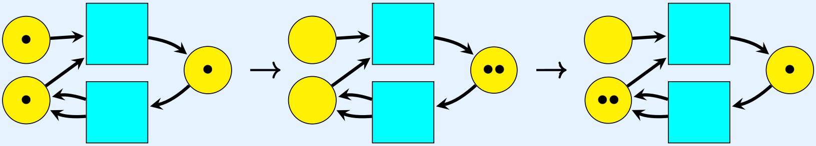
PETRI NETS
as a source of mathematical structures



John Baez

BLAST 2022

2022 August 8

Chemists often use ‘chemical reaction networks’ like this:

$$2\mathrm{H}_2 + \mathrm{O}_2 \to 2\mathrm{H}_2\mathrm{O}$$

$$\mathrm{C} + \mathrm{O}_2 \to \mathrm{CO}_2$$

Mathematically we give the molecules more abstract names, but name the reactions, working with ‘reaction networks’:

$$2A + B \xrightarrow{\tau_1} 2D$$

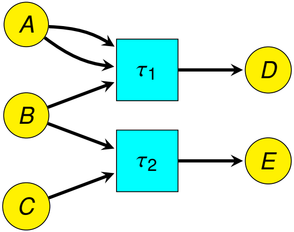
$$C + B \xrightarrow{\tau_2} E$$

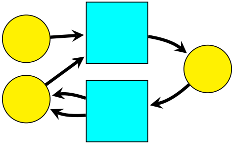
The information in a reaction network can also be expressed using a ‘Petri net’.

This reaction network

$$2A + B \xrightarrow{\tau_1} 2D$$
$$C + B \xrightarrow{\tau_2} E$$

corresponds to this Petri net:





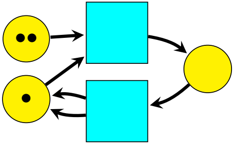
A **Petri net** has:

a set of **places** ,

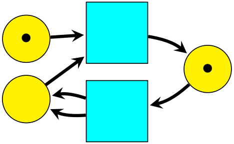
a set of **transitions** ,

a natural number of edges from each place to each transition,

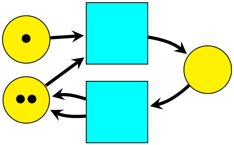
a natural number of edges from each transition to each place.



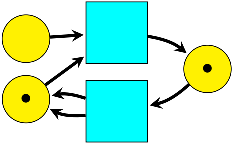
To ‘run’ a Petri net, we start by placing a finite number of **tokens** in each place. This is called a **marking**. Then we can repeatedly move the tokens using the transitions.



To ‘run’ a Petri net, we start by placing a finite number of **tokens** in each place. This is called a **marking**. Then we can repeatedly move the tokens using the transitions.



To ‘run’ a Petri net, we start by placing a finite number of **tokens** in each place. This is called a **marking**. Then we can repeatedly move the tokens using the transitions.



To ‘run’ a Petri net, we start by placing a finite number of **tokens** in each place. This is called a **marking**. Then we can repeatedly move the tokens using the transitions.

Mathematically, a **Petri net** is a diagram like this:

$$T \underset{t}{\overset{s}{\rightrightarrows}} \mathbb{N}[S]$$

S is the set of **places**,

T is the set of **transitions**,

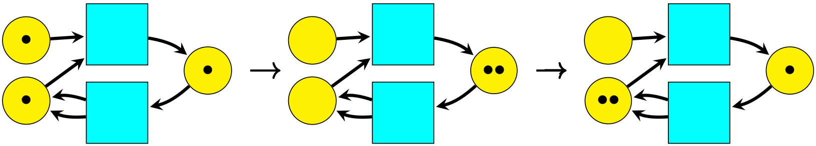
$\mathbb{N}[S]$ is the set of **markings**:
formal finite sums of elements of S.

More mathematically, $\mathbb{N}[S]$ is the underlying set of the free commutative monoid on S:

$$\mathrm{Set} \underset{K}{\overset{J}{\rightleftarrows}} \mathrm{CommMon} \quad (J \dashv K)$$

$$\mathbb{N} = K \circ J$$

Any Petri net gives a symmetric monoidal category where the objects are markings, the tensor product of objects is *addition* of markings, and the morphisms are generated by transitions.



What kind of symmetric monoidal category? A *commutative* monoidal category!

A **commutative monoidal category** is a commutative monoid object in Cat: a category C with a commutative and associative multiplication

$$\otimes \colon \mathrm{C} \times \mathrm{C} \to \mathrm{C}$$

and a unit for this multiplication:

$$I \in \mathrm{C}$$

Equivalently, it's a symmetric monoidal category where the

- braidings $\beta_{x,y} \colon x \otimes y \to y \otimes x$
- associators $\alpha_{x,y,z} \colon (x \otimes y) \otimes z \to x \otimes (y \otimes z)$, and
- left and right unitors $\lambda_x \colon I \otimes x \to x$, $\rho_x \colon x \otimes I \to x$

are all identity morphisms.

Any Petri net P gives a commutative monoidal category FP for which:

- objects are markings of P;
- morphisms are generated from the transitions of P by composition and tensor product, subject to the laws of a commutative monoidal category.

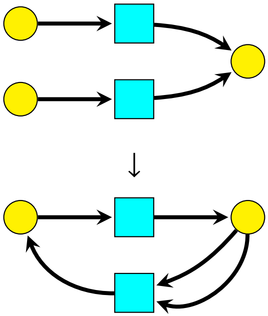
FP is the free commutative monoidal category on the Petri net P. Let me explain this.

There's a category **Petri**, where:

- an object is a Petri net;
- a morphism from $s, t \colon T \to \mathbb{N}[S]$ to $s', t' \colon T \to \mathbb{N}[S']$ is a pair of functions $f \colon T \to T', g \colon S \to S'$ such that these diagrams commute:

$$
\begin{array}{ccc}
T & \xrightarrow{s} & \mathbb{N}[S] \\
{\scriptstyle f}\downarrow & & \downarrow{\scriptstyle \mathbb{N}[g]} \\
T' & \xrightarrow[s']{} & \mathbb{N}[S']
\end{array}
\qquad
\begin{array}{ccc}
T & \xrightarrow{t} & \mathbb{N}[S] \\
{\scriptstyle f}\downarrow & & \downarrow{\scriptstyle \mathbb{N}[g]} \\
T' & \xrightarrow[t']{} & \mathbb{N}[S']
\end{array}
$$

A morphism of Petri nets:



There's also a category **CMCat** of commutative monoidal categories, where:

- objects are commutative monoidal categories;
- morphisms are strict monoidal functors (automatically symmetric).

Theorem (Master). There are adjoint functors

$$\mathrm{Petri} \underset{U}{\overset{F}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathrm{CMCat}$$

with F sending the Petri net P to the free commutative monoidal category FP described earlier.

Figuring out the right adjoint U is not as easy as you might think:

- Jade Master, Generalized Petri nets, arXiv:1904.09091

We can turn a category C into a preorder L_1C by decreeing $x \leq y$ whenever there exists a morphism $f\colon x \to y$.

We can turn a preorder X into a poset $L_2 X$ by decreeing $x = y$ whenever $x \leq y$ and $y \leq x$.

We can then turn a poset Y into a set $L_3 Y$ by decreeing $x = y$ whenever $x \leq y$ or $y \leq x$ and closing this relation under transitivity.

$$\text{Cat} \xrightarrow{L_1} \text{Preord} \xrightarrow{L_2} \text{Poset} \xrightarrow{L_3} \text{Set}$$

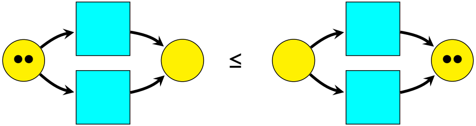
$$\mathrm{Cat} \xrightarrow{L_1} \mathrm{Preord} \xrightarrow{L_2} \mathrm{Poset} \xrightarrow{L_3} \mathrm{Set}$$

All the above functors are left adjoints, but they also preserve products, so they preserve commutative monoid objects. We thus get functors

$$\mathrm{Petri} \xrightarrow{F} \mathrm{CMCat} \xrightarrow{L_1} \mathrm{CMPreord} \xrightarrow{L_2} \mathrm{CMPoset} \xrightarrow{L_3} \mathrm{CMSet}$$

$$\mathsf{Petri} \xrightarrow{F} \mathsf{CMCat} \xrightarrow{L_1} \mathsf{CMPreord}$$

Given a Petri net P, the commutative monoidal preorder $L_1 FP$ has markings of P as elements, and $x \leq y$ if y is **reachable** from x: that is, there exists a morphism $f\colon x \to y$ in FP.



The **reachability problem** asks us to decide if $x \leq y$ when x, y are two markings of a Petri net.

For example, given these chemical reactions:

$$C + O_2 \longrightarrow CO_2$$

$$CO_2 + NaOH \longrightarrow NaHCO_3$$

$$NaHCO_3 + HCl \longrightarrow H_2O + NaCl + CO_2$$

can you turn

$$C + O_2 + NaOH + HCl$$

into

$$CO_2 + H_2O + NaCl?$$

Theorem (Czerwinski–Lasota–Lazic–Leroux–Mazowiecki). For any algorithm that decides the reachability problem, the worst-case runtime exceeds

$$2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}$$

where the number of layers in the tower can be any function $2^N, 2^{2^N}, 2^{2^{2^N}}, \ldots$. Here N is the size of the problem: the sum of the number of generating places, the total number of inputs and outputs of all transitions, and the number of summands in the markings x, y for which the problem is posed.

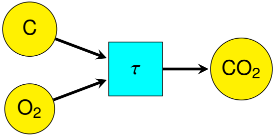
Theorem (Leroux–Schmitz). There is an algorithm that decides the reachability problem whose runtime is bounded by an Ackermann function of N.

$$\mathrm{Petri} \xrightarrow{F} \mathrm{CMCat} \xrightarrow{L_1} \mathrm{CMPreord} \xrightarrow{L_2} \mathrm{CMPoset} \xrightarrow{L_3} \mathrm{CMSet}$$

Given a Petri net P, the commutative monoid $L_3 L_2 L_1 FP$ has equivalence classes of markings of P as elements: we impose an equation $x = y$ whenever x is reachable from y or vice versa.

Any presentation of a commutative monoid can be expressed using a Petri net with one place per generator and one transition per relation.

For example:



gives the commutative monoid with three generators C, O_2, CO_2 and one relation

$$C + O_2 = CO_2$$

Theorem (Cardoza). The word problem in any fixed finitely presented commutative monoid can be solved in linear time — linear in the sum of the lengths of the two words being checked for equality.

Theorem (Mayr–Meyer). The word problem for finitely presented commutative monoids can be solved in exponential space: exponential in the sum of the lengths of the words in all the relations and the two words being checked for equality. It is exponential space complete.

If it can be solved in exponential time, then EXPSPACE = EXPTIME.

It can be solved in **doubly exponential time**: a runtime $\leq 2^{2^{P(N)}}$ for some polynomial P. It cannot be solved in polynomial time.

There is also a left adjoint functor

$$\text{Poset} \xrightarrow{L} \text{Suplat}$$

Here **Suplat** is the category of **suplattices**, where

- an object is a poset where all subsets have suprema;
- a morphism is an order-preserving map preserving all suprema.

If X is a poset, LX is the poset of **downsets** of X, i.e. downwards-closed subsets, ordered by inclusion. We have an inclusion of posets

$$X \hookrightarrow LX$$

sending $x \in X$ to the downset $\{y \in X : y \leq x\}$.

The supremum in LX is given by *union* of downsets of X.

Any suplattice has, not only suprema of all subset, but also infima.

But beware: maps of suplattices need not preserve infima!

A free suplattice on a poset is also **cartesian closed**, meaning that $x \wedge \cdot$ has a right adjoint $x \Rightarrow \cdot$, or in other words:

$$x \wedge y \leq z \text{ if and only if } x \leq (y \Rightarrow z)$$

There is a tensor product of suplattices such that a map of suplattices

$$L \otimes L' \to M$$

is the same as an order-preserving map

$$L \times L' \to M$$

that preserves suprema in each argument.

It resembles the tensor product of vector spaces, or modules of a commutative ring.

The functor

$$\text{Poset} \xrightarrow{L} \text{Suplat}$$

has

$$L(X \times X') \cong L(X) \otimes L(X')$$

for any posets X, X'. In fact it is a symmetric monoidal functor.

It thus sends commutative monoid objects in $(\text{Poset}, \times)$ to commutative monoid objects in $(\text{Suplat}, \otimes)$, which are called **commutative (unital) quantales**.

Indeed we have left adjoints

$$\text{Petri} \xrightarrow{F} \text{CMCat} \xrightarrow{L_1} \text{CMPreord} \xrightarrow{L_2} \text{CMPoset} \xrightarrow{L} \text{CQuant}$$

Concretely, a commutative quantale is a suplattice X with a commutative associative multiplication

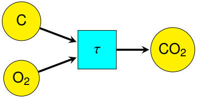
$$X \times X \to X$$

that distributes over arbitrary suprema, and a unit object for this multiplication.

When we get a commutative quantale from a Petri net P, this multiplication comes from the tensor product in our commutative monoidal category FP, which we've been calling $+$, as in $\mathrm{C} + \mathrm{O}_2$. The unit object is 0: "nothing".

$$\mathrm{Petri} \xrightarrow{F} \mathrm{CMCat} \xrightarrow{L_1} \mathrm{CMPreord} \xrightarrow{L_2} \mathrm{CMPoset} \xrightarrow{L} \mathrm{CQuant}$$

So, in the commutative quantale coming from the Petri net



we have relations like

$$C + O_2 \leq CO_2$$

$$C \leq C \vee O_2 \qquad O_2 \leq C \vee O_2$$

$$C + (C \vee O_2) = 2C \vee (C + O_2)$$

$$C + (C \vee O_2) \leq 2C \vee CO_2$$

$$(C \vee O_2) \wedge C = C$$

Thus, the commutative quantale coming from a Petri net describes a "logic of resources". It has *three* commutative monoid structures:

- $x + y$: "what you have is x together with y", as in $C + O_2$ = "what you have is a carbon atom together with an oxygen molecule".

- $x \vee y$: "what you have is an x or an y", as in $C \vee O_2$ = "what you have is a carbon atom or an oxygen molecule".

- $x \wedge y$: "what you have is both an x and a y", as in $C \wedge O_2$ = "what you have is both an carbon atom and an oxgen molecule" = $\emptyset$.

The material on commutative quantales from Petri nets is my interpretation of this:

- Uffe Engberg and Glynn Winskel, Petri nets as models of linear logic, in *Colloquium on Trees in Algebra and Programming*, Springer, Berlin, 1990, pp. 147–161.

They also show more. For example: in the commutative quantale coming from a Petri net, $x + \cdot$ has a right adjoint $x \multimap \cdot$. In other words:

$$x + y \leq z \text{ if and only if } x \leq (y \multimap z)$$

AND THERE'S MORE... BUT NOT TODAY!