the corresponding numbers of open ends. Composition is defined by* ; *and the identity of* $m \in \mathbb{N} \times \mathbb{N}$ *is the rightward open Petri net* $id_m$.

### 3.2 Monoidal Categorical Structure

We now further explore the categorical structure of **roPN**, showing that it is a traced symmetric monoidal category.

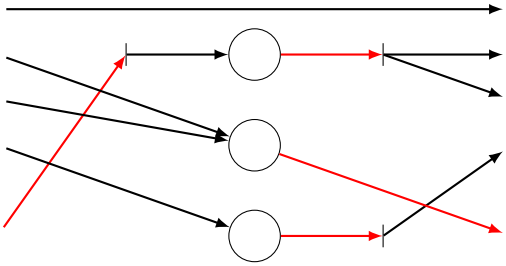

**Fig. 7.** An roPN with interfaces $(4, 1)$ and $(4, 1)$

**Definition 10 (tensor product).** *Given* $N \colon m \to n$ *and* $N' \colon m' \to n'$, *we define* $N \otimes N' \colon m + m' \to n + n'$ *the rightward open Petri net given by the following data:* $P_{N \otimes N'} = P \cup P'$, $T_{N \otimes N'} = T \cup T'$, ${}^{\circ}(\_)_{N \otimes N'} = {}^{\circ}(\_)_N + {}^{\circ}(\_)_{N'}$, $(\_)^{\circ}_{N \otimes N'} = (\_)^{\circ}_N + (\_)^{\circ}_{N'}$, ${}^{\bullet}(\_)_{N \otimes N'} = {}^{\bullet}(\_)_N + {}^{\bullet}(\_)_{N'}$, *and* $(\_)^{\bullet}_{N \otimes N'} = (\_)^{\bullet}_N + (\_)^{\bullet}_{N'}$.

Figure 7 shows the tensor product of the two roPNs from Figure 5. Note that this graphically corresponds to stacking the roPNs (except for the interfaces, where the red ends are drawn below the black ones, but which is only a convention).

**Lemma 1. roPN** *forms a symmetric monoidal category with* $+$ *as tensor on objects, and* $\otimes$ *on morphisms, and* $C_{m,n}$ *as the symmetries.*

### 3.3 Trace Operator

Our goal is to derive a bidirectional framework from the unidirectional one. To do this, we need to show that **roPN** is traced. We first introduce some notions, as defining the trace is trickier than for composition or tensor. Graphically, the trace operator "loops" over the first black and red inputs and outputs of the Petri net. Figure 8 shows an example and hints that the definition is non-trivial, as the looping may connect inputs and outputs in complex ways.

This example illustrates that if something (a place, a transition, or a left open end) is connected to one of the right open ends affected by the trace, simply looking at the value of the corresponding open end by $(\_)^{\circ}$ is not enough,

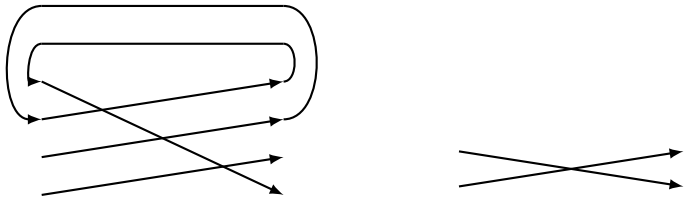

**Fig. 8.** A roPN with interfaces $(4,0)$ and $(4,0)$, traced along $(2,0)$ and the result

since we can "stay" in the loop. Here, the third open end is connected to the second, which is connected to the first, which is finally connected to the fourth.

Hence, given a rightward open Petri net $N : (m_b+p, m_r+q) \to (n_b+p, n_r+q)$, we need to define functions $\phi_p$ (for the $p$ black open ends that are looped) that output the final open end that does not lead back to $[p]$ (in the loop). To ensure such a function $\phi_p$ can be defined, we must prove that such an exiting open end exists. The following lemma ensures this.

**Lemma 2.** *Let $j \in [p]$, then there is at most one $i \in [m_b + p]$ such that ${}^\circ i_N = j$.*

This is easily proven using the conditions imposed over $(x)^\circ_N$. Finally, because $[p]$ is finite, we define $\phi_p(x)$ by induction: $\phi_p(x) = x$ if ${}^\circ x \notin [p]$, and $\phi_p(x) = \phi_p({}^\circ x)$ otherwise. This definition is well-founded because of Lemma 2 and the fact that $[p]$ is finite. We define $\phi_q$ equivalently.

**Definition 11 (Trace operator).** *Given a rightward open Petri net $N : m + p \to n + p$, we define $N' = \mathrm{Tr}^p_{m,n}(N) \colon m \to n$ as the rightward open Petri net with the following data: $P' = P$, $T' = T$,*

$$
\begin{aligned}
{}^\circ(x)' &= \begin{cases} {}^\circ\phi_p(x) & \text{if } x \in [m_b] \\ {}^\circ\phi_q(x) & \text{if } x \in [m_r], \end{cases} \\
(x)^{\circ\prime} &= \{{}^\circ(\phi_p(h) \mid h \in x^\circ \cap [p]\} \cup \{{}^\circ\phi_q(h) \mid h \in x^\circ \cap [q]\} \cup (x^\circ \setminus ([p] \cup [q])) \\
{}^\bullet(t)' &= \{x \in P \mid \exists h \in [q], h \in x^\circ \wedge \phi_q(h) \in {}^\bullet t\} \cup {}^\bullet t \\
(t)^{\bullet\prime} &= \{x \in P \mid \exists h \in [q]\,, h \in t^\circ \wedge {}^\circ\phi_p(h) = x\} \cup t^\bullet.
\end{aligned}
$$

**Lemma 3 (trace).** $\mathrm{Tr}^{p,q}_{m,n}(-)$ *defines a trace over* **roPN**.

**Theorem 1. roPN** *is a traced symmetric monoidal category (TSMC).*

### 3.4 Comparison with Petri Nets with Boundaries

Now that the categorical structure is spelt out, we move on to pointing out the syntactic and operational differences between **roPN** and **PNB**, the category of Petri net with boundaries defined in [19].

We first give out a quick summary of Petri net with boundaries. A Petri net with boundaries is thought of as a Petri net with extra structure, namely

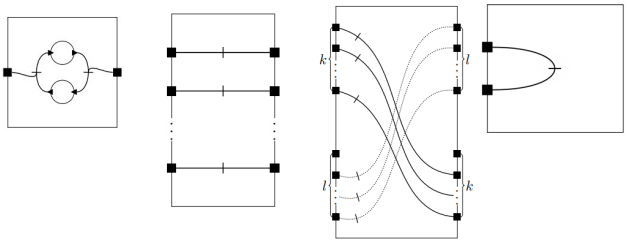


**Fig. 9.** Some examples of Petri nets with boundaries

a net $(P, T, {}^\bullet(\_), (\_)^\bullet)$ as defined in our work to which is added two finite ordinals that act as the interface, and two accessibility functions, called "in" and "out" (those are not the original notations introduced in [19]), encoding how the interface is linked to transitions. Petri nets with boundaries admit a graphical representation, as shown in Figure 9 (taken from [19]), where the boundaries are represented as black squares located on the edges of a box enclosing the Petri net. The two accessibility functions are represented as regular edges, connecting the black squares and the transition.

The first main difference, and the most important one, is that the boundaries representing the interface are connected to transitions *only*, meaning they are not leading in or out of the Petri net by default. Here, it is the function in (resp. out) that makes one boundary in the interface leading to or coming from transitions.

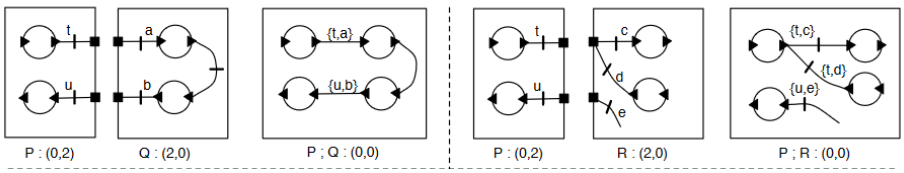


**Fig. 10.** composing Petri net with boundaries

The fact that boundaries are only connected to transitions also affects the composition of Petri nets with boundaries: we have chosen our definition of a rightward open Petri net to have an "implicit" interface, which allows us to follow graphical intuition of fusing edges. In the Petri net with boundary framework, the fusing is intuitively done on the boundaries, as demonstrated in the two examples shown in Figure 10 (taken from [18]). However, when fusing the edges, the transitions these edges are connected to also need to be fused (this is called

a minimal synchronisation in [19]). This fusing process is destructive in regard to the structure of the Petri net, as transitions can be either created or deleted, like in the rightmost net in Figure 10. For example, the middle two Petri nets with boundaries in Figure 9 are the categorical identities and crossing of the category **PNB**, and they only satisfy the required naturality equations *up to isomorphism*, making the categorical structure lax monoidal, contrary to **roPN**.

Another consequence of this fact is that by connecting two boundaries of the same interface via a transition, one can create the counit of **PNB** (depicted as the rightmost net in Figure 9). As the snake equation is satisfied, this means that **PNB** is a compact closed (lax) category, where the trace is defined by the canonical construction described in [10]. In comparison, in the case of open Petri nets, the compact closed structure is derived from the more primitive traced symmetric monoidal structure that describes the unidirectional case.

## 4 The Semantic Structure

In this section, we define the semantic category $\mathbb{S}_r$, prove that it has a traced symmetric monoidal structure, define a semantic functor $\mathcal{S}_r \colon \mathbf{roPN} \to \mathbb{S}_r$ and prove that it respects that traced symmetric monoidal structure.

### 4.1 The Semantic Category $\mathbb{S}_r$

The main idea is that what can be observed of an open Petri net is how many tokens enter and exit it through its interfaces. However, that is not enough to characterise the behaviour of an open Petri net. Indeed, some token can be left in the net, therefore modifying the behaviour of said net. The marking acts like an internal state to the Petri net, and that is how we represent it in the semantics.

**Definition 12 (The semantic category $\mathbb{S}_r$).** *We denote by $\mathbb{S}_r$ the* semantic category*. Its objects are pairs $(n_b, n_r)$ of natural numbers. The morphisms from $(m_b, m_r)$ to $(n_b, n_r)$ are pairs of a set $Q$ (called the set of* internal states*) and a function from $\mathbb{N}^{m_b+m_r} \times Q \times \mathbb{N}^{n_b+n_r}$ to $\mathcal{P}(\mathbb{N}^{m_b+m_r} \times Q \times \mathbb{N}^{n_b+n_r})$ that satisfies the following conditions*

- *(reflexivity)* $(\mu_\mathrm{i}, q, \mu_\mathrm{o}) \in f(\mu_\mathrm{i}, q, \mu_\mathrm{o})$,
- *(transitivity) if* $(\mu'_\mathrm{i}, q', \mu'_\mathrm{o}) \in f(\mu_\mathrm{i}, q, \mu_\mathrm{o})$ *and* $(\mu''_\mathrm{i}, q'', \mu''_\mathrm{o}) \in f(\mu'_\mathrm{i}, q', \mu'_\mathrm{o})$*, then* $(\mu''_\mathrm{i}, q'', \mu''_\mathrm{o}) \in f(\mu_\mathrm{i}, q, \mu_\mathrm{o})$,
- *(additivity)* $(\mu'_\mathrm{i}, q, \mu'_\mathrm{o}) \in f(\mu_\mathrm{i}, q, \mu_\mathrm{o})$ *iff* $(\mu'_\mathrm{i} + \tilde{\mu}_\mathrm{i}, q, \mu'_\mathrm{o} + \tilde{\mu_\mathrm{o}}) \in f(\mu_\mathrm{i} + \tilde{\mu}_\mathrm{i}, q, \mu_\mathrm{o} + \tilde{\mu_\mathrm{o}})$
- *(monotonicity) if* $(\mu'_\mathrm{i}, q, \mu'_\mathrm{o}) \in f(\mu_\mathrm{i}, q, \mu_\mathrm{o})$*, then* $\mu'_\mathrm{i} \leq \mu_\mathrm{i}$ *and* $\mu_\mathrm{o} \leq \mu'_\mathrm{o}$.

For simplicity, we sometimes just write $\mathbb{N}^m$ for $\mathbb{N}^{m_b+m_r}$. We write $(\mu_\mathrm{i}, q, \mu_\mathrm{o})$ for an element in $\mathbb{N}^m \times Q \times \mathbb{N}^n$, where $\mu_\mathrm{i}$ represents the entry marking, $q$ the internal state, and $\mu_\mathrm{o}$ the exit marking. Note that the function is equivalent to a relation on $\mathbb{N}^m \times Q \times \mathbb{N}^n$, and we write $(\mu_\mathrm{i}, q, \mu_\mathrm{o}) \to_f (\mu'_\mathrm{i}, q', \mu'_\mathrm{o})$ to denote that these two elements are related by $f$ (i.e., that $(\mu'_\mathrm{i}, q', \mu'_\mathrm{o}) \in f((\mu_\mathrm{i}, q, \mu_\mathrm{o}))$).

The idea is that two elements $(\mu_\mathrm{i}, q, \mu_\mathrm{o})$ and $(\mu'_\mathrm{i}, q', \mu'_\mathrm{o})$ are related by the interpretation of a Petri net $N$ if, starting with an entry marking $\mu_\mathrm{i}$ in internal state $q$, and assuming that $N$ has already output $\mu_\mathrm{o}$, then there exists a sequence of firings that consumes tokens from the entry marking until exactly $\mu'_\mathrm{i}$ is left, modifying the internal state to $q'$, and outputting tokens to reach exactly $\mu'_\mathrm{o}$.

There are two differences between the semantic category here and those of previous work. The first one is insignificant: in previous work, the objects were natural numbers rather than pairs, as there was a single type of edges. More importantly, in previous work, the semantic category was always defined via a monad, but this is not the case here. The current definition may seem too complicated, but we argue that such complexity is necessary. We go through several simpler ideas and explain why they fail.

As mentioned above, the internal state of the Petri net dynamically modifies its behaviour, hence if we considered a morphism from $m$ to $n$ to only be a function $\mathbb{N}^m \to \mathcal{P}(\mathbb{N}^n)$ (which would fall under the monad construction), it would be impossible to interpret roPNs into this semantic category.

A second idea would then be to define morphisms as functions $f\colon \mathbb{N}^m \times Q \to \mathcal{P}(Q \times \mathbb{N}^n)$, which solves the problem of taking the internal state of the Petri net into account, and it would be possible to interpret rightward-open Petri nets into this semantic category. The problem lies with what it means for $(q', \mu_\mathrm{o})$ to be in $f(\mu_\mathrm{i}, q)$. It would mean that $\mu_\mathrm{o}$ is reachable by consuming all the tokens from $\mu_\mathrm{i}$. However, it would then be impossible to define a trace operator that is compatible with that of roPNs, since some tokens should remain in the entry marking between two loops in order to reach some exit markings. Hence, this idea does not lead to a trace that is compatible with that of **roPN**.

This explains why we need to remember the entry marking even in the output of the function. Technically, it is possible to define the semantics as a category of functions of type $\mathbb{N}^m \times Q \to \mathcal{P}(\mathbb{N}^m \times Q \times \mathbb{N}^n)$, but we chose to add the exit marking to the argument of the function, making it symmetric and therefore, equivalent to a category of relations.

To define $\mathbb{S}_r$, we first need to define the identities and composition.

**Definition 13 (Identities, composition of $\mathbb{S}_r$).** *The identity on $n$ is $\mathrm{id}_n\colon n \to n$ defined as the set $\{*\}$ of internal states and the relation $(\mu_\mathrm{i}, *, \mu_\mathrm{o}) \to_{\mathrm{id}_n} (\mu'_\mathrm{i}, *, \mu'_\mathrm{o})$ if and only if $\mu'_\mathrm{i} \leq \mu_\mathrm{i}$ and $\mu_\mathrm{i} + \mu_\mathrm{o} = \mu'_\mathrm{i} + \mu'_\mathrm{o}$.*

*Let $f\colon n \to m$ and $g\colon m \to p$ be two morphisms in $\mathbb{S}_r$, we define their composition $f;g$ as having $Q_f \times Q_g$ as its set of internal states, and $(\mu_\mathrm{i}, (q_f, q_g), \mu_\mathrm{o}) \to_{f;g} (\mu'_\mathrm{i}, (q'_f, q'_g), \mu'_\mathrm{o})$ if and only if there exists $\mu^*$ such that $(\mu_\mathrm{i}, q_f, 0^m) \to_f (\mu'_\mathrm{i}, q'_f, \mu^*)$ and $(\mu^*, q_g, \mu_\mathrm{o}) \to_g (0^m, q'_g, \mu'_\mathrm{o})$. With this data, $\mathbb{S}_r$ forms a category.*

The idea of identities is that they connect each entry $i$ to its corresponding exit, and each wire can slide any number of tokens from the entry marking to the exit one. Note that additivity and monotonicity are necessary to prove that $\mathbb{S}_r$ forms a category.

We depict how composition intuitively works in Figure 11. In the configuration on the left, before the Petri net starts evolving, there should be no "floating"

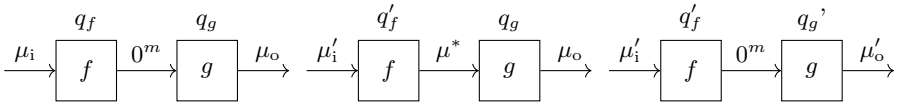


**Fig. 11.** Composition in the semantic category $\mathbb{S}_r$

tokens on the interface between $f$ and $g$. The execution starts from $f$, which gives a new initial marking and internal states, as well as tokens on the interface between $f$ and $g$, as depicted in the middle of Figure 11. Then $g$ is executed, but its execution should consume all the tokens on the interface, so that there are no tokens floating on the interface after execution, as depicted on the right of Figure 11.

### 4.2 The Monoidal Categorical Structure

We now show that $\mathbb{S}_r$ has a monoidal structure as a first step towards showing that it is a TSMC.

**Definition 14 (tensor product of $\mathbb{S}_r$).** *Given $n, m \in \mathbb{N}$, we define $\otimes$ as the addition. Given $f_1 \colon m_1 \to n_1$ and $f_2 \colon m_2 \to n_2$, we define $f_1 \otimes f_2 \colon m_1 + m_2 \to n_1 + n_2$ as the morphism with the internal state $Q_1 \times Q_2$ and $(\mu_{\mathrm{i},1} \otimes \mu_{\mathrm{i},2}, (q_1, q_2), \mu_{\mathrm{o},1} \otimes \mu_{\mathrm{o},2}) \to_{f_1 \otimes f_2} (\mu'_{\mathrm{i},1} \otimes \mu'_{\mathrm{i},2}, (q'_1, q'_2), \mu'_{\mathrm{o},1} \otimes \mu'_{\mathrm{o},2})$ if and only if $(\mu_{\mathrm{i},i}, q_i \mu_{\mathrm{o},i}) \to_{f_i} (\mu'_{\mathrm{i},i}, q'_i, \mu'_{\mathrm{o},i})$ for $i \in \{1, 2\}$.*

**Definition 15 (Symmetries of $\mathbb{S}_r$).** *Given two pairs of natural numbers $m$ and $n$, we define the symmetry $C_{m,n} : m+n \to n+m$ as the morphism whose set of internal states is $\{*\}$ and such that $(\mu_\mathrm{i} \otimes \nu_\mathrm{i}, *, \nu_\mathrm{o} \otimes \mu_\mathrm{o}) \to_{C_{n,m}} (\mu'_\mathrm{i} \otimes \nu'_\mathrm{i}, *, \nu'_\mathrm{o} \otimes \mu'_\mathrm{o})$ if and only if for all $i \in [m_b] + [m_r]$, there exists $k$ such that $\mu'_\mathrm{i}(i) = \mu_\mathrm{i}(i) - k$ and $\mu'_\mathrm{o}(i) = \mu_\mathrm{o}(i) + k$, and similarly for for all $i \in [n_b] + [n_r]$, there exists $k$ such that $\nu'_\mathrm{i}(i) = \nu_\mathrm{i}(i) - k$ and $\nu'_\mathrm{o}(i) = \nu_\mathrm{o}(i) + k$.*

Finally, $\mathbb{S}_r$ also possesses a trace operator, as defined in

**Definition 16 (trace of $\mathbb{S}_r$).** *Given three pairs of natural numbers $m, n, p$ and a morphism $f \colon p + m \to p + n$, the* trace $\mathrm{Tr}^p_{m,n}(f)$ *of $f$ along $p$ is defined as having $Q$ as its set of internal states, and $(\mu_\mathrm{i}, q, \mu_\mathrm{o}) \to_{\mathrm{Tr}^p_{m,n}(f)} (\mu'_\mathrm{i}, q', \mu'_\mathrm{o})$ if and only if there exist $(\mu^0_\mathrm{i}, q^0, \mu^0_\mathrm{o}), \ldots, (\mu^n_\mathrm{i}, q^n, \mu^n_\mathrm{o})$ such that*

- *$\mu^0_\mathrm{i} = 0^p \otimes \mu_\mathrm{i}$, $q^0 = q$, and $\mu^0_\mathrm{o} = 0^p \otimes \mu_\mathrm{o}$,*
- *$\mu^n_\mathrm{i} = 0^p \otimes \mu'_\mathrm{i}$, $q^n = q'$, and $\mu^n_\mathrm{o} = 0^p \otimes \mu'_\mathrm{o}$, and*
- *for all $j < n$, $(\mu_\mathrm{i}^{\leftarrow,j}, q^j, \mu_\mathrm{o}^{\rightarrow,j}) \to_f (\mu^{j+1}_\mathrm{i}, q^{j+1}, \mu^{j+1}_\mathrm{o})$, where $\mu_\mathrm{o}^{\rightarrow,j} = 0^p \otimes \mu^j_{\mathrm{o}|>p}$, $\mu_\mathrm{i}^{\leftarrow,j} = (\mu^j_\mathrm{i} + \mu^j_\mathrm{o})_{|\leq p} \otimes \mu^j_{\mathrm{i}|>p}$, and $(\mu_i)_{i<n|P} = (\mu_i)_{i<n \wedge P(i)}$ is the restriction of $\mu$ to those indices that satisfy $P$.*

This definition is somewhat similar to that of trace on **Rel** [10], but taking the internal state into account. It must also take into account the fact that the tokens that have reached a looping exit must be made available in the corresponding entry, which is encoded into $\mu_{\mathrm{o}}^{\rightarrow,j}$ and $\mu_{\mathrm{i}}^{\leftarrow,j}$. They can be viewed as "reloading" the Petri net by moving all tokens that have reached a looping exit to the corresponding looping entry: $\mu_{\mathrm{o}}^{\rightarrow,j}$ has no tokens on the first $p$ exits and is equal to $\mu_{\mathrm{o}}^{j}$ on the other ones, while $\mu_{\mathrm{i}}^{\leftarrow,j}$ is equal to $\mu_{\mathrm{i}}^{j} + \mu_{\mathrm{o}}^{j}$ on the looping entries (gaining tokens from $\mu_{\mathrm{o}}^{j}$) and to $\mu_{\mathrm{i}}^{j}$ on the other ones.

**Lemma 4.** *$\mathbb{S}_r$ is a traced symmetric monoidal category.*

We need reflexivity and transitivity in Definition 12 to prove that $\mathrm{Tr}^{p}_{m,n}(-)$ is a trace operator.

### 4.3 The Semantic Functor $\mathcal{S}_r$

We now define the semantic function $\mathcal{S}_r \colon \mathbf{roPN} \to \mathbb{S}_r$ that interprets rightward open Petri nets in the semantic category.

**Definition 17 (the functor $\mathcal{S}_r$).** *We define the* semantic functor $\mathcal{S}_r \colon \mathbf{roPN} \to \mathbb{S}_r$ *on objects by $\mathcal{S}_r(n) = n$ and on arrows $N \colon m \to n$ as $\mathcal{S}_r(N) \colon m \to n$ whose set of internal states is $Q = \mathbb{N}^P$ the set of all markings of $N$, and $(\mu_{\mathrm{i}}, \mu, \mu_{\mathrm{o}}) \to_{\mathcal{S}_r(N)} (\mu'_{\mathrm{i}}, \mu', \mu'_{\mathrm{o}})$ if and only if $(\mu'_{\mathrm{i}}, \mu', \mu'_{\mathrm{o}})$ is reachable from $(\mu_{\mathrm{i}}, \mu, \mu_{\mathrm{o}})$ in $N$ by a sequence of firings and slidings.*

Our goal is to prove that $\mathcal{S}$ is a traced symmetric monoidal functor. The difficult points to prove are $\mathcal{S}_r(N; N') = \mathcal{S}_r(N); \mathcal{S}_r(N')$ and $\mathcal{S}_r(\mathrm{Tr}^{p}_{m,n}(N)) = \mathrm{Tr}^{p}_{m,n}(\mathcal{S}_r(N))$. We only give informal proofs of these two identities.

We start by proving $\mathcal{S}_r(N; N') = \mathcal{S}_r(N); \mathcal{S}_r(N')$, which amounts to proving $\mathcal{S}_r(N; N')(c) = (\mathcal{S}_r(N); \mathcal{S}_r(N'))(c)$ for all $c = (\mu_{\mathrm{i}}, \mu, \mu_{\mathrm{o}})$. First, given two sequences $(\mu_{\mathrm{i}}, \mu_N, 0^n) \to \ldots \to (\mu'_{\mathrm{i}}, \mu'_N, \mu^*)$ in $N$ and $(\mu^*, \mu_M, \mu_{\mathrm{o}}) \to \ldots \to (0^n, \mu'_M, \mu'_{\mathrm{o}})$, it is simple to build a sequence $c \to \ldots \to (\mu'_{\mathrm{i}}, \mu', \mu'_{\mathrm{o}})$ in $N; N'$ by concatenating the two sequences and turning pairs of transitions/slidings that touch the middle interface to a single transition/sliding in $N; N'$.

It is slightly more difficult to reconstruct $\mu^*$ and the sequences on $N$ and $N'$ from the sequence on $N; N'$, for which the following lemma is convenient.

**Lemma 5 (Priority).** *Given Petri nets $N \colon m \to n$ and $N' : n \to p$, a configuration $c$ for $N; N'$, $k \in T \cup (l_i)_{i \in [m_b]+}$, $k' \in T' \cup (r_j)_{j \in [p_r]}$ if $k'$ is enabled on $c$ and $k$ is enabled on $k'(c)$, then $k$ is enabled on $c$, $k'$ on $k(c)$, and $k'(k(c)) = k(k'(c))$.*

This lemma is crucial, as given a sequence of transitions and slidings in $N; N'$, it guarantees that all the sliding and transitions in $N$ can be prioritised over the ones in $N'$. This means we can always assume that all transitions and sliding of $N$ are fired before those of $N'$. Given a sequence of transitions and slidings on $N; N'$, we can permute all entry slidings and transitions in $N$ first. We can then rebuild $\mu^*$ by counting how many tokens are output by the input slidings

and transitions in $N$, then reconstruct the sequence on $N$ (resp. $N'$) by copying those slidings in the sequence on $N;N'$ that are in $N$ (resp. $N'$). This proves that $\mathcal{S}_r(N;N') = \mathcal{S}_r(N);\mathcal{S}_r(N')$.

Finally, we move on to proving $\mathcal{S}_r(\mathrm{Tr}^p_{m,n}(N)) = \mathrm{Tr}^p_{m,n}(\mathcal{S}_r(N))$. Here again, the idea is to turn sequences $c \to \ldots \to c'$ in $\mathrm{Tr}^p_{m,n}(N)$ to sequences in $N$ for the (semantic) trace and vice-versa. The only difficulty is managing tokens on the looping part of the interface. Given a sequence $c \to \ldots \to c'$ in $\mathrm{Tr}^p_{m,n}(N)$ whose general term is $(\mu_{\mathrm{i}}^j, \mu^j, \mu_{\mathrm{o}}^j)$, we can recreate a sequence for the trace by copying all transitions/slidings and "reloading" the net (see Definition 16) after each move $\max\{\phi_p(i) \mid i \in [p]\}$ times, so that reloading moves no tokens anymore. Its general term is $(\tilde{\mu}_{\mathrm{i}}^j \otimes \mu_{\mathrm{i}}^j, \mu^j, \tilde{\mu}_{\mathrm{o}}^j \otimes \mu_{\mathrm{o}}^j)$, and is a witness that $c \to \ldots \to c'$ in the trace. The other direction is simpler. Given a sequence of configurations such that $(\mu_{\mathrm{i}}^{\to,j}, \mu^j, \mu_{\mathrm{o}}^{\to,j}) \to (\mu_{\mathrm{i}}^{j+1}, \mu^{j+1}, \mu_{\mathrm{o}}^{j+1})$ in $N$, we can create a sequence for $\mathrm{Tr}^p_{m,n}(N)$ by simply copying all firings/slidings.

## 5 The Semantic Interpretation Functor $[\![.]\!]$

### 5.1 Moving to the Bidirectional Case

One idea of this line of work [21,20,22] is that the bidirectional framework can be derived from the unidirectional one automatically using the Int-construction, instead of being an ad hoc construction. Here, we define the category **oPN** of open Petri nets from that of rightward open Petri nets. This simplifies the development, readability, and conciseness of the ideas, in addition to giving a compact closed structure "for free" to **oPN** (and the semantic domain $\mathbb{S}$).

The basic idea of the Int-construction [10] is that objects of $\mathrm{Int}(\mathbb{C})$ are pairs $(X, Y)$ of objects of $\mathbb{C}$ where $X$ represents objects going "forward", while $Y$ represents objects going "backward". Morphisms from $(X, Y)$ to $(X', Y')$ in $\mathrm{Int}(\mathbb{C})$ are morphisms $X \otimes Y' \to Y \otimes X'$ (notice that the $Y$'s are reversed). In our case, this allows us to model open ends going leftward, allowing for simpler modelling especially for looping structures, which do not need to use trace explicitly.

**Definition 18.** *We define* $\mathbf{oPN} = \mathrm{Int}(\mathbf{roPN})$ *and* $\mathbb{S} := \mathrm{Int}(\mathbb{S}_r)$ *the category of open Petri nets and semantic category, respectively. Moreover, because* $\mathcal{S}_r$ *respects the traced symmetric monoidal structure, the* Int*-construction applies to it too, and we define* $\mathcal{S} = \mathrm{Int}(\mathcal{S}_r) \colon \mathbf{oPN} \to \mathbb{S}$.

From the property of the Int-construction, it follows that :

**Lemma 6.** **oPN** *and* $\mathbb{S}$ *are compact closed categories and* $\mathcal{S}$ *is a compact closed functor.*

This lemma gives a formal meaning to the claim we have been making that the bidirectional framework is derived canonically from the unidirectional one, and gives meaning to Figure 1.

### 5.2 The Coloured Graphical PROP

Until now, we have considered **oPN** (or **roPN**) to be our syntax and $\mathbb{S}$ to be our semantics. However, the definitions of the different elements of **oPN** such as composition and trace are hard to read and to work with, and they obscure their graphical essences. In this section, we give a simpler syntax for open Petri nets based on free coloured PROPs.

PROPs are a categorical concept widely used in many works to describe various graphical structures and fit well to define a categorical syntax in the form of string diagrams. Precisely, a PROP is a symmetric monoidal category whose objects are the elements of the free monoid on a single object. Concretely, the elements are built as the powers of a single base object; hence, every object admits a unique decomposition. In particular, this allows us to draw string diagrams where wires do not carry specific information. This formalism is expanded upon by considering the free monoid on a finite set $C$ (called the set of *colours*) in an object called a coloured PROP [7]. Here, the usage of coloured PROPs is doubly beneficial: because of the nature of Petri nets, we need to consider two colours (as explained before) for the arrows, and we have to double the number of colours (going from 2 to 4) to encompass the bidirectional setting. We define our syntax category as a free PROP on a set of generators and equations. A similar approach is presented in [20,21,22], from which we will adapt the results without fully detailing them. Let $C = \{\bullet_L, \bullet_R, \bullet_L, \bullet_R\}$ be the set of colours. The colour $\bullet_L$ (resp. $\bullet_R$) correspond to red arrows going leftward (resp. red arrows going rightward), and analogously for $\bullet_L$ and $\bullet_L$. We now move on to defining the signature $\Sigma_{PN}^C$, containing the generators. They are illustrated in Figure 12.

**Definition 19** ($\Sigma_{PN}^C$). *We pose* $\Sigma_{PN}^C : C^* \times C^* \to \mathbf{Set}$ *the functor defined by* $\Sigma_{PN}^C(\epsilon, \bullet_R\bullet_L) = \{\eta_b, t_{0,2}\}$, $\Sigma_{PN}^C(\bullet_L\bullet_R, \epsilon) = \{\epsilon_b, p_{2,0}\}$, $\Sigma_{PN}^C(\epsilon, \bullet_R\bullet_L) = \{\eta_r, p_{0,2}\}$, $\Sigma_{PN}^C(\bullet_L\bullet_R, \epsilon) = \{\epsilon_r, t_{2,0}\}$, $\Sigma_{PN}^C(u, v) = \{p_{|u|,|v|}\}$ *if* $u \in \bullet_L^*$ *and* $v \in \bullet_L^*$, *and* $\Sigma_{PN}^C(u, v) = \{t_{|u|,|v|}\}$ *if* $u \in \bullet_L^*$ *and* $v \in \bullet_L^*$.

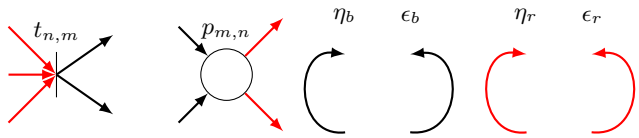


**Fig. 12.** The generators in $\Sigma_{PN}^C$

We define $E_{PN}^C$ to be the set containing the black and red snake equations for $\epsilon_b$, $\epsilon_r$, $\eta_b$, $\eta_r$. Our syntactic PROP is $\mathbb{F}(\Sigma_{PN}^C, E_{PN}^C)$, where $\mathbb{F}$ is defined as the free construction in the category **PROP** detailed in [7]. This freeness allows us to define the realisation functor $\mathcal{R} \colon \mathbb{F}(\Sigma_{PN}^C, E_{PN}^C) \to \mathbf{oPN}$ canonically, by considering the valuation $V \colon \Sigma_{PN}^C \to \mathrm{Int}(\mathbf{roPN})$ defined by $V(\epsilon_b) = V(\eta_b) =$

$\mathrm{id}_{(1,0)}$, $V(\epsilon_r) = V(\eta_r) = \mathrm{id}_{(0,1)}$, $V(p_{m,n})$ the open Petri net $m \to n$ with a single place with $n$ inputs and $m$ outputs and no transitions, and $V(t_{m,n})$ the open Petri net $m \to n$ with no places and a single transition with $n$ inputs and $m$ outputs.

Not only is $\mathbb{F}(\Sigma_{PN}^C, E_{PN}^C)$ the free PROP over $\Sigma$ by definition, it also possesses a more interesting property here.

**Lemma 7.** *There is a signature $\Sigma$ s.t $\mathbb{F}(\Sigma_{PN}^C, E_{PN}^C) \cong \mathrm{Int}(\mathbb{F}_{tr}(\Sigma))$*

The nature of the signature $\Sigma$ is not useful on its own for this work but is developed in [20]. This theorem allows us to consider the second universal property of $\mathbb{F}(\Sigma_{PN}^C, E_{PN}^C)$, as it is also a free compact closed category. First, we take the valuation $V$ defined earlier and name $\mathcal{R}$ the unique compact closed functor that makes the leftmost triangle commutes. Second, we take the valuation $\mathcal{S} \circ V : \Sigma_{PN}^C \to \mathbb{S}$, and and we name $[\![.]\!] : \mathbb{F}(\Sigma_{PN}^C, E_{PN}^C) \to \mathbb{S}$ the unique functor that makes the larger triangle in Figure 13 commute.

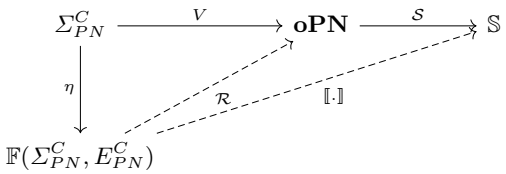


**Fig. 13.** The construction of $\mathcal{R}$ and $[\![.]\!]$

This means that both $\mathcal{S} \circ \mathcal{R}$ and $[\![.]\!]$ satisfies the lifting property of $\mathbb{F}(\Sigma_{PN}^C, E_{PN}^C)$ as the free compact closed category, hence the following result.

**Theorem 2.** $[\![.]\!] = \mathcal{S} \circ \mathcal{R}$.

We could have defined $[\![.]\!]$ as $\mathcal{S} \circ \mathcal{R}$, but this would have hidden the fact that $[\![.]\!]$ is given by a freeness property and can thus be computed inductively. This property is important to solve the open reachability problem inductively.

## 6 Conclusion and Future Work

In this work, we have designed a compositional approach to Petri nets to solve open reachability. To achieve such a result, we have defined a category of open Petri nets, as well as a graphical language of string diagram and a semantic domain in which open Petri nets can be interpreted.

Directions for future work include the bounded case, in which there is a finite limit to the number of tokens present on places/open ends. However, semantic composition is more complex in this case. Moreover, we could implement an algorithm in the bounded case. Finally, a general work on compositionality, following [20,21] and the current work, would prove useful to design frameworks for open structures without starting from scratch every time.

## References


1. Baez, J.C., Coya, B., Rebro, F.: Props in network theory. Theory and Applications of Categories **33**(25), 727–783 (2018)
2. Baez, J.C., Maste, J.: Open Petri nets. Mathematical Structures in Computer Science **30**(3), 314–341 (2020)
3. Baez, J.C., Pollard, B.S.: A compositional framework for reaction networks. Reviews in Mathematical Physics **29**(09), 1750028 (2017)
4. Bonchi, F., Holland, J., Piedeleu, R., Sobociński, P., Zanasi, F.: Diagrammatic algebra: from linear to concurrent systems. Proceedings of the ACM on Programming Languages **3**(POPL), 1–28 (2019)
5. Bonchi, F., Sobociński, P., Zanasi, F.: Interacting Hopf algebras. Journal of Pure and Applied Algebra **221**(1), 144–184 (2017)
6. Carette, T., Horsman, D., Perdrix, S.: Szx-calculus: Scalable graphical quantum reasoning. In: 44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany. vol. 138, pp. 55:1–55:15 (2019)
7. Carette, T., Perdrix, S.: Colored props for large scale graphical reasoning (2020)
8. Fiore, M., Campos, M.D.: The algebra of directed acyclic graphs. In: Computation, Logic, Games, and Quantum Foundations. The Many Facets of Samson Abramsky, pp. 37–51. Springer (2013)
9. Joyal, A., Street, R.: Braided tensor categories. Advances in Mathematics **102**(1), 20–78 (1993)
10. Joyal, A., Street, R., Verity, D.: Traced monoidal categories. In: Mathematical proceedings of the cambridge philosophical society. vol. 119, pp. 447–468. Cambridge University Press (1996)
11. Mac Lane, S.: Categories for the working mathematician, vol. 5. Springer Science & Business Media (2013)
12. Penrose, R.: Applications of negative dimensional tensors. Combinatorial mathematics and its applications **1**, 221–244 (1971)
13. Peterson, J.L.: Petri nets. ACM Computing Surveys (CSUR) **9**(3), 223–252 (1977)
14. Rathke, J., Sobociński, P., Stephens, O.: Compositional reachability in petri nets. In: Reachability Problems: 8th International Workshop, RP 2014, Oxford, UK, September 22-24, 2014. Proceedings 8. pp. 230–243. Springer (2014)
15. Reisig, W.: Petri nets: an introduction, vol. 4. Springer Science & Business Media (2012)
16. Reisig, W.: Understanding Petri nets. Springer (2016)
17. Selinger, P.: A survey of graphical languages for monoidal categories. In: New structures for physics, pp. 289–355. Springer (2010)
18. Sobociński, P.: Compositional model checking of concurrent systems, with petri nets. arXiv preprint arXiv:1603.00976 (2016)
19. Stephens, O.: Compositional specification and reachability checking of net systems. Ph.D. thesis, University of Southampton (2015)
20. Watanabe, K., Eberhart, C., Asada, K., Hasuo, I.: A compositional approach to parity games. arXiv preprint arXiv:2112.14058 (2021)
21. Watanabe, K., Eberhart, C., Asada, K., Hasuo, I.: Compositional probabilistic model checking with string diagrams of mdps. In: 35th International Conference on Computer Aided Verification (CAV 2023) (2023)
22. Watanabe, K., Eberhart, C., Asada, K., Hasuo, I.: Compositional solution of mean payoff games by string diagrams. arXiv preprint arXiv:2307.08034 (2023)