Categorical Models of Concurrency

Candidate #1080882

Contents

1. Introduction

The dining philosophers problem is often used to introduce students to the concepts of concurrency. The problem was first given by Edsger Djisktra as a student exam exercise [Dij65] and was later reformulated to its modern day form by Tony Hoare [Hoa85]. He presents it in the following way:

> In ancient times, a wealthy philanthropist endowed a College to accommodate five eminent philosophers. Each philosopher had a room in which he could engage in his professional activity of thinking; there was also a common dining room, furnished with a circular table, surrounded by five chairs, each labelled by the name of the philosopher who was to sit in it. The names of the philosophers were PHIL0 , PHIL1, PHIL2 , PHIL3 , PHIL4, and they were disposed in this order anticlockwise around the table. To the left of each philosopher there was laid a golden fork, and in the centre stood a large bowl of spaghetti, which was constantly replenished. A philosopher was expected to spend most of his time thinking; but when he felt hungry, he went to the dining room, sat down in his own chair, picked up his own fork on his left, and plunged it into the spaghetti. But such is the tangled nature of spaghetti that a second fork is required to

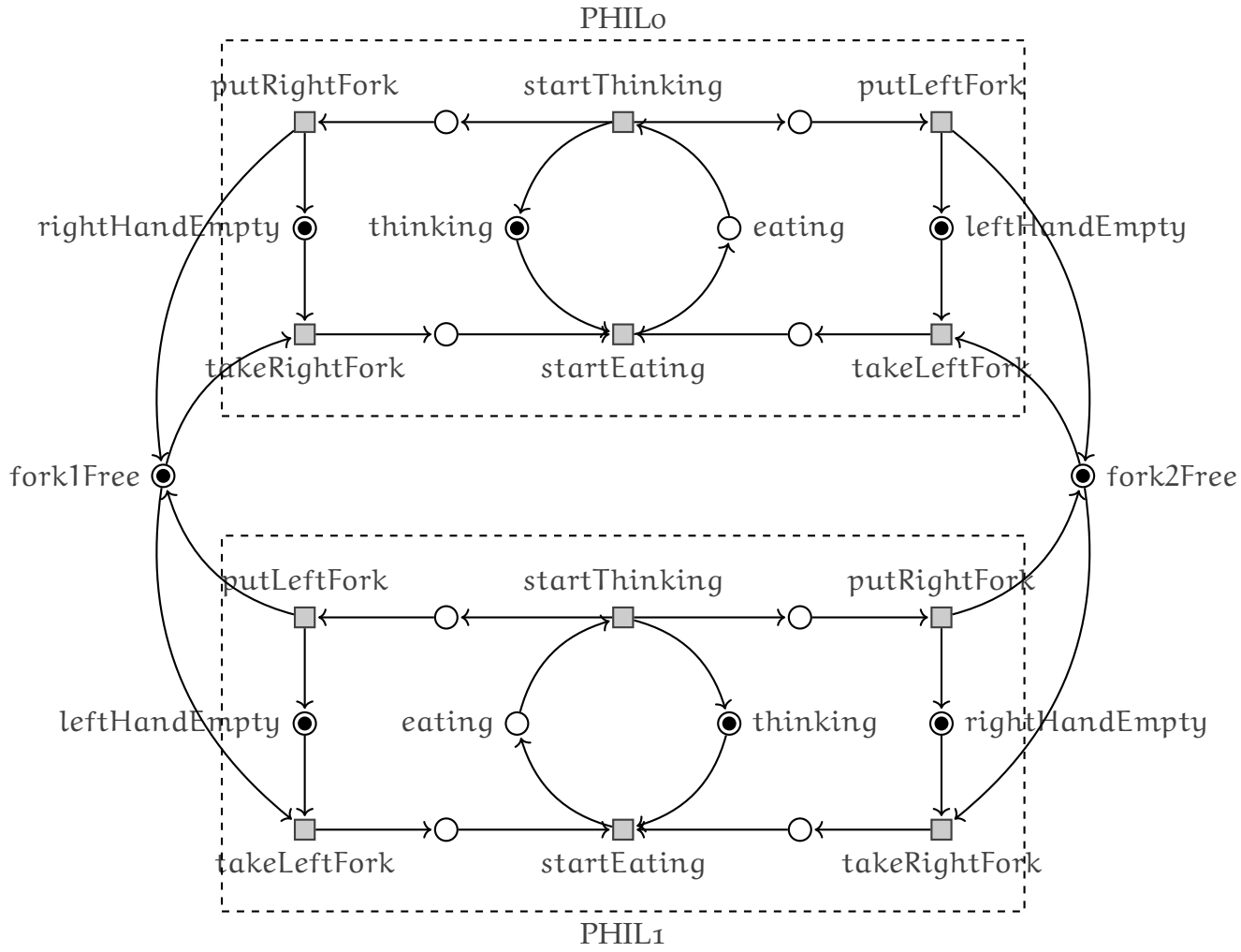


FIGURE 1. A Petri net model for the dining philosophers problem in the special case of only two philosophers. The tokens have been placed in the starting position where both philosophers are currently in the thinking state and have both hands free.

> carry it to the mouth. The philosopher therefore had also to pick up the fork on his right. When he was finished he would put down both his forks, get up from his chair, and continue thinking. Of course, a fork can be used by only one philosopher at a time. If the other philosopher wants it, he just has to wait until the fork is available again.

In Fig. 1 we have pictured this scenario for two philosophers—the situation for five philosophers is analogous, but there is little to gain in terms of intuition by drawing the full Petri net—as a *Petri net* with the specific starting state of both philosophers having their hands free and thinking. One could imagine a scenario where each philosopher has sat down to eat, picked up their right fork, and realized the fork to their left is currently in use by another philosopher. The process has ended up in a *deadlock*. The corresponding Petri net is the one depicted in Fig. 2 where we can see that neither philosopher has enough tokens to move into the eating state.

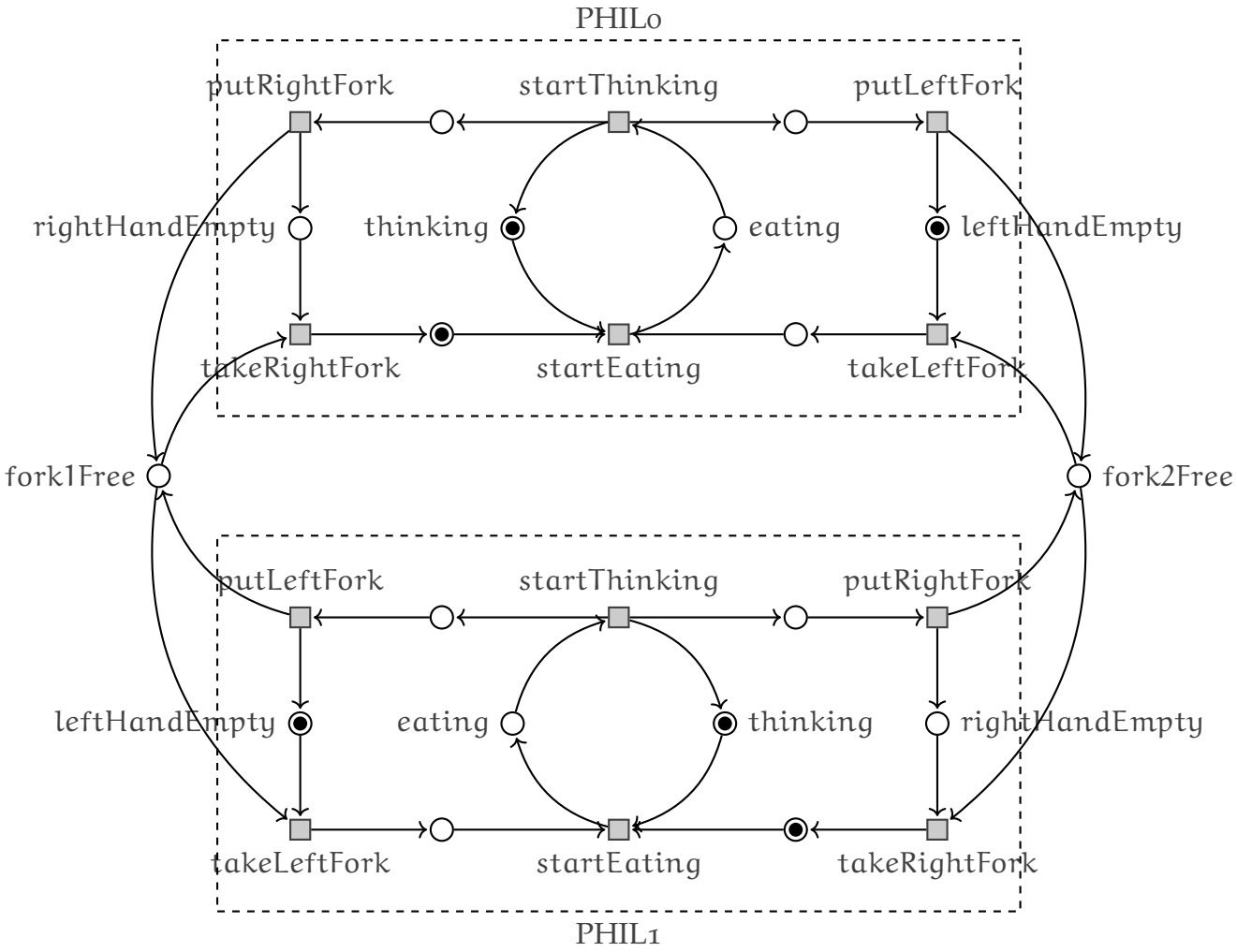


FIGURE 2. The two philosophers have now transitioned to a deadlock state where neither are able to eat unless the other decides to put down his or her fork. In the case of two greedy philosophers this would cause both philosophers to die of starvation.

The goal is to design a protocol which ensures no philosopher dies of starvation. One approach could be to try to extend the Petri net in Figure 1 with more places and transitions and then try to reason about this new system. However, a problem with this traditional approach to studying Petri nets is that it is monolithic, in the sense that one studies Petri nets as closed systems. This is contrary to the ethos of category theory where the central theme is compositionality. Taking a categorical approach would allow us to develop small "Lego pieces" which we could piece together into more complex systems.

1.1. Our contribution. The main new contribution of this text is the construction of the *conductor* in the framework of resource calculi. More specifically, in Section 4.3 we introduce a class of components in Rc_{sa} with zero input wires that has the ability to synchronize the otherwise asynchronous Petri places. Proposition 4.7 gives their semantics in the stateful extension of polyhedral relations, $\mathsf{St}(\mathsf{PolyRel})$

2. Petri nets

We have already seen how Petri nets can be used to model concurrent processes. They were introduced by Carl Adam Petri in his PhD dissertation [Pet62] as a graphical formalism for studying distributed systems. There are several equivalent ways of defining them, but we will use the formalism presented in [Bru+13].

Definition 2.1 (Petri net). *A **Petri net**, is a tuple* $\mathcal{P} = (\mathsf{P}, \mathsf{T}, {}^\circ{-}, {-}^\circ)$ *consisting of*

- *a finite set of places* P,
- *a finite set of transitions* T,
- *functions* ${}^\circ{-}, {-}^\circ : \mathsf{T} \to \mathbb{N}^{\mathsf{P}}$.

In the Petri net of the dining philosophers we also had a way of indicating the current state via tokens. The formal way to do this is via *markings*.

Definition 2.2 (Marked Petri net). *A **marked Petri net** is a tuple* $\mathcal{M} = (\mathcal{P}, \mathfrak{m}_0)$ *consisting of*

- *a Petri net* $\mathcal{P} = (\mathsf{P}, \mathsf{T}, {}^\circ{-}, {-}^\circ)$,
- *a function* $\mathfrak{m}_0 : \mathsf{P} \to \mathbb{N}$ *called the **initial marking** of* $\mathcal{P}$.

The initial marking of a marked Petri net specifies the starting state of the system. In the dining philosophers problem, the initial marking is the one where all philosophers are thinking and all forks are free. That is to say, we place one token on each of these places, as in Fig. 1.

Now, the way we think about a Petri net is in terms of what states can be reached from the current one if we expend the necessary tokens. The formal way of defining this is through a *firing*.

Definition 2.3 (Firing). *Let* $\mathcal{P} = (\mathsf{P}, \mathsf{T}, {}^\circ{-}, {-}^\circ)$ *be a Petri net. A **firing** of* $\mathcal{P}$ *is a tuple* $(\mathfrak{m}, \mathfrak{m}')$ *where* $\mathfrak{m}, \mathfrak{m}' \in \mathbb{N}^{\mathsf{P}}$ *are markings such that there exist* $\mathsf{t} \in \mathsf{T}$ *with* ${}^\circ\mathsf{t} \leqslant \mathfrak{m}$ *and* $\mathfrak{m}' = \mathfrak{m} - {}^\circ\mathsf{t} + \mathsf{t}^\circ$. *We prefer to write a firing* $(\mathfrak{m}, \mathfrak{m}')$ *as* $\mathfrak{m} \to \mathfrak{m}'$ *and define the set of firings to be*

$$\mathsf{Fire}(\mathcal{P}) := \{(\mathfrak{m}, \mathfrak{m}') \mid \mathfrak{m} \to \mathfrak{m}'\}. \tag{1}$$

The set of firings, $\mathsf{Fire}(\mathcal{P})$, gives us the *firing semantics* of a Petri net $\mathcal{P}$. This is the desired operational semantics which instructs us to understand Petri nets in terms of their firing relations. An important point, which will be made clear later, is that $\mathsf{Fire}(\mathcal{P})$ is an *additive relation*.

3. Resource calculi

The starting point for finding a categorical model which extends the model of Petri nets we explored in the previous section, will be resource calculi. In the following we shall assume familiarity with the technicalities of string diagrams for symmetric monoidal categories (SMCs), a great resource for those not familiar is [Sel10]. We will use the standard convention of representing the identity, symmetry, and identity on the monoidal unit by —, $\times$, and $\square$, respectively.

3.1. Syntax.

We will define different resource calculi in terms of signature and hence we take a small detour to discuss some aspects of this construction.

Definition 3.1 (Signature). *The category of **signatures** is the functor category* $\mathsf{Set}^{\mathbb{N}\times\mathbb{N}}$ *where objects are functors and morphisms are natural transformations.*

The following result allows us to speak about freely generated props on a signature.

Proposition 3.2 ([Pie18]). *There is a monadic forgetful functor* $U : \mathsf{Prop} \to \mathsf{Set}^{\mathbb{N}\times\mathbb{N}}$ *sending a prop to its underlying signature. This means that* U *has a left adjoint* $P : \mathsf{Set}^{\mathbb{N}\times\mathbb{N}} \to \mathsf{Prop}$ *and that the category of algebras for the monad* UP *is equivalent to the category of props.*

Proof. As Piedeleu [Pie18] mentions, this is a generalization of Lawvere's original result [Law63] and the reader is referred to [Pie18, Proposition 29] for the details. □

Another common way to describe a prop is through a presentation, i.e., a set of generators and equations between terms built out of the generators. More formally, we have the following result.

Proposition 3.3 ([Pie18]). *Any prop is the coequalizer in* Prop *of two parallel prop morphisms of the form*

$$PE \underset{\rho}{\overset{\lambda}{\rightrightarrows}} P\Sigma$$

for two signatures E *and* Σ.

Proof. Let T be a prop and set $\epsilon : PU \to 1$ to be the counit in the adjunction of Proposition 3.2. Set $\Sigma = UT$, $E = UPUT$, $\lambda = PU\epsilon_T$, and $\rho = \epsilon_{PUT}$. We then have the map $\epsilon_T : P\Sigma \to T$ which one can check satisfies the universal property. □

Definition 3.4 ([Pie18]). *A presentation of a prop* T *is the data* $(E, \Sigma, \lambda : PE \to P\Sigma, \rho : PE \to P\Sigma)$ *such that* T *is the coequalizer of the corresponding diagram.*

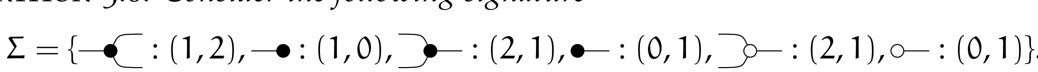
It is often the case that we start with some prop T which has a presentation $(E, \Sigma, \rho, \lambda)$ and want to add some equations between pairs of morphisms in T. That is to say, we have a functor $X : \mathbb{N} \times \mathbb{N} \to \mathsf{Set}$ and a pair of morphisms $l, r : X \to UP\Sigma$. We want to create a new prop, which by Proposition 3.3 is the same as specifying a new presentation.

Definition 3.5. *Let* $(E, \Sigma, \lambda, \rho)$ *be a presentation of a prop* T *and suppose* $X : \mathbb{N} \times \mathbb{N} \to \mathsf{Set}$ *is a set of equations with corresponding maps* $l, r : X \to UP\Sigma$. *The prop* $T_{/X}$ *is defined to be the coequalizer of the following diagram*

$$PE + PX \underset{\rho + Pl;\epsilon_{P\Sigma}}{\overset{\lambda + Pl;\epsilon_{P\Sigma}}{\rightrightarrows}} P\Sigma.$$

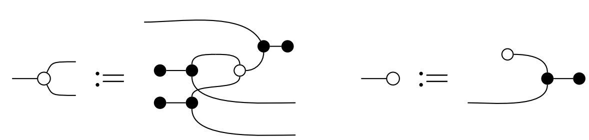
We are now in a position to formally define Rc through a presentation.

Definition 3.6. *Consider the following signature*



(2)

Define the ***prop of circuits*** *to be* $\mathsf{Circ} := P\Sigma$. *Then, using the syntactic sugar*



(3)

we define the ***resource calculus***, Rc, *to be* Circ *modulo the equations presented in Fig. 3.*

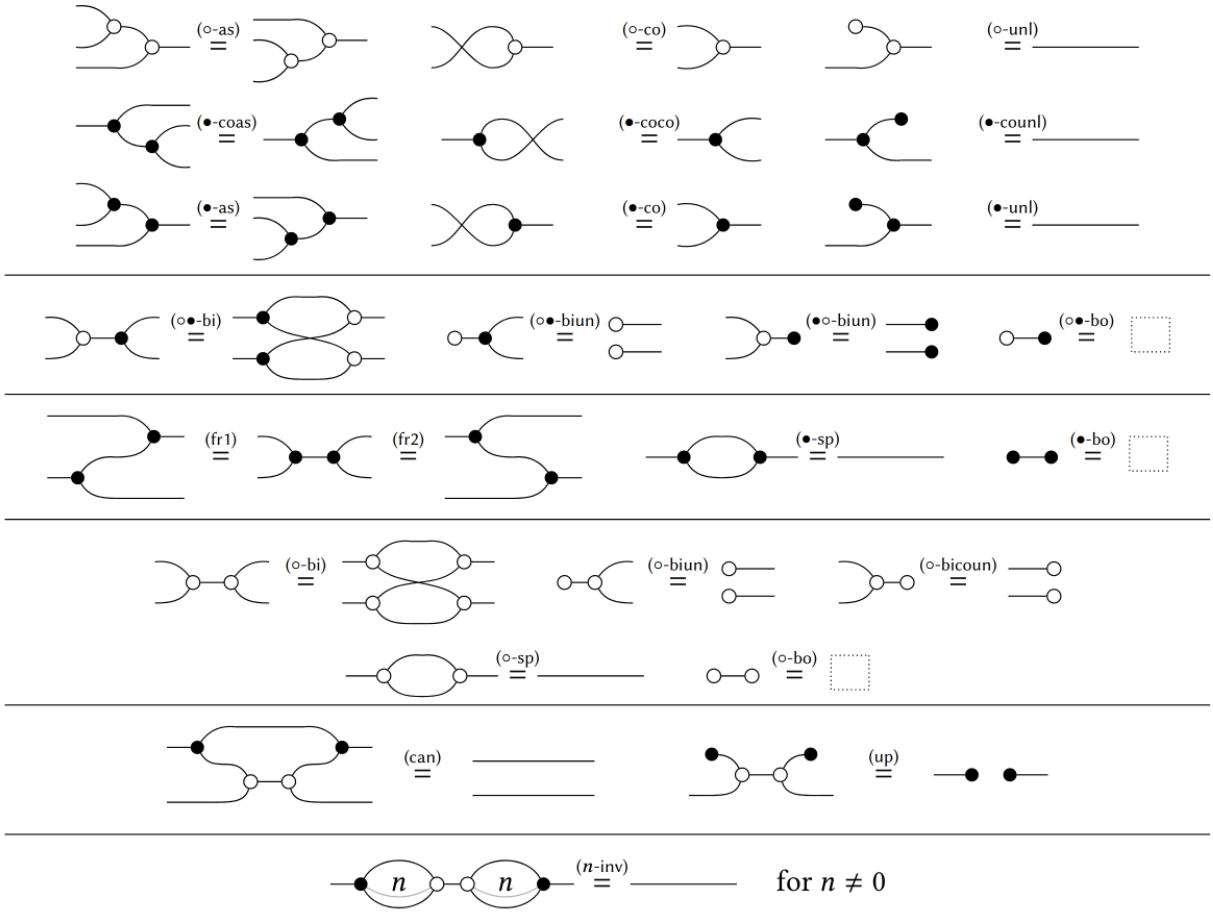


FIGURE 3. Presentation of Rc due to [Bon+19a].

3.2. Semantics. The next goal is to define a category of semantics and find a prop isomorphism between the syntax category Rc and a certain operational semantics category.

DEFINITION 3.7. *Let S be a semiring. Define* Rel_S *to be the prop whose arrows* $n \to m$ *are relations* $R \subset S^n \times S^m$. *Moreover, given* $R : n \to m$ *and* $R' : m \to l$ *we let*

$$R ; R' := \{(x, z) \mid \exists (x, y) \in R \wedge (y, z) \in R'\}. \tag{4}$$

The monoidal product of two relations $R : n_1 \to m_1$ *and* $R' : n_2 \to m_2$ *is gotten by taking the Cartesian product of the two relations, i.e.,*

$$R_1 \oplus R_2 := \left\{ \left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \right) \mid (x_1, y_1) \in R_1 \wedge (x_2, y_2) \in R_2 \right\}. \tag{5}$$

Identities and symmetry are defined in the obvious ways.

An important special case for the resource calculus is that of $S = \mathbb{N}$ and its sub-prop of additive relations.

DEFINITION 3.8 ([Bon+19b]). *An* ***additive relation*** *of type* $k \to l$ *is a subset* $R \subset \mathbb{N}^k \times \mathbb{N}^l$ *such that*

(i) $(0, 0) \in R$ *and,*

(ii) if $(\mathbf{a},\mathbf{b}),(\mathbf{a}',\mathbf{b}') \in R$ *then* $(\mathbf{a}+\mathbf{a}',\mathbf{b}+\mathbf{b}') \in R$.

Define AddRel *to be the sub-prop of* $\mathsf{Rel}_{\mathbb{N}}$ *where the maps of type* $k \to l$ *are finitely generated additive relations instead of general relations on* $\mathbb{N}$.

Inherent in the above definition is the claim that AddRel is a prop. The proof of this can be found in [Pie18].

We can now define the prop morphism which gives us our semantics.

Definition 3.9. *Let* $[\![-]\!] : \mathsf{Rc} \to \mathsf{AddRel}$ *be the unique map which on the generators of* Rc *is given by:*

$$[\![\multimap]\!] = \left\{ \left(\binom{n}{m}, n+m \right) \mid (n,m) \in \mathbb{N}^2 \right\} \qquad [\![\circ\!-]\!] = \{(\epsilon, 0)\}$$

$$[\![\bullet\!\multimap]\!] = \left\{ \left(\binom{n}{n}, n \right) \mid n \in \mathbb{N} \right\} \qquad [\![-\!\bullet]\!] = \{(n,\epsilon) \mid n \in \mathbb{N}\}$$

$$[\![\multimap\!\bullet]\!] = \left\{ \left(n, \binom{n}{n} \right) \mid n \in \mathbb{N} \right\} \qquad [\![\bullet\!-]\!] = \{(\epsilon, n) \mid n \in \mathbb{N}\}$$

where ϵ *is the unique element of* $\mathbb{N}^0$.

It turns out that $[\![-]\!]$ is an isomorphism of props. For the details of the proof we refer to [Bon+19a, Section 3].

Theorem 3.10. *The semantics* $[\![-]\!] : \mathsf{Rc} \to \mathsf{AddRel}$ *is a prop isomorphism.*

3.3. An affine extension. The prop isomorphism $[\![-]\!] : \mathsf{Rc} \to \mathsf{AddRel}$ shows us how the resource calculus Rc completely describes additive relations. However, when we want to design concurrent systems, it is often a good idea to make use of mutual exclusion which we can think of as a gate $\multimap\!+$ which has semantics

$$[\![\multimap\!+]\!] = \left\{ \left(\binom{1}{0}, 1 \right) \left(\binom{0}{1}, 1 \right), \left(\binom{0}{0}, 0 \right) \right\}. \tag{6}$$

The problem with this is that the above relation is not additive. Hence we need another category of operational semantics to encode this behaviour. Piedeleu [Pie18] identified this category as the category of polyhedral relations and we will give a description of it here.

Definition 3.11 (Discrete polyhedron). *A* ***discrete polyhedron*** *is a set* $Q \subset \mathbb{N}^d$ *for which there exist finite* $B, D \subset \mathbb{N}^d$ *such that* $Q = P(B,D)$ *where*

$$P(B,D) = \bigcup_{\mathbf{b} \in B} \{\mathbf{b} + \langle D \rangle\} \tag{7}$$

and $\langle D \rangle$ *is the set of finite linear combinations of elements of* D.

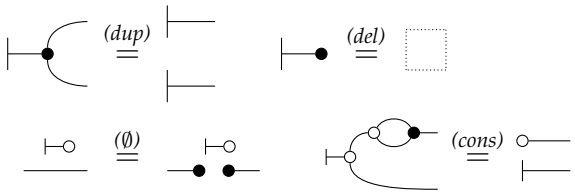
The intuition behind $P(B,D)$ is that it is the polyhedron generated by B and D. The elements of B are **base points** and elements of D are the **directions**.

Definition 3.12 (Polyhedral relation). *A* ***polyhedral relation*** *of type* $k \to l$ *is a discrete polyhedron* $R \subset \mathbb{N}^k \times \mathbb{N}^l$.

Piedeleu [Pie18] showed that the composition of two polyhedral relations is again a polyhedral relation. Hence we can talk about the sub-prop $\mathsf{PolyRel} \subset \mathsf{Rel}_{\mathbb{N}}$ of polyhedral relations.

The last thing we need to do is to specify our syntax.

Definition 3.13. *Let* Rc_a *be the prop freely generated over the same signature as* Rc *with the additional generator* $\vdash : (0,1)$ *and the following equations*



It turns out that Rc_a gives a complete and sound calculus for PolyRel.

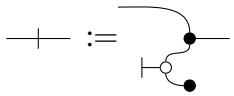
Theorem 3.14 ([Pie18]). *Let* $[\![-]\!]_a : \mathsf{Rc}_a \to \mathsf{PolyRel}$ *be the unique extension of* $[\![-]\!] : \mathsf{Rc} \to \mathsf{AddRel}$ *where*

$$[\![\vdash]\!]_a := \{(\epsilon, 1)\}. \tag{8}$$

Then $[\![-]\!]_a : \mathsf{Rc}_a \to \mathsf{PolyRel}$ *is an isomorphism of props.*

Proof. See [Pie18, Theorem 124]. □

Going back to the motivation for looking at Rc_a and PolyRel we now have a way to build our mutual exclusion gate. To do this, we first start by defining



(9)

which then has semantics

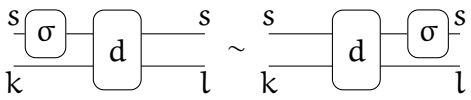
$$[\![-\!\!+\!\!-]\!]_a = \{(0,0),(1,1)\}. \tag{10}$$

Composing  with $-\!\!+\!\!-$ we then get our desired mutual exclusion gate. The point of this is that Rc_a is an expressive resource calculus which allows us to model stateless concurrent processes. However, to get state, we need to extend it.

3.4. A stateful extension.

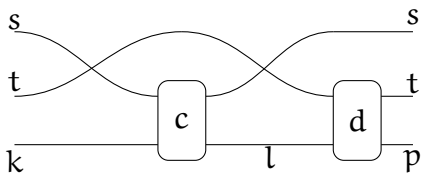
Definition 3.15 ([Pie18]). *Let* T *be a prop. Define* St(T) *as the prop where:*

- *morphisms of type* $k \to l$ *are pairs* (s, c) *where* $s \in \mathbb{N}$ *and* $c : s + k \to s + l$ *is a morphism of* T*, quotiented by the smallest equivalence relation which includes*



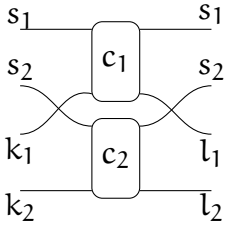
(11)

  for all permutations $\sigma : s \to s$,
- *the composition of* $(s, c) : k \to l$ *and* $(t, d) : l \to p$ *is* $(s + t, e)$ *where* e *is the arrow of* T *given by*



(12)

- *the monoidal product of* $(s_1, c_1) : k_1 \to l_1$ *and* $(s_2, c_2) : k_2 \to l_2$ *is* $(s_1 + s_2, e)$ *where* e *is given by*



(13)

- *the identity on* k *is* $(0, 1_j)$ *and the symmetry of* k, l *is* $(0, \sigma_{k,l})$.

Skipping some straightforward, but tedious, details this extends to an endofunctor $\mathsf{St} : \mathsf{Prop} \to \mathsf{Prop}$.

Now, define X to be the prop freely generated on the generator $-\boxed{x}- : (1, 1)$ and no equations. We then have the following result which characterizes $\mathsf{St}(T)$.

Theorem 3.16. *If* T *is compact closed, then there exist an isomorphism of props* $T + X \cong \mathsf{St}(T)$.

Proof. We will only give a sketch of the proof here, the full version of the proof can be found in [Pie18].

To show that $T + X$ is isomorphic to $\mathsf{St}(T)$ we must find two monoidal functors $Z : T \to \mathsf{St}(T)$ and $R : X \to \mathsf{St}(T)$ which together gives an isomorphism $F := \langle Z, R \rangle : T + X \to \mathsf{St}(T)$. Defining R is straightforward as we only need to specify where $-\boxed{x}-$ is mapped to. Thus, let $R(-\boxed{x}-) := (1, \times)$. The intuition here is that $-\boxed{x}-$ acts as a register and hence in the stateful extension this amounts to switching what is in the current state with the new value.

Now, define $Z : T \to \mathsf{St}(T)$ by $Z(t) = (0, t)$. To see that F is an isomorphism we explicitly define the inverse $G : \mathsf{St}(T) \to T + X$ by

$$G\left(\begin{array}{c}\text{s}\\ \text{d}\\ \text{k}\end{array}\begin{array}{c}\text{s}\\ \\ \text{l}\end{array}\right) := \begin{array}{c}\text{s}\\ \text{x}\\ \text{d}\\ \text{k}\end{array}\begin{array}{c}\\ \\ \\ \text{l}\end{array} \tag{14}$$

We first show that $FG = 1$. Thus, let (s, d) be a morphism in $\mathsf{St}(T)$. We then have that

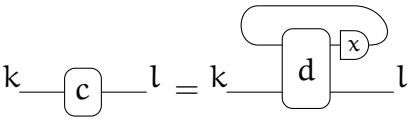
$$\mathsf{FG}((s,d)) = \mathsf{FG}\left(\begin{array}{c}\text{s}\\ \text{d}\\ \text{k}\end{array}\begin{array}{c}\text{s}\\ \\ \text{l}\end{array}\right)$$

$$= F\left(\begin{array}{c}\text{s}\\ \text{x}\\ \text{d}\\ \text{k}\end{array}\begin{array}{c}\\ \\ \\ \text{l}\end{array}\right)$$

$$= \begin{array}{c}\text{s}\\ \\ \text{d}\\ \text{k}\end{array}\begin{array}{c}\text{s}\\ \\ \text{l}\end{array}$$

$$= \begin{array}{c}\text{s}\\ \text{d}\\ \text{k}\end{array}\begin{array}{c}\text{s}\\ \\ \text{l}\end{array}.$$

The other direction uses a technical result which says that every morphism in $\mathsf{T}+\mathsf{X}$ can be written in trace canonical form when T is compact closed. More concretely it says that for every $c : k \to l$ in $\mathsf{T}+\mathsf{X}$ there exists a unique morphism $d : s+k \to s+l$ of T such that



(15)

Thus, given a $c : k \to l$ in $\mathsf{T}+\mathsf{X}$ we have that

$$\mathsf{GF}(c) = \mathsf{GF}\left(\begin{array}{c}\text{k}\\ \boxed{\text{c}}\\ \text{l}\end{array}\right)$$

$$= \mathsf{GF}\left(\begin{array}{c}\text{k}\\ \boxed{\begin{array}{c}\text{d}\\ \text{x}\end{array}}\\ \text{l}\end{array}\right)$$

$$= \mathsf{G}\left(\begin{array}{c}\text{s}\\ \boxed{\text{d}}\\ \text{k}\end{array}\right)$$

$$= \begin{array}{c}\text{k}\\ \boxed{\begin{array}{c}\text{d}\\ \text{x}\end{array}}\\ \text{l}\end{array}$$

$$= \begin{array}{c}\text{k}\\ \boxed{\text{c}}\\ \text{l}\end{array}.$$

□

The moral of the story is that adding state to a compact closed prop, is the same as adding a generator which acts as a synchronous register. Now, as Rc_a is compact closed we define the **stateful affine resource calculus** to be $\mathsf{Rc}_{sa} := \mathsf{Rc}_a + \mathsf{X}$. We then have a sound and complete semantics, i.e., an isomorphism $[\![-]\!]_{sa} : \mathsf{Rc}_{sa} \to \mathsf{St}(\mathsf{PolyRel})$ via the composition

$$\mathsf{Rc}_a + \mathsf{X} \xrightarrow{\langle Z,R\rangle} \mathsf{St}(\mathsf{Rc}_a) \xrightarrow{\mathsf{St}([\![-]\!]_a)} \mathsf{St}(\mathsf{PolyRel}). \tag{16}$$

4. Putting it all together

4.1. The Petri prop. We are now in a position to properly define the prop Petri in which we can embed our previous notion of Petri nets. The benefit of this is that we immediately get a notion of an *open Petri net* which lends itself to being studied compositionally as opposed to monolithically.

Definition 4.1. *Let* Petri *be the prop* $\mathsf{Rc}+\mathsf{Pl}$ *where* Pl *is the prop generated by* $\to\!\bigcirc\!-$: $(1,1)$ *and no equations.*

Theorem 3.16 implies that Petri is isomorphic to $\mathsf{St}(\mathsf{Rc})$. However, this leads to an unnatural semantics in $\mathsf{St}(\mathsf{AddRel})$. We therefore start by choosing an appropriate semantics $[\![-]\!]_p : \mathsf{Petri} \to \mathsf{St}(\mathsf{AddRel})$ and then use the semantics to interpret Petri in $\mathsf{Rc}_s := \mathsf{Rc}+\mathsf{X}$. To specify a semantics $[\![-]\!]_p : \mathsf{Petri} \to \mathsf{St}(\mathsf{AddRel})$ it suffices to specify the semantics of $\to\!\bigcirc\!-$, seeing as the semantics of Rc is given by $[\![-]\!] : \mathsf{Rc} \to \mathsf{AddRel}$.

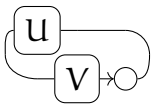
Definition 4.2. *Define* $[\![-]\!]_p : \mathsf{Petri} \to \mathsf{St}(\mathsf{AddRel})$ *to be the unique extension of* $[\![-]\!] : \mathsf{Rc} \to \mathsf{AddRel}$ *such that*

$$[\![\to\!\bigcirc\!-]\!]_p := \left\{ \left(\binom{p}{m}, \binom{p+m-n}{n} \right) \mid n \leqslant p \right\} = \left[\!\!\left[\quad \right]\!\!\right]. \tag{17}$$

The intuition behind $[\![-]\!]_p$ is that it interprets $\to\!\bigcirc\!-$ to currently hold p tokens and ensure that the tokens that flows out of the state are no more than what it currently holds.

For the prop Petri to be of use it must tell us how to reinterpret Petri nets in this new formalism.

Definition 4.3 ([Pie18]). *Let* $\mathcal{P} = (\mathsf{P}, \mathsf{T}, {}^\circ-, -^\circ)$ *be a Petri net. Define* $d_{\mathcal{P}} \in \mathsf{Petri}(0,0)$ *to be*



(18)

where U *and* V *are the matrix representations of* ${}^\circ-$ *and* $-^\circ$ *respectively.*

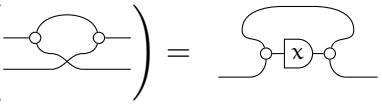
One needs to be careful in the above definition to check that $\mathcal{P} \mapsto d_{\mathcal{P}}$ is independent of the ordering on the transitions and places. The interested reader can see [Pie18, Lemma 144] for the details of this.

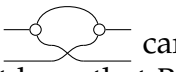
For Petri to be a good extension of Petri nets, the semantics we are interested in, i.e., the firing semantics, should remains the same after the assignment $\mathcal{P} \mapsto d_{\mathcal{P}}$.

Proposition 4.4 ([Pie18]). *Let* $\mathcal{P} = (\mathsf{P}, \mathsf{T}, {}^\circ-, -^\circ)$ *be a Petri net. We then have that* $\mathsf{Fire}(\mathcal{P}) \sim [\![d_{\mathcal{P}}]\!]$, *where* $\sim$ *is the equivalence relation on morphisms of* $\mathsf{St}(\mathsf{AddRel})$.

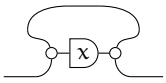
4.2. Expressiveness of Rc_s and Rc_{sa}. Having defined the semantics of Petri, a natural question is how it compares to Rc_s. Using G from the proof of Theorem 3.16 tells us what the image of $\to\!\bigcirc\!-$ must be in Rc_s. We have that

$$G\left(\quad \right) = \quad \tag{19}$$



The full details of why this must be the image of  can be found in [Pie18]. Thus, for $\to\!\bigcirc\!-$ to have the semantics we intend, we must have that $\mathsf{P} : \mathsf{Petri} \to \mathsf{Rc}_s$ is given by

$$\mathsf{P}(\to\!\bigcirc\!-) := \tag{20}$$



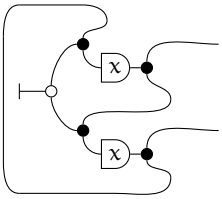
and the identity on Rc. We then have the following result.

Theorem 4.5 ([Pie18]). *For all morphisms* d *in* Petri *we have that* $[\![d]\!]_p \neq [\![-\boxed{x}-]\!]_s$.

Proof. Let d be a morphism in Petri. The claim follows if we can show that $[\![d]\!]_p$ has an element which $[\![-\boxed{x}-]\!]_s$ does not have. Thus, consider the pair $\left(\binom{a}{0}, \binom{a}{0} \right)$. Using structural induction on d together with the definition of $[\![\to\!\bigcirc\!-]\!]$ we can conclude that $\left(\binom{a}{0}, \binom{a}{0} \right) \in [\![d]\!]_p$. Now, as the image of $-\boxed{x}-$ under $\langle Z, R \rangle$ is $(1, \times)$ it follows that $\left(\binom{a}{0}, \binom{a}{0} \right) \notin [\![-\boxed{x}-]\!]_s$. □

The main benefit of using $-\boxed{x}-$ is that we get much more predictable behaviour.

4.3. Conductors. Using the full power of Rc_{sa} we can define a *conductor* which will allow us to synchronize the asynchronous Petri places $\to\!\bigcirc\!-$. This will allow us to gain precise control of the dining philosophers and ensure that no one starves. To gain some intuition about this conductor, consider the following circuit



(21)

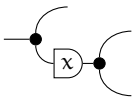
The semantics of this diagram in $\mathsf{St}(\mathsf{PolyRel})$ is the following

$$\left[\!\!\left[\right.\right. \quad \left.\left. \right]\!\!\right]_{sa} = \left\{ \left(\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \right), \left(\begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \right) \right\}. \tag{22}$$

The meaning of the above in words is the following: the only allowed states of the registers are $(1,0)$ and $(0,1)$, and the updated state after one "clock-cycle" is the bit-shifted value of the current state. Moreover, the output of the two wires is the current state. In this way, if we imagine the clock continuing to tick, then the output of the two wires on the right will cycle between $(1,0)$ and $(0,1)$ indefinitely.

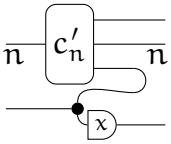
To make this more formal, we make the following definition.

Definition 4.6. *Let*

$c_1' :=$ 

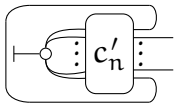
(23)

and define the $n+1$*th* ***pre-conductor*** *to be*

$c_{n+1}' :=$ 

(24)

The n*th* ***conductor*** *is defined to be the circuit*

$c_n :=$ 

(25)

We then have the following result.

Proposition 4.7. *The semantics of the* n*th conductor is given by*

$$[\![c_n]\!]_{sa} = \left\{ \left(e_i^n, \begin{pmatrix} e_i^n \\ e_{i+1 \,(\mathrm{mod})\, n}^n \end{pmatrix} \right) \mid i \in \underline{n} \right\} \tag{26}$$

where $\underline{n} = \{0, 1, \ldots, n-1\}$ *and* e_i^n *is the* $i+1$*th unit vector in* $\mathbb{N}^n$*.*

Proof. One can verify this by using the isomorphism constructed in [Pie18], going backwards from $\mathsf{St}(\mathsf{PolyRel})$ to Rc_{sa}. The details are quite tedious, so we omit them here for brevity. □

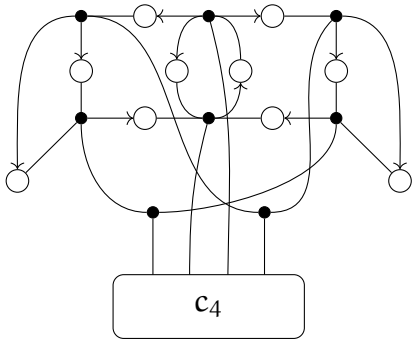


FIGURE 4. A conductor connected to a single philosopher. With the starting state is as indicated in Figure 1 the conductor forces the philosopher to perform the transitions in the correct order.

We can now use the conductors to precisely control the actions of the philosophers. To see how this might be done, consider how the conductor is connected to a single philosopher in Figure 4.[1] Given that the starting state is as in Figure 1 then the following sequence of firings are forced:

(1) The philosopher picks up the right and left fork simultaneously.
(2) The philosopher starts eating.
(3) The philosopher stops eating.
(4) The philosopher puts both forks down simultaneously.

The sequence then repeats indefinitely. Now, if we had used c_{20} instead of c_4, then we could ensure every philosopher got to eat in order. This is by no means the most efficient solution, but it is a process that cannot end into a deadlock. Hence no philosopher dies of starvation, which is our overall goal.

References

[Pet62] Carl Adam Petri. "Kommunikation mit Automaten". Doctoral Dissertation. PhD thesis. Bonn, Germany: University of Bonn, 1962.

[Law63] F. William Lawvere. "Functorial Semantics of Algebraic Theories". In: *Proceedings of the National Academy of Sciences of the United States of America* 50.5 (1963), pp. 869–872. ISSN: 00278424, 10916490. URL: http://www.jstor.org/stable/71935 (visited on 07/04/2024).

[Dij65] Edsger W. Dijkstra. "Dining Philosophers Problem". In: Presented as a student exam exercise. 1965.

[Hoa85] C. A. R. Hoare. *Communicating sequential processes*. USA: Prentice-Hall, Inc., 1985. ISBN: 0131532715.

[Sel10] P. Selinger. "A Survey of Graphical Languages for Monoidal Categories". In: *Lecture Notes in Physics*. Springer Berlin Heidelberg, 2010, pp. 289–355. ISBN: 9783642128219. DOI: 10.1007/978-3-642-12821-9_4. URL: http://dx.doi.org/10.1007/978-3-642-12821-9_4.

[1] In order to enhance the readability of the diagram, we have drawn it in a non-standard way. If one is careful enough, then it is possible to translate this into a proper diagram according to the syntax in Rc_{sa}.

[Bru+13] Roberto Bruni et al. "Connector algebras for C/E and P/T nets' interactions". In: *Logical Methods in Computer Science* Volume 9, Issue 3 (Sept. 2013). ISSN: 1860-5974. DOI: 10.2168/lmcs-9(3:16)2013. URL: http://dx.doi.org/10.2168/LMCS-9(3:16)2013.

[Pie18] R Piedeleu. "Picturing resources in concurrency". PhD thesis. University of Oxford, 2018.

[Bon+19a] Filippo Bonchi et al. "Diagrammatic algebra: from linear to concurrent systems". In: *Proc. ACM Program. Lang.* 3.POPL (Jan. 2019). DOI: 10.1145/3290338. URL: https://doi.org/10.1145/3290338.

[Bon+19b] Filippo Bonchi et al. "Graphical Affine Algebra". In: *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 2019, pp. 1–12. DOI: 10.1109/LICS.2019.8785877.