
INSTITUT DE RECHERCHE EN INFORMATIQUE FONDAMENTALE

# Towards Executable Applied Category Theory

CAP'21, IHÉS, November 30, 2021

Nicolas Behr

CNRS, Université de Paris, IRIF (UMR 8243)

université PARIS DIDEROT

Université de Paris

# Plan of the talk:

I. A Case Study in Applied Category Theory: from Categorical Rewriting to Rule-algebraic Combinatorics

II. The `coreact.wiki` Initiative

# Plan of the talk:

I. A Case Study in Applied Category Theory: from Categorical Rewriting to Rule-algebraic Combinatorics

II. The `coreact.wiki` Initiative

# A quick tour of categorical rewriting

$$\begin{tikzcd} O & I \arrow[l, hook', "r"'] \arrow[d, tail, "m"] \\ & X \end{tikzcd}$$

# A quick tour of categorical rewriting

$$
\begin{tikzcd}
O \arrow[d, tail] & I \arrow[l, "r"'] \arrow[d, tail, "m"] \arrow[dl, phantom, "\mathbb{T}"] \\
r_m(X) & X \arrow[l, "r_m"]
\end{tikzcd}
$$

"type $\mathbb{T}$ cell"

# A quick tour of categorical rewriting

$$\begin{tikzcd}
O \arrow[d, tail] \arrow[dr, phantom, "\mathbb{T}"] & I \arrow[l, "r"'] \arrow[d, tail, "m"] \\
r_m(X) & X \arrow[l, "r_m"]
\end{tikzcd}
\quad := \quad
\begin{tikzcd}
O \arrow[d, tail] \arrow[dr, phantom, "B_{\mathbb{T}}"] & K \arrow[l, tail] \arrow[r, tail] \arrow[d, tail] \arrow[dr, phantom, "A_{\mathbb{T}}"] & I \arrow[d, tail, "m"] \arrow[ll, bend right] \\
r_m(X) & \overline{X} \arrow[l, tail] \arrow[r, tail] & X \arrow[ll, bend left]
\end{tikzcd}$$

**O**utput

**K**eep

**I**nput

**m**atch

“type $\mathbb{T}$ cell”

# Four flavours of categorical rewriting semantics

$$
\begin{tikzcd}
O \arrow[d, tail, "m^*"'] \arrow[dr, phantom, "\mathrm{PO}"] & K \arrow[l, tail, "o"'] \arrow[r, tail, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dr, phantom, "\mathrm{POC}"] & I \arrow[d, tail, "m"] \\
r_m(X) & \bar{X} \arrow[l, tail, "\bar{o}"'] \arrow[r, tail, "\bar{i}"] & X
\end{tikzcd}
$$

($\mathscr{M}$-) **linear** **Double-Pushout (DPO)**

$$
\begin{tikzcd}
O \arrow[d, tail, "m^*"'] \arrow[dr, phantom, "\mathrm{PO}"] & K \arrow[l, "o"'] \arrow[r, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dr, phantom, "\mathrm{mPOC}"] & I \arrow[d, tail, "m"] \\
r_m(X) & \bar{X} \arrow[l, "\bar{o}"'] \arrow[r, "\bar{i}"] & X
\end{tikzcd}
$$

**non-linear** **Double-Pushout (DPO)**

$$
\begin{tikzcd}
O \arrow[d, tail, "m^*"'] \arrow[dr, phantom, "\mathrm{PO}"] & K \arrow[l, tail, "o"'] \arrow[r, tail, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dr, phantom, "\mathrm{FPC}"] & I \arrow[d, tail, "m"] \\
r_m(X) & \bar{X} \arrow[l, tail, "\bar{o}"'] \arrow[r, tail, "\bar{i}"] & X
\end{tikzcd}
$$

($\mathscr{M}$-) **linear** **Sesqui-Pushout (SqPO)**

$$
\begin{tikzcd}
O \arrow[d, tail, "m^*"'] \arrow[dr, phantom, "\mathrm{PO}"] & K \arrow[l, "o"'] \arrow[r, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dr, phantom, "\mathrm{FPC}"] & I \arrow[d, tail, "m"] \\
r_m(X) & \bar{X} \arrow[l, "\bar{o}"'] \arrow[r, "\bar{i}"] & X
\end{tikzcd}
$$

**non-linear** **Sesqui-Pushout (SqPO)**

# Four flavours of categorical rewriting semantics

$$\begin{tikzcd}
O \arrow[d, tail, "m^*"'] & K \arrow[l, tail, "o"'] \arrow[r, tail, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dl, phantom, "\mathrm{PO}"] & I \arrow[d, tail, "m"] \arrow[dl, phantom, "\mathrm{POC}"] \\
r_m(X) & \bar{X} \arrow[l, tail, "\bar{o}"'] \arrow[r, tail, "\bar{i}"] & X
\end{tikzcd}$$

($\mathscr{M}$-) **linear** **Double-Pushout (DPO)**

$$\begin{tikzcd}
O \arrow[d, tail, "m^*"'] & K \arrow[l, "o"'] \arrow[r, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dl, phantom, "\mathrm{PO}"] & I \arrow[d, tail, "m"] \arrow[dl, phantom, "\mathrm{mPOC}"] \\
r_m(X) & \bar{X} \arrow[l, "\bar{o}"'] \arrow[r, "\bar{i}"] & X
\end{tikzcd}$$

**non-linear** **Double-Pushout (DPO)**

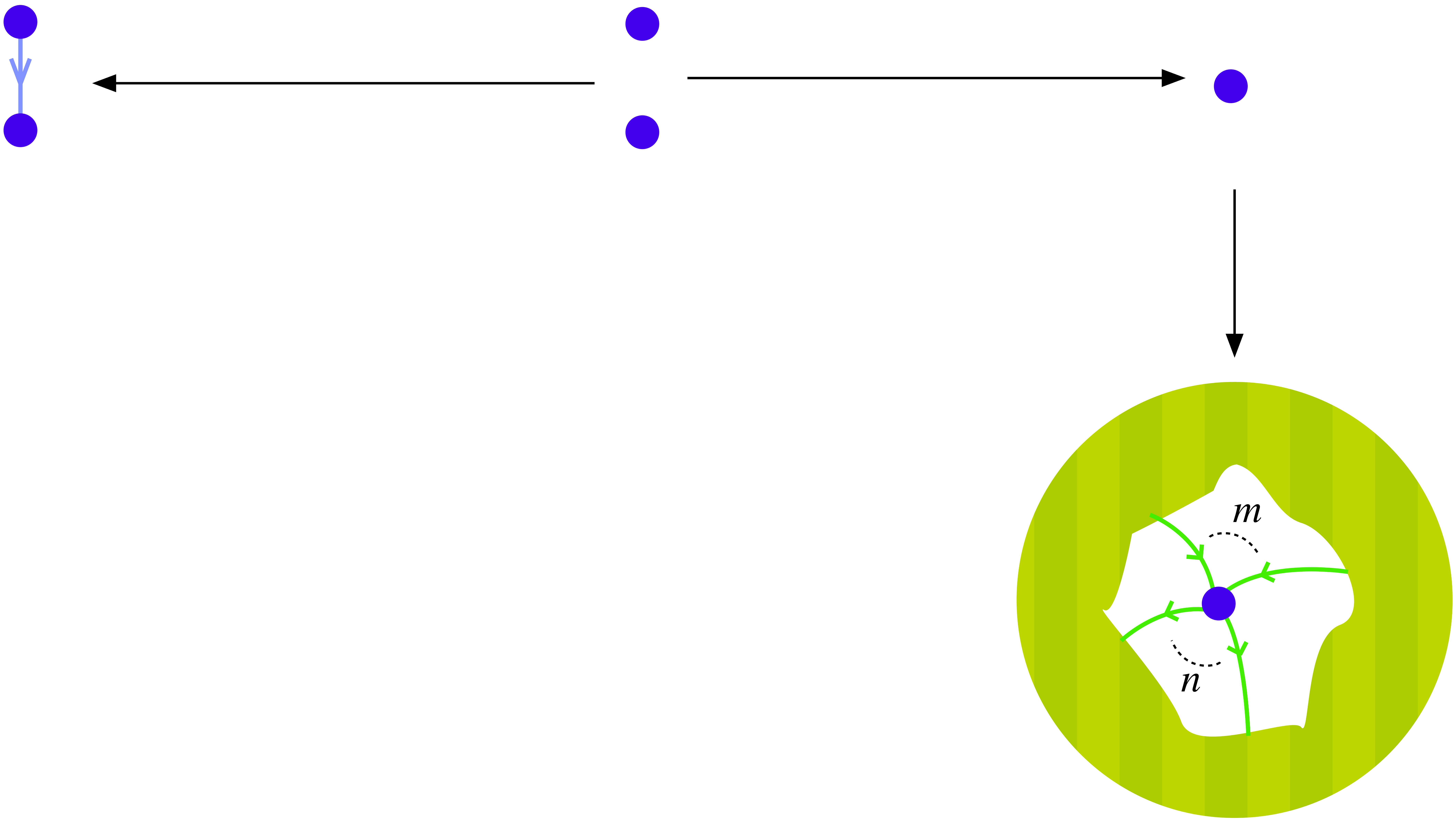
$$\begin{tikzcd}
O \arrow[d, tail, "m^*"'] & K \arrow[l, tail, "o"'] \arrow[r, tail, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dl, phantom, "\mathrm{PO}"] & I \arrow[d, tail, "m"] \arrow[dl, phantom, "\mathrm{FPC}"] \\
r_m(X) & \bar{X} \arrow[l, tail, "\bar{o}"'] \arrow[r, tail, "\bar{i}"] & X
\end{tikzcd}$$

($\mathscr{M}$-) **linear** **Sesqui-Pushout (SqPO)**

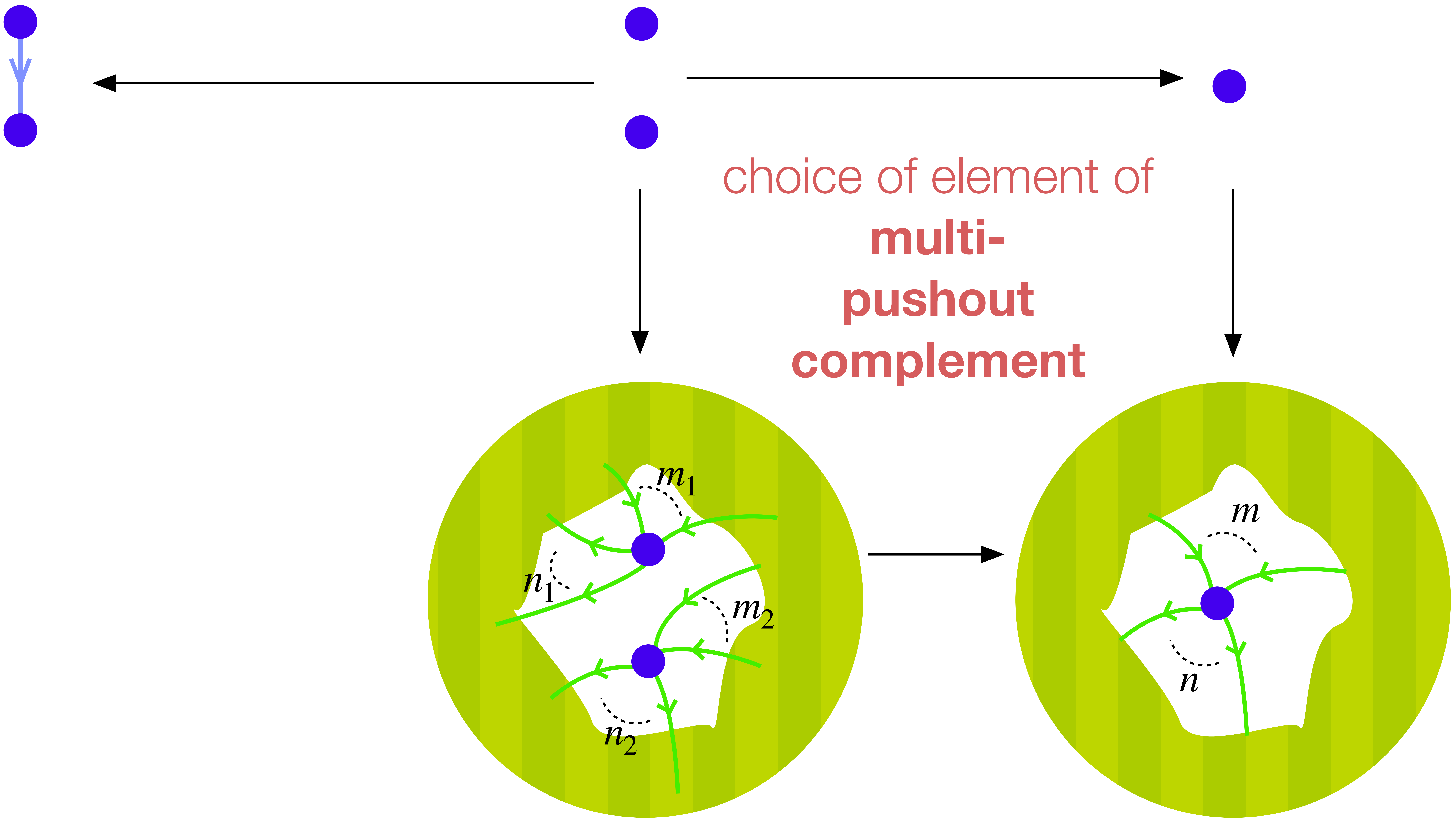
$$\begin{tikzcd}
O \arrow[d, tail, "m^*"'] & K \arrow[l, "o"'] \arrow[r, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dl, phantom, "\mathrm{PO}"] & I \arrow[d, tail, "m"] \arrow[dl, phantom, "\mathrm{FPC}"] \\
r_m(X) & \bar{X} \arrow[l, "\bar{o}"'] \arrow[r, "\bar{i}"] & X
\end{tikzcd}$$

**non-linear** **Sesqui-Pushout (SqPO)**

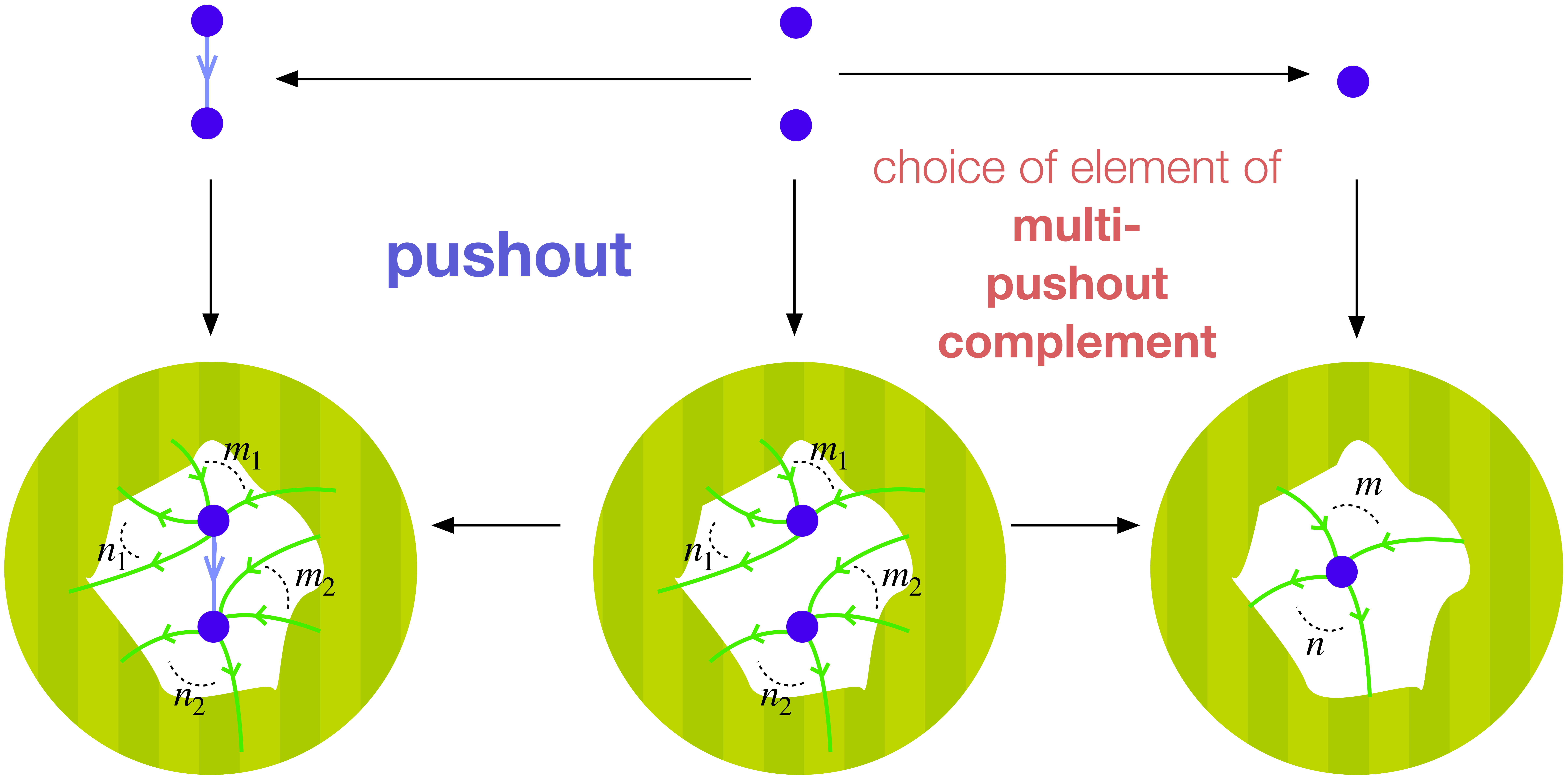
# Cloning in **non-linear Double Pushout (DPO) rewriting**



# Cloning in **non-linear Double Pushout (DPO) rewriting**



# Cloning in **non-linear Double Pushout (DPO) rewriting**



# Four flavours of categorical rewriting semantics

$$
\begin{tikzcd}
O \arrow[d, tail, "m^*"'] \arrow[dr, phantom, "\mathrm{PO}"] & K \arrow[l, tail, "o"'] \arrow[r, tail, "i"] \arrow[d, tail, "\overline{m}"] \arrow[dr, phantom, "\mathrm{POC}"] & I \arrow[d, tail, "m"] \\
r_m(X) & \overline{X} \arrow[l, tail, "\overline{o}"'] \arrow[r, tail, "\overline{i}"] & X
\end{tikzcd}
$$

($\mathscr{M}$-) linear **Double-Pushout (DPO)**

$$
\begin{tikzcd}
O \arrow[d, tail, "m^*"'] \arrow[dr, phantom, "\mathrm{PO}"] & K \arrow[l, "o"'] \arrow[r, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dr, phantom, "\mathrm{mPOC}"] & I \arrow[d, tail, "m"] \\
r_m(X) & \bar{X} \arrow[l, "\bar{o}"'] \arrow[r, "\bar{i}"] & X
\end{tikzcd}
$$

non-linear **Double-Pushout (DPO)**

$$
\begin{tikzcd}
O \arrow[d, tail, "m^*"'] \arrow[dr, phantom, "\mathrm{PO}"] & K \arrow[l, tail, "o"'] \arrow[r, tail, "i"] \arrow[d, tail, "\overline{m}"] \arrow[dr, phantom, "\mathrm{FPC}"] & I \arrow[d, tail, "m"] \\
r_m(X) & \overline{X} \arrow[l, tail, "\overline{o}"'] \arrow[r, tail, "\overline{i}"] & X
\end{tikzcd}
$$

($\mathscr{M}$-) linear **Sesqui-Pushout (SqPO)**

$$
\begin{tikzcd}
O \arrow[d, tail, "m^*"'] \arrow[dr, phantom, "\mathrm{PO}"] & K \arrow[l, "o"'] \arrow[r, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dr, phantom, "\mathrm{FPC}"] & I \arrow[d, tail, "m"] \\
r_m(X) & \bar{X} \arrow[l, "\bar{o}"'] \arrow[r, "\bar{i}"] & X
\end{tikzcd}
$$

non-linear **Sesqui-Pushout (SqPO)**

# Four flavours of categorical rewriting semantics

$\mathscr{M}$**-adhesive categories**
(+ some extra assumptions…)

$$
\begin{tikzcd}
O \arrow[d, tail, "m^*"'] \arrow[dr, phantom, "\mathbf{PO}"] & K \arrow[l, tail, "o"'] \arrow[r, tail, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dr, phantom, "\mathbf{POC}"] & I \arrow[d, tail, "m"] \\
r_m(X) & \bar{X} \arrow[l, tail, "\bar{o}"] \arrow[r, tail, "\bar{i}"'] & X
\end{tikzcd}
$$

($\mathscr{M}$-) **linear Double-Pushout (DPO)**

$$
\begin{tikzcd}
O \arrow[d, tail, "m^*"'] \arrow[dr, phantom, "\mathbf{PO}"] & K \arrow[l, "o"'] \arrow[r, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dr, phantom, "\mathbf{mPOC}"] & I \arrow[d, tail, "m"] \\
r_m(X) & \bar{X} \arrow[l, "\bar{o}"] \arrow[r, "\bar{i}"'] & X
\end{tikzcd}
$$

**non-linear Double-Pushout (DPO)**

$$
\begin{tikzcd}
O \arrow[d, tail, "m^*"'] \arrow[dr, phantom, "\mathbf{PO}"] & K \arrow[l, tail, "o"'] \arrow[r, tail, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dr, phantom, "\mathbf{FPC}"] & I \arrow[d, tail, "m"] \\
r_m(X) & \bar{X} \arrow[l, tail, "\bar{o}"] \arrow[r, tail, "\bar{i}"'] & X
\end{tikzcd}
$$

($\mathscr{M}$-) **linear Sesqui-Pushout (SqPO)**

$$
\begin{tikzcd}
O \arrow[d, tail, "m^*"'] \arrow[dr, phantom, "\mathbf{PO}"] & K \arrow[l, "o"'] \arrow[r, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dr, phantom, "\mathbf{FPC}"] & I \arrow[d, tail, "m"] \\
r_m(X) & \bar{X} \arrow[l, "\bar{o}"] \arrow[r, "\bar{i}"'] & X
\end{tikzcd}
$$

**non-linear Sesqui-Pushout (SqPO)**

# Four flavours of categorical rewriting semantics

**rm-adhesive categories**

**$\mathscr{M}$-adhesive categories** (+ some extra assumptions…)

$$\begin{tikzcd}
O \arrow[d, tail, "m^*"'] \arrow[dr, phantom, "\mathbf{PO}"] & K \arrow[l, tail, "o"'] \arrow[r, tail, "i"] \arrow[d, tail, "\overline{m}"] \arrow[dr, phantom, "\mathbf{POC}"] & I \arrow[d, tail, "m"] \\
r_m(X) & \overline{X} \arrow[l, tail, "\overline{o}"'] \arrow[r, tail, "\overline{i}"] & X
\end{tikzcd}$$

($\mathscr{M}$-) **linear Double-Pushout (DPO)**

$$\begin{tikzcd}
O \arrow[d, tail, "m^*"'] \arrow[dr, phantom, "\mathbf{PO}"] & K \arrow[l, "o"'] \arrow[r, "i"] \arrow[d, tail, "\overline{m}"] \arrow[dr, phantom, "\mathbf{mPOC}"] & I \arrow[d, tail, "m"] \\
r_m(X) & \overline{X} \arrow[l, "\overline{o}"'] \arrow[r, "\overline{i}"] & X
\end{tikzcd}$$

**non-linear Double-Pushout (DPO)**

$$\begin{tikzcd}
O \arrow[d, tail, "m^*"'] \arrow[dr, phantom, "\mathbf{PO}"] & K \arrow[l, tail, "o"'] \arrow[r, tail, "i"] \arrow[d, tail, "\overline{m}"] \arrow[dr, phantom, "\mathbf{FPC}"] & I \arrow[d, tail, "m"] \\
r_m(X) & \overline{X} \arrow[l, tail, "\overline{o}"'] \arrow[r, tail, "\overline{i}"] & X
\end{tikzcd}$$

($\mathscr{M}$-) **linear Sesqui-Pushout (SqPO)**

$$\begin{tikzcd}
O \arrow[d, tail, "m^*"'] \arrow[dr, phantom, "\mathbf{PO}"] & K \arrow[l, "o"'] \arrow[r, "i"] \arrow[d, tail, "\overline{m}"] \arrow[dr, phantom, "\mathbf{FPC}"] & I \arrow[d, tail, "m"] \\
r_m(X) & \overline{X} \arrow[l, "\overline{o}"'] \arrow[r, "\overline{i}"] & X
\end{tikzcd}$$

**non-linear Sesqui-Pushout (SqPO)**

# Four flavours of categorical rewriting semantics

**rm-adhesive categories**

**$\mathscr{M}$-adhesive categories**
(+ some extra assumptions…)

$$\begin{tikzcd}
O & K \arrow[l, tail, "o"'] \arrow[r, tail, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dl, phantom, "\textbf{PO}"] \arrow[dr, phantom, "\textbf{POC}"] & I \arrow[d, tail, "m"] \\
r_m(X) \arrow[from=u, tail, "m^*"'] & \bar{X} \arrow[l, tail, "\bar{o}"'] \arrow[r, tail, "\bar{i}"] & X
\end{tikzcd}$$

**($\mathscr{M}$-) linear Double-Pushout (DPO)**

$$\begin{tikzcd}
O & K \arrow[l, "o"'] \arrow[r, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dl, phantom, "\textbf{PO}"] \arrow[dr, phantom, "\textbf{mPOC}"] & I \arrow[d, tail, "m"] \\
r_m(X) \arrow[from=u, tail, "m^*"'] & \bar{X} \arrow[l, "\bar{o}"'] \arrow[r, "\bar{i}"] & X
\end{tikzcd}$$

**non-linear Double-Pushout (DPO)**

$$\begin{tikzcd}
O & K \arrow[l, tail, "o"'] \arrow[r, tail, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dl, phantom, "\textbf{PO}"] \arrow[dr, phantom, "\textbf{FPC}"] & I \arrow[d, tail, "m"] \\
r_m(X) \arrow[from=u, tail, "m^*"'] & \bar{X} \arrow[l, tail, "\bar{o}"'] \arrow[r, tail, "\bar{i}"] & X
\end{tikzcd}$$

**($\mathscr{M}$-) linear Sesqui-Pushout (SqPO)**

$$\begin{tikzcd}
O & K \arrow[l, "o"'] \arrow[r, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dl, phantom, "\textbf{PO}"] \arrow[dr, phantom, "\textbf{FPC}"] & I \arrow[d, tail, "m"] \\
r_m(X) \arrow[from=u, tail, "m^*"'] & \bar{X} \arrow[l, "\bar{o}"'] \arrow[r, "\bar{i}"] & X
\end{tikzcd}$$

**non-linear Sesqui-Pushout (SqPO)**

**quasi-topoi**

## multigraphs

## simple graphs

**directed**

$$
\begin{tikzcd}
\mathsf{E} \arrow[r, "\iota"] \arrow[d, "\varphi_{\mathsf{E}}"] & \mathsf{V}\times\mathsf{V} \arrow[d, "\Delta(\varphi_{\mathsf{V}})"] & \mathsf{V} \arrow[l, "\Delta"'] \arrow[d, "\varphi_{\mathsf{V}}"] \\
\mathsf{E}' \arrow[r, "\iota'"] & \mathsf{V}'\times\mathsf{V}' & \mathsf{V}' \arrow[l, "\Delta"']
\end{tikzcd}
$$

$\underline{\mathsf{Set}}/\Delta$

**undirected**

**multigraphs** **simple graphs**

**directed**

$$\begin{tikzcd} E \arrow[r, "\iota"] \arrow[d, "\varphi_E"] & V \times V \arrow[d, "\varphi_V \times \varphi_V"] & V \arrow[l, "\Delta"'] \arrow[d, "\varphi_V"] \\ E' \arrow[r, "\iota'"] & V' \times V' & V' \arrow[l, "\Delta"'] \end{tikzcd}$$

**comma category** over an **adhesive category**
& functor **preserves all pullbacks**
$\Rightarrow$ **adhesive category**

$\underline{\mathsf{Set}} / \Delta$

**undirected**

# multigraphs

# simple graphs

**directed**

$$\begin{tikzcd}
\mathsf{E} \arrow[r, "\iota"] \arrow[d, "\varphi_{\mathsf{E}}"] & \mathsf{V}\times\mathsf{V} \arrow[d, "\Delta(\varphi_{\mathsf{V}})"] & \mathsf{V} \arrow[l, "\Delta"'] \arrow[d, "\varphi_{\mathsf{V}}"] \\
\mathsf{E}' \arrow[r, "\iota'"] & \mathsf{V}'\times\mathsf{V}' & \mathsf{V}' \arrow[l, "\Delta"']
\end{tikzcd}$$

**comma category** over an **adhesive category**
& functor **preserves all pullbacks**
$\Rightarrow$ **adhesive category**

$\underline{\mathsf{Set}}/\Delta$

$$\begin{tikzcd}
\mathsf{E} \arrow[r, hook, "\iota"] \arrow[d, "\varphi_{\mathsf{E}}"] & \mathsf{V}\times\mathsf{V} \arrow[d, "\Delta(\varphi_{\mathsf{V}})"] & \mathsf{V} \arrow[l, "\Delta"'] \arrow[d, "\varphi_{\mathsf{V}}"] \\
\mathsf{E}' \arrow[r, hook, "\iota'"] & \mathsf{V}'\times\mathsf{V}' & \mathsf{V}' \arrow[l, "\Delta"']
\end{tikzcd}$$

$\underline{\mathsf{Set}}/\!\!/\Delta$

**undirected**

**directed**

## multigraphs

$$\begin{tikzcd} E \arrow[r, "\iota"] \arrow[d, "\varphi_E"'] & V \times V \arrow[d, "\varphi_V \times \varphi_V"] & V \arrow[l, "\Delta"'] \arrow[d, "\varphi_V"] \\ E' \arrow[r, "\iota'"] & V' \times V' & V' \arrow[l, "\Delta"'] \end{tikzcd}$$

**comma category** over an **adhesive category**
& functor **preserves all pullbacks**
$\Rightarrow$ **adhesive category**

$\underline{\mathsf{Set}} / \Delta$

## simple graphs

$$\begin{tikzcd} E \arrow[r, hook, "\iota"] \arrow[d, "\varphi_E"'] & V \times V \arrow[d, "\varphi_V \times \varphi_V"] & V \arrow[l, "\Delta"'] \arrow[d, "\varphi_V"] \\ E' \arrow[r, hook, "\iota'"] & V' \times V' & V' \arrow[l, "\Delta"'] \end{tikzcd}$$

**Artin gluing** based upon **quasi-topos**
& functor **preserves all pullbacks**
$\Rightarrow$ **quasi-topos** (but **not adhesive!**)

$\underline{\mathsf{Set}} /\!\!/ \Delta$

**undirected**

| | **multigraphs** | **simple graphs** |
|---|---|---|
| **directed** | see below | see below |
| **undirected** | see below | |

# multigraphs

## directed

$$\begin{tikzcd}
\mathsf{E} \arrow[r, "\iota"] \arrow[d, "\varphi_{\mathsf{E}}"'] & \mathsf{V}\times\mathsf{V} \arrow[d, "\varphi_{\mathsf{V}}\times\varphi_{\mathsf{V}}"] & \mathsf{V} \arrow[l, "\Delta"'] \arrow[d, "\varphi_{\mathsf{V}}"] \\
\mathsf{E}' \arrow[r, "\iota'"] & \mathsf{V}'\times\mathsf{V}' & \mathsf{V}' \arrow[l, "\Delta"']
\end{tikzcd}$$

**comma category** over an **adhesive category**
& functor **preserves all pullbacks**
$\Rightarrow$ **adhesive category**

$\underline{\mathsf{Set}}/\Delta$

## undirected

$$\begin{tikzcd}
\mathsf{E} \arrow[r, "\iota"] \arrow[d, "\varphi_{\mathsf{E}}"'] & \mathcal{P}^{(1,2)}(\mathsf{V}) \arrow[d, "\mathcal{P}^{(1,2)}(\varphi_{\mathsf{V}})"] & \mathsf{V} \arrow[l, "\mathcal{P}^{(1,2)}"'] \arrow[d, "\varphi_{\mathsf{V}}"] \\
\mathsf{E}' \arrow[r, "\iota'"] & \mathcal{P}^{(1,2)}(\mathsf{V}') & \mathsf{V}' \arrow[l, "\mathcal{P}^{(1,2)}"']
\end{tikzcd}$$

$\underline{\mathsf{Set}}/\mathcal{P}^{(1,2)}$

# simple graphs

## directed

$$\begin{tikzcd}
\mathsf{E} \arrow[r, hook, "\iota"] \arrow[d, "\varphi_{\mathsf{E}}"'] & \mathsf{V}\times\mathsf{V} \arrow[d, "\varphi_{\mathsf{V}}\times\varphi_{\mathsf{V}}"] & \mathsf{V} \arrow[l, "\Delta"'] \arrow[d, "\varphi_{\mathsf{V}}"] \\
\mathsf{E}' \arrow[r, hook, "\iota'"] & \mathsf{V}'\times\mathsf{V}' & \mathsf{V}' \arrow[l, "\Delta"']
\end{tikzcd}$$

**Artin gluing** based upon **quasi-topos**
& functor **preserves all pullbacks**
$\Rightarrow$ **quasi-topos** (but **not adhesive!**)

$\underline{\mathsf{Set}}/\!\!/\Delta$

| | **multigraphs** | **simple graphs** |
|---|---|---|
| **directed** | **comma category** over an **adhesive category** & functor **preserves all pullbacks** ⇒ **adhesive category** | **Artin gluing** based upon **quasi-topos** & functor **preserves all pullbacks** ⇒ **quasi-topos** (but **not adhesive!**) |
| | $\underline{\mathsf{Set}}\,/\Delta$ | $\underline{\mathsf{Set}}\,/\!\!/\Delta$ |
| **undirected** | **comma category** over an **adhesive category** & functor **preserves only pullbacks along monos** ⇒ $\mathscr{M}$**-adhesive category** | |
| | $\underline{\mathsf{Set}}\,/\mathcal{P}^{(1,2)}$ | |

$$\begin{tikzcd} \mathsf{E} \arrow[r, "\iota"] \arrow[d, "\varphi_\mathsf{E}"] & \mathsf{V}\times\mathsf{V} \arrow[d, "\varphi_\mathsf{V}\times\varphi_\mathsf{V}"] & \mathsf{V} \arrow[l, "\Delta"'] \arrow[d, "\varphi_\mathsf{V}"] \\ \mathsf{E}' \arrow[r, "\iota'"] & \mathsf{V}'\times\mathsf{V}' & \mathsf{V}' \arrow[l, "\Delta"'] \end{tikzcd}$$

$$\begin{tikzcd} \mathsf{E} \arrow[r, hook, "\iota"] \arrow[d, "\varphi_\mathsf{E}"] & \mathsf{V}\times\mathsf{V} \arrow[d, "\varphi_\mathsf{V}\times\varphi_\mathsf{V}"] & \mathsf{V} \arrow[l, "\Delta"'] \arrow[d, "\varphi_\mathsf{V}"] \\ \mathsf{E}' \arrow[r, hook, "\iota'"] & \mathsf{V}'\times\mathsf{V}' & \mathsf{V}' \arrow[l, "\Delta"'] \end{tikzcd}$$

$$\begin{tikzcd} \mathsf{E} \arrow[r, "\iota"] \arrow[d, "\varphi_\mathsf{E}"] & \mathcal{P}^{(1,2)}(\mathsf{V}) \arrow[d, "\mathcal{P}^{(1,2)}(\varphi_\mathsf{V})"] & \mathsf{V} \arrow[l, "p^{(1,2)}"'] \arrow[d, "\varphi_\mathsf{V}"] \\ \mathsf{E}' \arrow[r, "\iota'"] & \mathcal{P}^{(1,2)}(\mathsf{V}') & \mathsf{V}' \arrow[l, "p^{(1,2)}"'] \end{tikzcd}$$

| | multigraphs | simple graphs |
|---|---|---|
| **directed** | **comma category** over an **adhesive category** & functor **preserves all pullbacks** $\Rightarrow$ **adhesive category** | **Artin gluing** based upon **quasi-topos** & functor **preserves all pullbacks** $\Rightarrow$ **quasi-topos** (but **not adhesive!**) |
| | $\underline{\mathsf{Set}} / \Delta$ | $\underline{\mathsf{Set}} /\!\!/ \Delta$ |
| **undirected** | **comma category** over an **adhesive category** & functor **preserves only pullbacks along monos** $\Rightarrow$ $\mathscr{M}$**-adhesive category** | |
| | $\underline{\mathsf{Set}} / \mathcal{P}^{(1,2)}$ | $\underline{\mathsf{Set}} /\!\!/ \mathcal{P}^{(1,2)}$ |

$$\begin{tikzcd} \mathsf{E} \arrow[r, "\iota"] \arrow[d, "\varphi_{\mathsf{E}}"'] & \mathsf{V}\times\mathsf{V} \arrow[d, "\Delta(\varphi_{\mathsf{V}})"] & \mathsf{V} \arrow[l, "\Delta"'] \arrow[d, "\varphi_{\mathsf{V}}"] \\ \mathsf{E}' \arrow[r, "\iota'"] & \mathsf{V}'\times\mathsf{V}' & \mathsf{V}' \arrow[l, "\Delta"'] \end{tikzcd}$$

$$\begin{tikzcd} \mathsf{E} \arrow[r, hook, "\iota"] \arrow[d, "\varphi_{\mathsf{E}}"'] & \mathsf{V}\times\mathsf{V} \arrow[d, "\Delta(\varphi_{\mathsf{V}})"] & \mathsf{V} \arrow[l, "\Delta"'] \arrow[d, "\varphi_{\mathsf{V}}"] \\ \mathsf{E}' \arrow[r, hook, "\iota'"] & \mathsf{V}'\times\mathsf{V}' & \mathsf{V}' \arrow[l, "\Delta"'] \end{tikzcd}$$

$$\begin{tikzcd} \mathsf{E} \arrow[r, "\iota"] \arrow[d, "\varphi_{\mathsf{E}}"'] & \mathcal{P}^{(1,2)}(\mathsf{V}) \arrow[d, "\mathcal{P}^{(1,2)}(\varphi_{\mathsf{V}})"] & \mathsf{V} \arrow[l, "\mathcal{P}^{(1,2)}"'] \arrow[d, "\varphi_{\mathsf{V}}"] \\ \mathsf{E}' \arrow[r, "\iota'"] & \mathcal{P}^{(1,2)}(\mathsf{V}') & \mathsf{V}' \arrow[l, "\mathcal{P}^{(1,2)}"'] \end{tikzcd}$$

$$\begin{tikzcd} \mathsf{E} \arrow[r, hook, "\iota"] \arrow[d, "\varphi_{\mathsf{E}}"'] & \mathcal{P}^{(1,2)}(\mathsf{V}) \arrow[d, "\mathcal{P}^{(1,2)}(\varphi_{\mathsf{V}})"] & \mathsf{V} \arrow[l, "\mathcal{P}^{(1,2)}"'] \arrow[d, "\varphi_{\mathsf{V}}"] \\ \mathsf{E}' \arrow[r, hook, "\iota'"] & \mathcal{P}^{(1,2)}(\mathsf{V}') & \mathsf{V}' \arrow[l, "\mathcal{P}^{(1,2)}"'] \end{tikzcd}$$

| | **multigraphs** | **simple graphs** |
|---|---|---|
| **directed** | **comma category** over an **adhesive category** & functor **preserves all pullbacks** ⇒ **adhesive category** | **Artin gluing** based upon **quasi-topos** & functor **preserves all pullbacks** ⇒ **quasi-topos** (but **not adhesive!**) |
| | $\underline{\mathsf{Set}}\,/\Delta$ | $\underline{\mathsf{Set}}\,/\!\!/\Delta$ |
| **undirected** | **comma category** over an **adhesive category** & functor **preserves only pullbacks along monos** ⇒ $\mathscr{M}$**-adhesive category** | **?** |
| | $\underline{\mathsf{Set}}\,/\mathcal{P}^{(1,2)}$ | $\underline{\mathsf{Set}}\,/\!\!/\mathcal{P}^{(1,2)}$ |

$$\begin{tikzcd} \mathsf{E} \arrow[r, "\iota"] \arrow[d, "\varphi_{\mathsf{E}}"'] & \mathsf{V}\times\mathsf{V} \arrow[d] & \mathsf{V} \arrow[l, "\Delta"'] \arrow[d, "\varphi_{\mathsf{V}}"] \\ \mathsf{E}' \arrow[r, "\iota'"] & \mathsf{V}'\times\mathsf{V}' & \mathsf{V}' \arrow[l, "\Delta"'] \end{tikzcd} \qquad \begin{tikzcd} \mathsf{E} \arrow[r, hook, "\iota"] \arrow[d, "\varphi_{\mathsf{E}}"'] & \mathsf{V}\times\mathsf{V} \arrow[d] & \mathsf{V} \arrow[l, "\Delta"'] \arrow[d, "\varphi_{\mathsf{V}}"] \\ \mathsf{E}' \arrow[r, hook, "\iota'"] & \mathsf{V}'\times\mathsf{V}' & \mathsf{V}' \arrow[l, "\Delta"'] \end{tikzcd}$$

$$\begin{tikzcd} \mathsf{E} \arrow[r, "\iota"] \arrow[d, "\varphi_{\mathsf{E}}"'] & \mathcal{P}^{(1,2)}(\mathsf{V}) \arrow[d, "\mathcal{P}^{(1,2)}(\varphi_{\mathsf{V}})"] & \mathsf{V} \arrow[l, "\mathcal{P}^{(1,2)}"'] \arrow[d, "\varphi_{\mathsf{V}}"] \\ \mathsf{E}' \arrow[r, "\iota'"] & \mathcal{P}^{(1,2)}(\mathsf{V}') & \mathsf{V}' \arrow[l, "\mathcal{P}^{(1,2)}"'] \end{tikzcd} \qquad \begin{tikzcd} \mathsf{E} \arrow[r, hook, "\iota"] \arrow[d, "\varphi_{\mathsf{E}}"'] & \mathcal{P}^{(1,2)}(\mathsf{V}) \arrow[d, "\mathcal{P}^{(1,2)}(\varphi_{\mathsf{V}})"] & \mathsf{V} \arrow[l, "\mathcal{P}^{(1,2)}"'] \arrow[d, "\varphi_{\mathsf{V}}"] \\ \mathsf{E}' \arrow[r, hook, "\iota'"] & \mathcal{P}^{(1,2)}(\mathsf{V}') & \mathsf{V}' \arrow[l, "\mathcal{P}^{(1,2)}"'] \end{tikzcd}$$

# **Quasi-topoi** — a natural setting for non-linear rewriting

## Definition

A category **C** is a quasi-topos iff

1. it has finite limits and colimits
2. it is locally Cartesian closed
3. it has a regular-subobject-classifier.

Johnstone, P.T., Lack, S., Sobociński, P.: Quasitoposes, Quasiadhesive Categories and Artin Glueing. In: Algebra and Coalgebra in Computer Science. LNCS, vol. 4624, pp. 312–326 (2007). https://doi.org/10.1007/978-3-540-73859-6_21

# **Quasi-topoi** — a natural setting for non-linear rewriting

## Proposition

Every quasi-topos $\mathbf{C}$ enjoys the following properties:

- It has (by definition) a stable system of monics $\mathcal{M} = \mathsf{rm}(\mathbf{C})$ (the class of regular monos), which coincides with the class of extremal monomorphisms, i.e., if $\mathsf{m} = \mathsf{f} \circ \mathsf{e}$ for $\mathsf{m} \in \mathsf{rm}(\mathbf{C})$ and $\mathsf{e} \in \mathsf{epi}(\mathbf{C})$, then $\mathsf{e} \in \mathsf{iso}(\mathbf{C})$.
- It has (by definition) a $\mathcal{M}$-partial map classifier $(\mathsf{T}, \eta)$.
- It is rm-quasi-adhesive, i.e., it has pushouts along regular monomorphisms, these are stable under pullbacks, and pushouts along regular monos are pullbacks.
- It is $\mathcal{M}$-adhesive.
- For all pairs of composable morphisms $\mathsf{A} \xrightarrow{\mathsf{f}} \mathsf{B}$ and $\mathsf{B} \xrightarrow{\mathsf{m}} \mathsf{C}$ with $\mathsf{m} \in \mathcal{M}$, there exists a final pullback-complement (FPC) $\mathsf{A} \xrightarrow{\mathsf{n}} \mathsf{F} \xrightarrow{\mathsf{g}} \mathsf{C}$, and with $\mathsf{n} \in \mathcal{M}$.
- It possesses an epi-$\mathcal{M}$-factorization: each morphism $\mathsf{A} \xrightarrow{\mathsf{f}} \mathsf{B}$ factors as $\mathsf{f} = \mathsf{m} \circ \mathsf{e}$, with morphisms $\mathsf{A} \xrightarrow{\mathsf{e}} \mathsf{B}$ in $\mathsf{epi}(\mathbf{C})$ and $\mathsf{B} \xrightarrow{\mathsf{m}} \mathsf{A}$ in $\mathcal{M}$ (uniquely up to isomorphism in $\mathsf{B}$).

# Case of **directed simple graphs** (a **quasi-topos!**)

**rm-adhesive categories**

**$\mathscr{M}$-adhesive categories**
(+ some extra assumptions…)

$$\begin{tikzcd}
\mathsf{O} & \mathsf{K} \arrow[l, tail, "\mathsf{o}"'] \arrow[r, tail, "\mathsf{i}"] \arrow[d, tail, "\overline{\mathsf{m}}"] \arrow[dl, phantom, "\textbf{PO}"] \arrow[dr, phantom, "\textbf{POC}"] & \mathsf{I} \arrow[d, tail, "\mathsf{m}"] \\
\mathsf{r_m(X)} \arrow[from=1-1, tail, "\mathsf{m}^*"'] & \overline{\mathsf{X}} \arrow[l, tail, "\overline{\mathsf{o}}"'] \arrow[r, tail, "\overline{\mathsf{i}}"] & \mathsf{X}
\end{tikzcd}$$

($\mathscr{M}$-) **linear Double-Pushout (DPO)**

$$\begin{tikzcd}
O & K \arrow[l, "o"'] \arrow[r, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dl, phantom, "\textbf{PO}"] \arrow[dr, phantom, "\textbf{mPOC}"] & I \arrow[d, tail, "m"] \\
r_m(X) \arrow[from=1-1, tail, "m^*"'] & \bar{X} \arrow[l, "\bar{o}"'] \arrow[r, "\bar{i}"] & X
\end{tikzcd}$$

**non-linear Double-Pushout (DPO)**

$$\begin{tikzcd}
\mathsf{O} & \mathsf{K} \arrow[l, tail, "\mathsf{o}"'] \arrow[r, tail, "\mathsf{i}"] \arrow[d, tail, "\overline{\mathsf{m}}"] \arrow[dl, phantom, "\textbf{PO}"] \arrow[dr, phantom, "\textbf{FPC}"] & \mathsf{I} \arrow[d, tail, "\mathsf{m}"] \\
\mathsf{r_m(X)} \arrow[from=1-1, tail, "\mathsf{m}^*"'] & \overline{\mathsf{X}} \arrow[l, tail, "\overline{\mathsf{o}}"'] \arrow[r, tail, "\overline{\mathsf{i}}"] & \mathsf{X}
\end{tikzcd}$$

($\mathscr{M}$-) **linear Sesqui-Pushout (SqPO)**

$$\begin{tikzcd}
O & K \arrow[l, "o"'] \arrow[r, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dl, phantom, "\textbf{PO}"] \arrow[dr, phantom, "\textbf{FPC}"] & I \arrow[d, tail, "m"] \\
r_m(X) \arrow[from=1-1, tail, "m^*"'] & \bar{X} \arrow[l, "\bar{o}"'] \arrow[r, "\bar{i}"] & X
\end{tikzcd}$$

**non-linear Sesqui-Pushout (SqPO)**

**quasi-topoi**

# Case of **directed simple graphs** (a **quasi-topos!**)

**$\mathscr{M}$-adhesive categories**
(+ some extra assumptions…)

$$\begin{tikzcd}
O \arrow[d, tail, "m^*"'] \arrow[dr, phantom, "\mathrm{PO}"] & K \arrow[l, tail, "o"'] \arrow[r, tail, "i"] \arrow[d, tail, "\overline{m}"] \arrow[dr, phantom, "\mathrm{POC}"] & I \arrow[d, tail, "m"] \\
r_m(X) & \overline{X} \arrow[l, tail, "\overline{o}"] \arrow[r, tail, "\overline{i}"'] & X
\end{tikzcd}$$

($\mathscr{M}$-) **linear Double-Pushout (DPO)**

$$\begin{tikzcd}
O \arrow[d, tail, "m^*"'] \arrow[dr, phantom, "\mathrm{PO}"] & K \arrow[l, tail, "o"'] \arrow[r, tail, "i"] \arrow[d, tail, "\overline{m}"] \arrow[dr, phantom, "\mathrm{FPC}"] & I \arrow[d, tail, "m"] \\
r_m(X) & \overline{X} \arrow[l, tail, "\overline{o}"] \arrow[r, tail, "\overline{i}"'] & X
\end{tikzcd}$$

($\mathscr{M}$-) **linear Sesqui-Pushout (SqPO)**

**rm-adhesive categories**

$$\begin{tikzcd}
O \arrow[d, tail, "m^*"'] \arrow[dr, phantom, "\mathrm{PO}"] & K \arrow[l, "o"'] \arrow[r, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dr, phantom, "\mathrm{mPOC}"] & I \arrow[d, tail, "m"] \\
r_m(X) & \bar{X} \arrow[l, "\bar{o}"] \arrow[r, "\bar{i}"'] & X
\end{tikzcd}$$

**non-linear Double-Pushout (DPO)**

$$\begin{tikzcd}
O \arrow[d, tail, "m^*"'] \arrow[dr, phantom, "\mathrm{PO}"] & K \arrow[l, "o"'] \arrow[r, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dr, phantom, "\mathrm{FPC}"] & I \arrow[d, tail, "m"] \\
r_m(X) & \bar{X} \arrow[l, "\bar{o}"] \arrow[r, "\bar{i}"'] & X
\end{tikzcd}$$

**non-linear Sesqui-Pushout (SqPO)**

**quasi-topoi**

# Case of **directed simple graphs** (a **quasi-topos!**)

**rm-adhesive categories**

**$\mathscr{M}$-adhesive categories**
(+ some extra assumptions…)

$$
\begin{tikzcd}
O \arrow[d, tail, "m^*"'] \arrow[dr, phantom, "\mathbf{PO}"] & K \arrow[l, tail, "o"'] \arrow[r, tail, "i"] \arrow[d, tail, "\overline{m}"] \arrow[dr, phantom, "\mathbf{POC}"] & I \arrow[d, tail, "m"] \\
r_m(X) & \overline{X} \arrow[l, tail, "\overline{o}"] \arrow[r, tail, "\overline{i}"'] & X
\end{tikzcd}
$$

($\mathscr{M}$-) **linear Double-Pushout (DPO)**

$$
\begin{tikzcd}
O \arrow[d, tail, "m^*"'] \arrow[dr, phantom, "\mathbf{PO}"] & K \arrow[l, "o"'] \arrow[r, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dr, phantom, "\mathbf{mPOC}"] & I \arrow[d, tail, "m"] \\
r_m(X) & \bar{X} \arrow[l, "\bar{o}"] \arrow[r, "\bar{i}"'] & X
\end{tikzcd}
$$

**non-linear Double-Pushout (DPO)**

$$
\begin{tikzcd}
O \arrow[d, tail, "m^*"'] \arrow[dr, phantom, "\mathbf{PO}"] & K \arrow[l, tail, "o"'] \arrow[r, tail, "i"] \arrow[d, tail, "\overline{m}"] \arrow[dr, phantom, "\mathbf{FPC}"] & I \arrow[d, tail, "m"] \\
r_m(X) & \overline{X} \arrow[l, tail, "\overline{o}"] \arrow[r, tail, "\overline{i}"'] & X
\end{tikzcd}
$$

($\mathscr{M}$-) **linear Sesqui-Pushout (SqPO)**

$$
\begin{tikzcd}
O \arrow[d, tail, "m^*"'] \arrow[dr, phantom, "\mathbf{PO}"] & K \arrow[l, "o"'] \arrow[r, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dr, phantom, "\mathbf{FPC}"] & I \arrow[d, tail, "m"] \\
r_m(X) & \bar{X} \arrow[l, "\bar{o}"] \arrow[r, "\bar{i}"'] & X
\end{tikzcd}
$$

**non-linear Sesqui-Pushout (SqPO)**

**quasi-topoi**

# Case of **directed simple graphs** (a **quasi-topos!**)

**$\mathscr{M}$-adhesive categories**
(+ some extra assumptions…)

$$\begin{tikzcd} O \arrow[d, tail, "m^*"'] & K \arrow[l, tail, "o"'] \arrow[r, tail, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dl, phantom, "\mathrm{PO}"] & I \arrow[d, tail, "m"] \arrow[dl, phantom, "\mathrm{POC}"] \\ r_m(X) & \bar{X} \arrow[l, tail, "\bar{o}"] \arrow[r, tail, "\bar{i}"'] & X \end{tikzcd}$$

($\mathscr{M}$-) **linear Double-Pushout (DPO)**

$$\begin{tikzcd} O \arrow[d, tail, "m^*"'] & K \arrow[l, tail, "o"'] \arrow[r, tail, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dl, phantom, "\mathrm{PO}"] & I \arrow[d, tail, "m"] \arrow[dl, phantom, "\mathrm{FPC}"] \\ r_m(X) & \bar{X} \arrow[l, tail, "\bar{o}"] \arrow[r, tail, "\bar{i}"'] & X \end{tikzcd}$$

($\mathscr{M}$-) **linear Sesqui-Pushout (SqPO)**

**rm-adhesive categories**

$$\begin{tikzcd} O \arrow[d, tail, "m^*"'] & K \arrow[l, "o"'] \arrow[r, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dl, phantom, "\mathrm{PO}"] & I \arrow[d, tail, "m"] \arrow[dl, phantom, "\mathrm{mPOC}"] \\ r_m(X) & \bar{X} \arrow[l, "\bar{o}"] \arrow[r, "\bar{i}"'] & X \end{tikzcd}$$

**non-linear Double-Pushout (DPO)**

$$\begin{tikzcd} O \arrow[d, tail, "m^*"'] & K \arrow[l, "o"'] \arrow[r, "i"] \arrow[d, tail, "\bar{m}"] \arrow[dl, phantom, "\mathrm{PO}"] & I \arrow[d, tail, "m"] \arrow[dl, phantom, "\mathrm{FPC}"] \\ r_m(X) & \bar{X} \arrow[l, "\bar{o}"] \arrow[r, "\bar{i}"'] & X \end{tikzcd}$$

**non-linear Sesqui-Pushout (SqPO)**

**quasi-topoi**

# **Rule algebras** for categorical rewriting systems

$$\begin{tikzcd}
 & O_1 \arrow[dl, tail] & I_1 \arrow[l, "r_1"'] \arrow[d, tail] \arrow[dl, phantom, "\mathbb{T}"] \\
X_1 & & X_0 \arrow[ll]
\end{tikzcd}$$

# **Rule algebras** for categorical rewriting systems

$$\begin{tikzcd}
O_2 \arrow[d, tail] \arrow[dr, phantom, "\mathbb{T}"] & I_2 \arrow[l, hook', "r_2"'] \arrow[dr, tail] & & O_1 \arrow[dl, tail] \arrow[dr, phantom, "\mathbb{T}"] & I_1 \arrow[l, hook', "r_1"'] \arrow[d, tail] \\
X_2 & & X_1 \arrow[ll, hook'] & & X_0 \arrow[ll, hook']
\end{tikzcd}$$

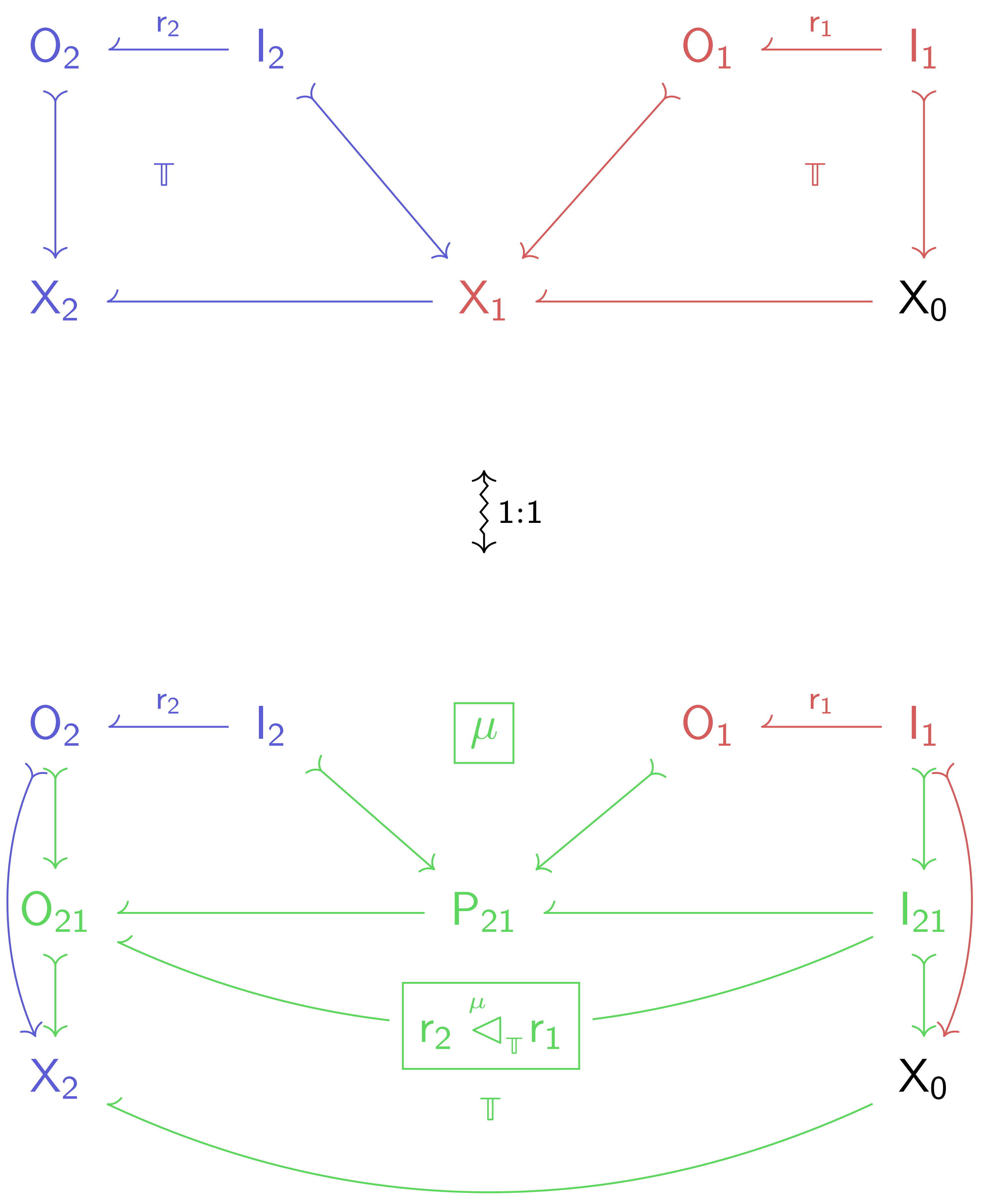
# **Rule algebras** for categorical rewriting systems

$$
\begin{tikzcd}
O_2 \arrow[d, tail] & I_2 \arrow[l, hook', "r_2"'] \arrow[dr, tail] & & O_1 \arrow[dl, tail] & I_1 \arrow[l, hook', "r_1"'] \arrow[d, tail] \\
X_2 \arrow[ur, phantom, "\mathbb{T}"] & & X_1 \arrow[ll, hook'] & & X_0 \arrow[ll, hook'] \arrow[ul, phantom, "\mathbb{T}"]
\end{tikzcd}
$$

$$
\Big\updownarrow 1{:}1
$$

$$
\begin{tikzcd}
O_2 \arrow[d, tail] \arrow[dd, bend right, tail] & I_2 \arrow[l, hook', "r_2"'] \arrow[dr, tail] & \boxed{\mu} & O_1 \arrow[dl, tail] & I_1 \arrow[l, hook', "r_1"'] \arrow[d, tail] \arrow[dd, bend left, tail] \\
O_{21} \arrow[d, tail] & & P_{21} \arrow[ll, hook'] & & I_{21} \arrow[ll, hook'] \arrow[d, tail] \arrow[llll, bend left, hook', "\boxed{r_2 \overset{\mu}{\triangleleft}_{\mathbb{T}} r_1}"] \\
X_2 & & & & X_0 \arrow[llll, bend left, hook', "\mathbb{T}"]
\end{tikzcd}
$$

# **Rule algebras** for categorical rewriting systems




$$\left(O \xleftarrow{r} I\right) \rightsquigarrow \delta\left(O \xleftarrow{r} I\right)$$

(iso-class of a) **rule**

**basis vector** of a **vector space** $\mathcal{R}$

**Definition:** the **rule algebra product** $*_{\mathcal{R}} : \mathcal{R} \times \mathcal{R} \to \mathcal{R}$ is defined via

$$\delta(r_2) *_{\mathcal{R}} \delta(r_1) := \sum_{\mu} \delta\left(r_2 \overset{\mu}{\triangleleft}_{\mathbb{T}} r_1\right)$$

"sum over ways to compose the rules"

**Theorem** LiCS 2016, CSL 2018, GCM 2019, LMCS 2020, ICGT 2020

The **rule algebra** $(\mathcal{R}, *_{\mathcal{R}})$ is an **associative unital algebra**, with **unit element** $\delta(\varnothing \leftharpoonup \varnothing)$.

# **Rule algebras** for categorical rewriting systems

$$X \rightsquigarrow |X\rangle$$

(iso-class of an) **object**

**basis vector** of a **vector space** $\hat{\mathbf{C}}$

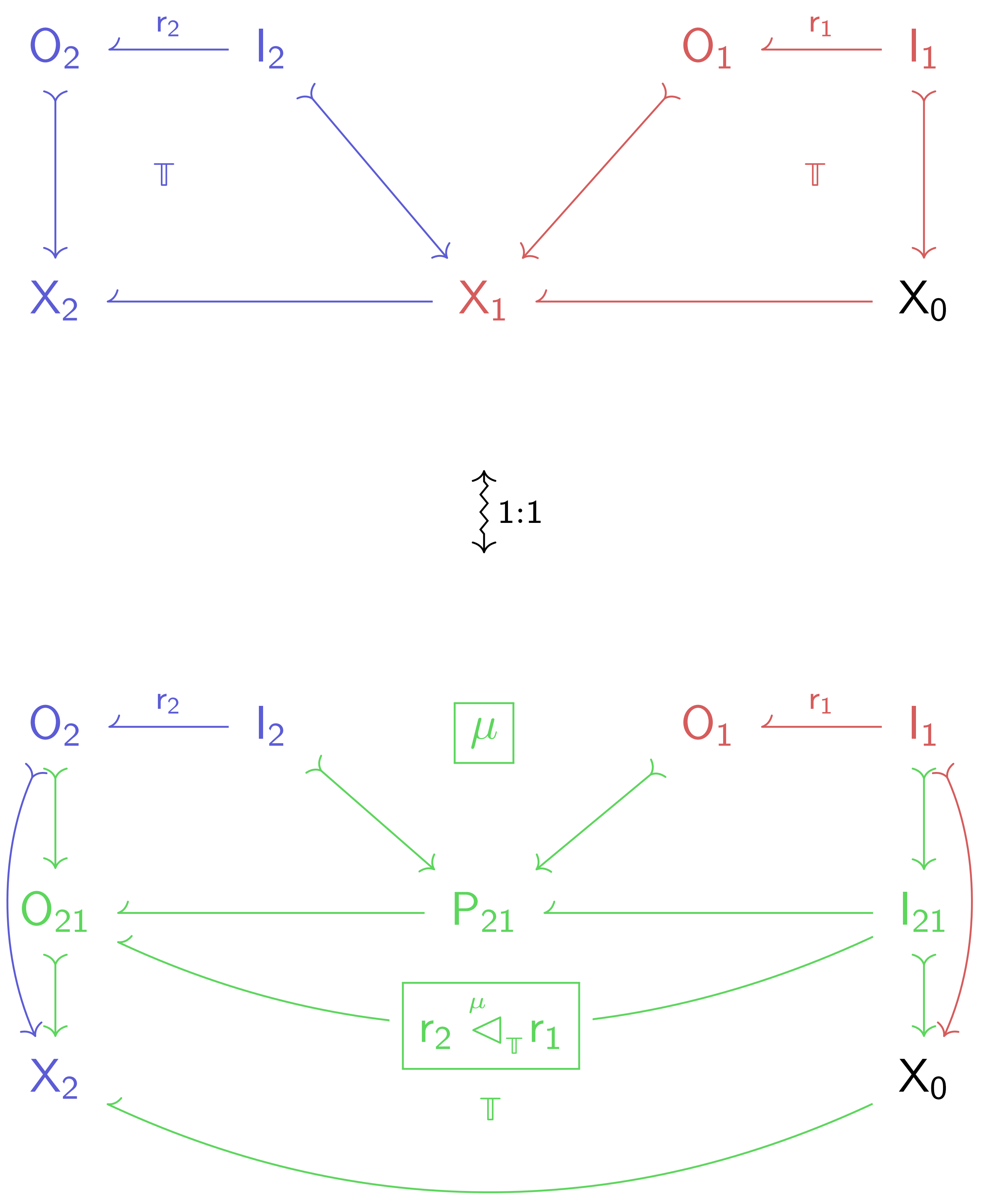
$$\begin{tikzcd} O \arrow[d, tail] & I \arrow[l, "r"'] \arrow[d, tail, "m"] \arrow[dl, phantom, "\mathbb{T}"] \\ r_m(X) & X \arrow[l, "r_m"] \end{tikzcd}$$

$$\rho_{\mathbb{T}} : \mathcal{R} \to \mathrm{End}(\hat{\mathbf{C}})$$

$$\rho_{\mathbb{T}}(\delta(r))\,|X\rangle := \sum_{m} |r_m(X)\rangle$$

"sum over **all ways** to apply **r** to **X**"

# **Rule algebras** for categorical rewriting systems




$$\delta(\mathsf{r}_2) *_{\mathcal{R}} \delta(\mathsf{r}_1) := \sum_{\mu} \delta\left(\mathsf{r}_2 \overset{\mu}{\triangleleft}_{\mathbb{T}} \mathsf{r}_1\right)$$

$$\rho_{\mathbb{T}} : \mathcal{R} \to \mathrm{End}\left(\hat{\mathbf{C}}\right)$$

$$\rho_{\mathbb{T}}\left(\delta(\mathsf{r})\right) |X\rangle := \sum_{m} |\mathsf{r}_m(X)\rangle$$

**Theorem** LiCS 2016, CSL 2018, GCM 2019, LMCS 2020, ICGT 2020

$\rho_{\mathbb{T}} : \mathcal{R} \to \mathrm{End}\left(\hat{\mathbf{C}}\right)$ is a **representation** of the **rule algebra** $(\mathcal{R}, *_{\mathcal{R}})$, i.e.

$$\rho_{\mathbb{T}}\left(\delta(\mathsf{r}_2)\right)\rho_{\mathbb{T}}\left(\delta(\mathsf{r}_1)\right)|X\rangle = \rho_{\mathbb{T}}\left(\delta(\mathsf{r}_2) *_{\mathcal{R}} \delta(\mathsf{r}_1)\right)|X\rangle$$

# On Stochastic Rewriting and Combinatorics via Rule-Algebraic Methods*

Nicolas Behr

Université de Paris, CNRS, IRIF
F-75006, Paris, France

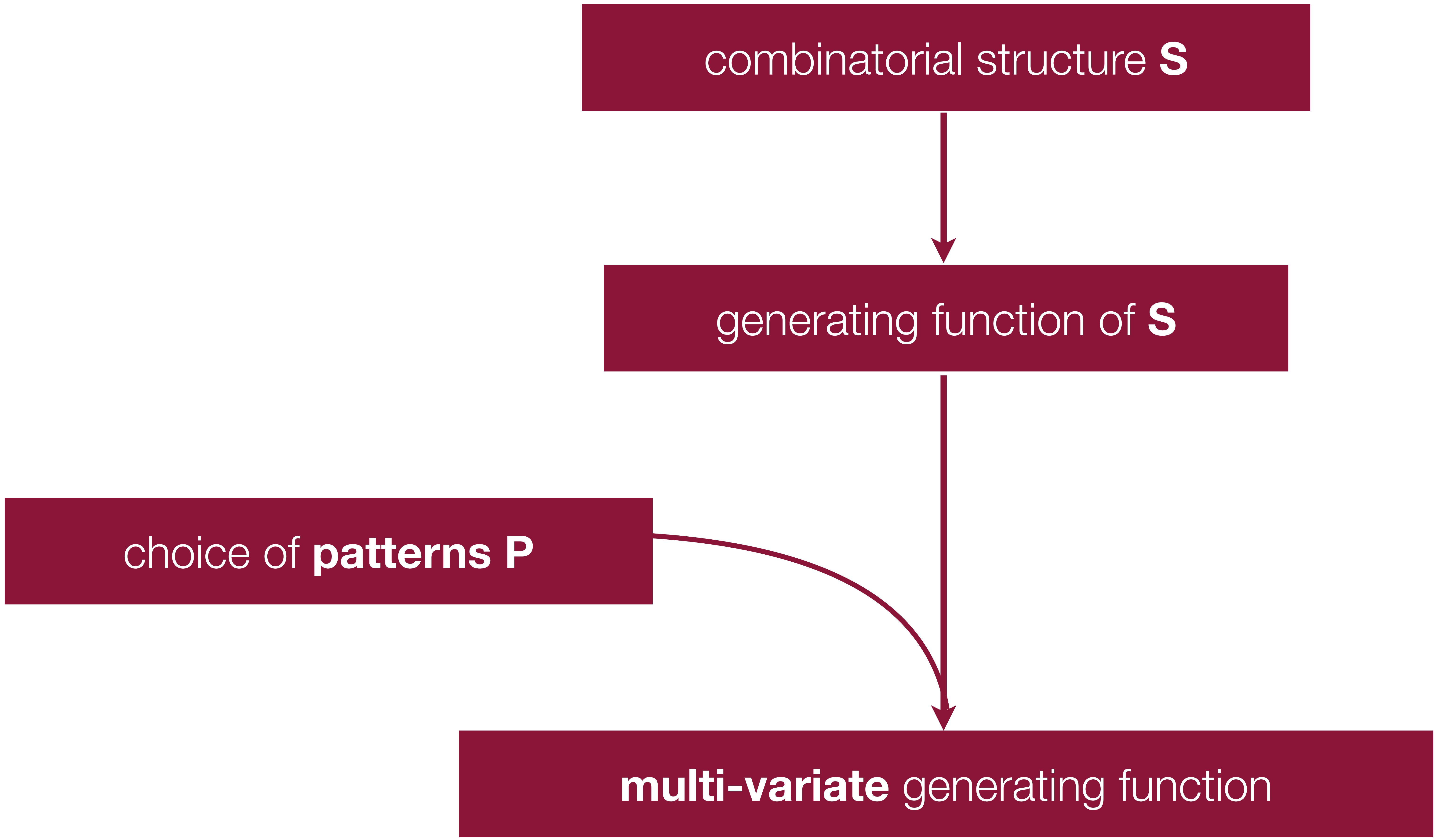
nicolas.behr@irif.fr

Building upon the rule-algebraic stochastic mechanics framework, we present new results on the relationship of stochastic rewriting systems described in terms of continuous-time Markov chains, their embedded discrete-time Markov chains and certain types of generating function expressions in combinatorics. We introduce a number of generating function techniques that permit a novel form of static analysis for rewriting systems based upon marginalizing distributions over the states of the rewriting systems via pattern-counting observables.

TERMGRAPH 2020

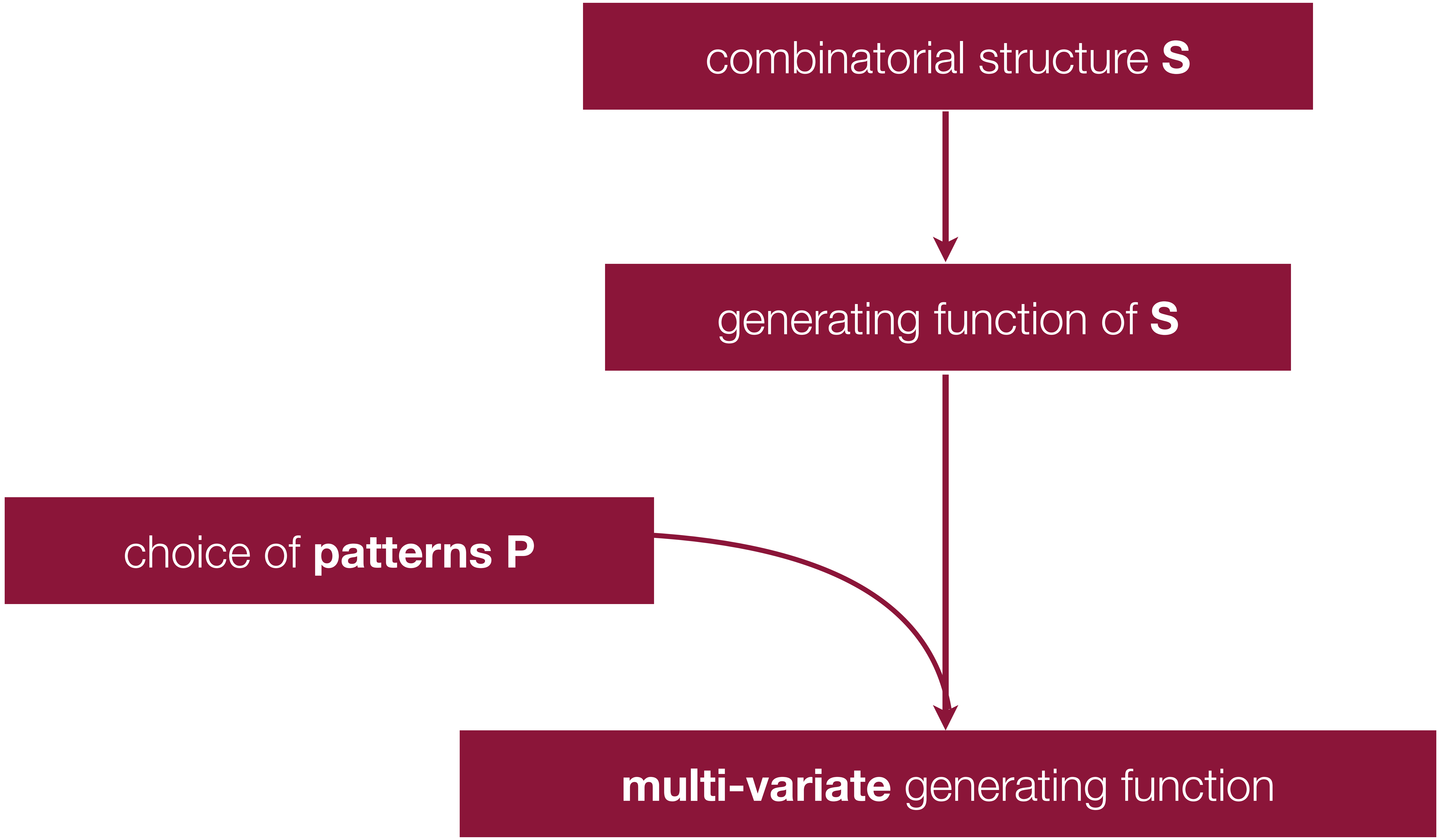
# Motivation

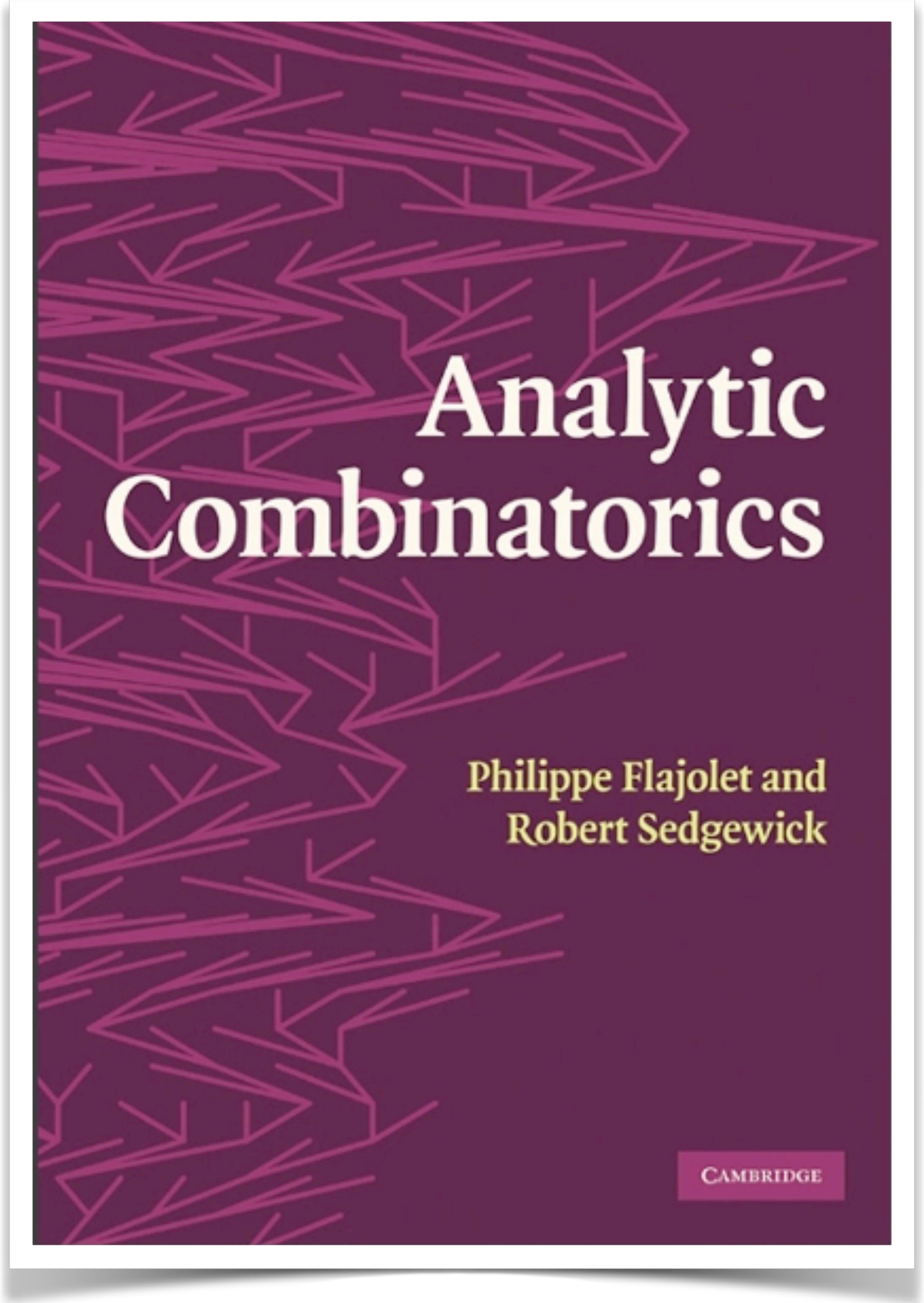
The enumerative combinatorics "workflow" (à la Flajolet):



# Motivation

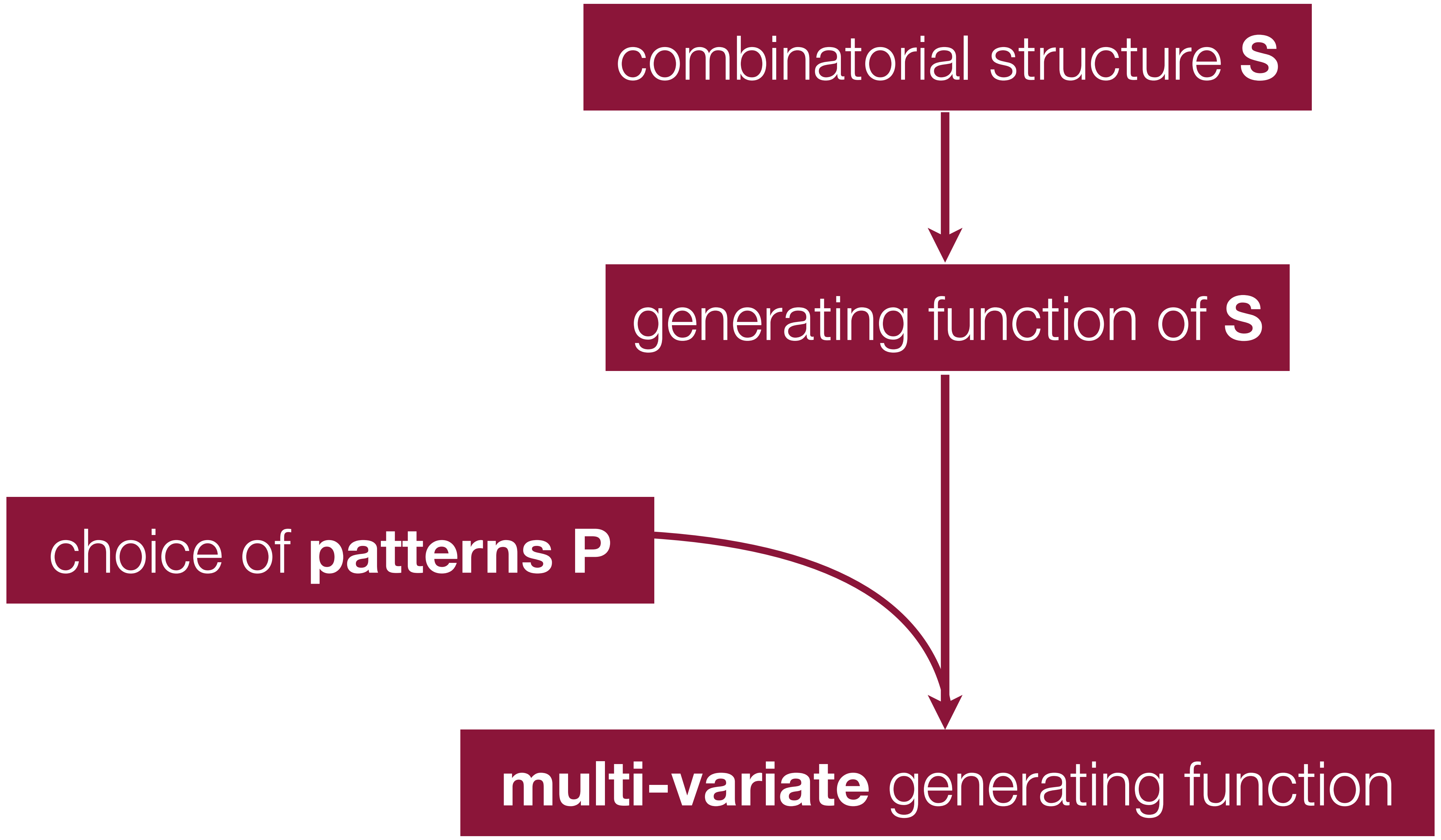
The enumerative combinatorics "workflow" (à la Flajolet):



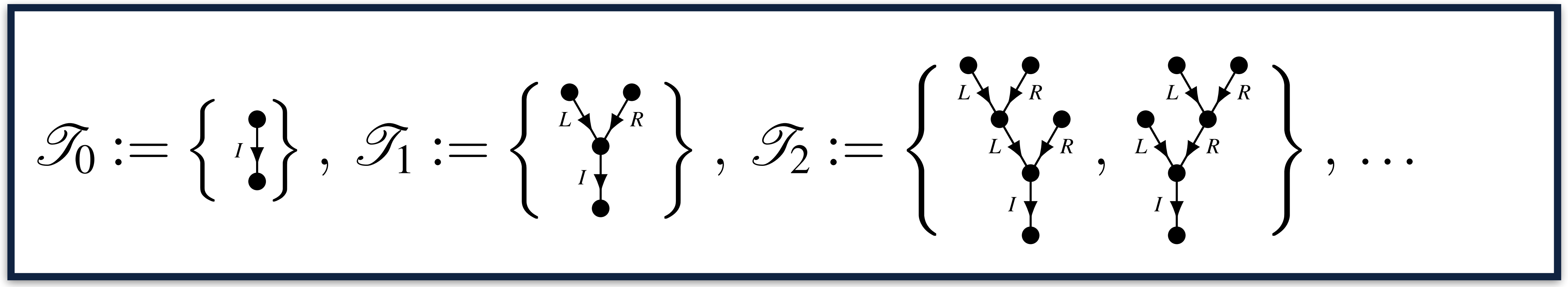
# Motivation

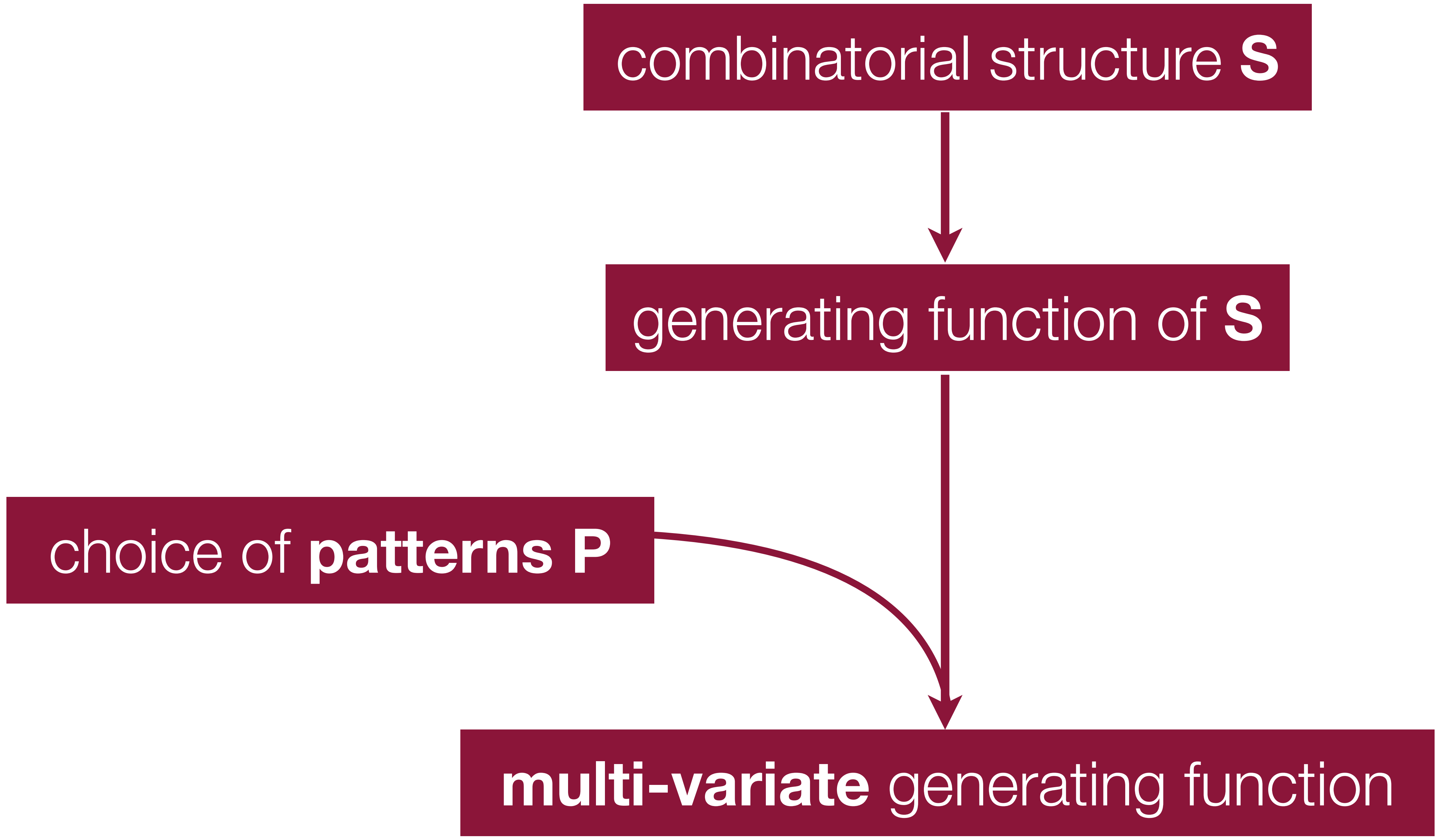
**Example:** planar rooted binary trees (PRBTs)



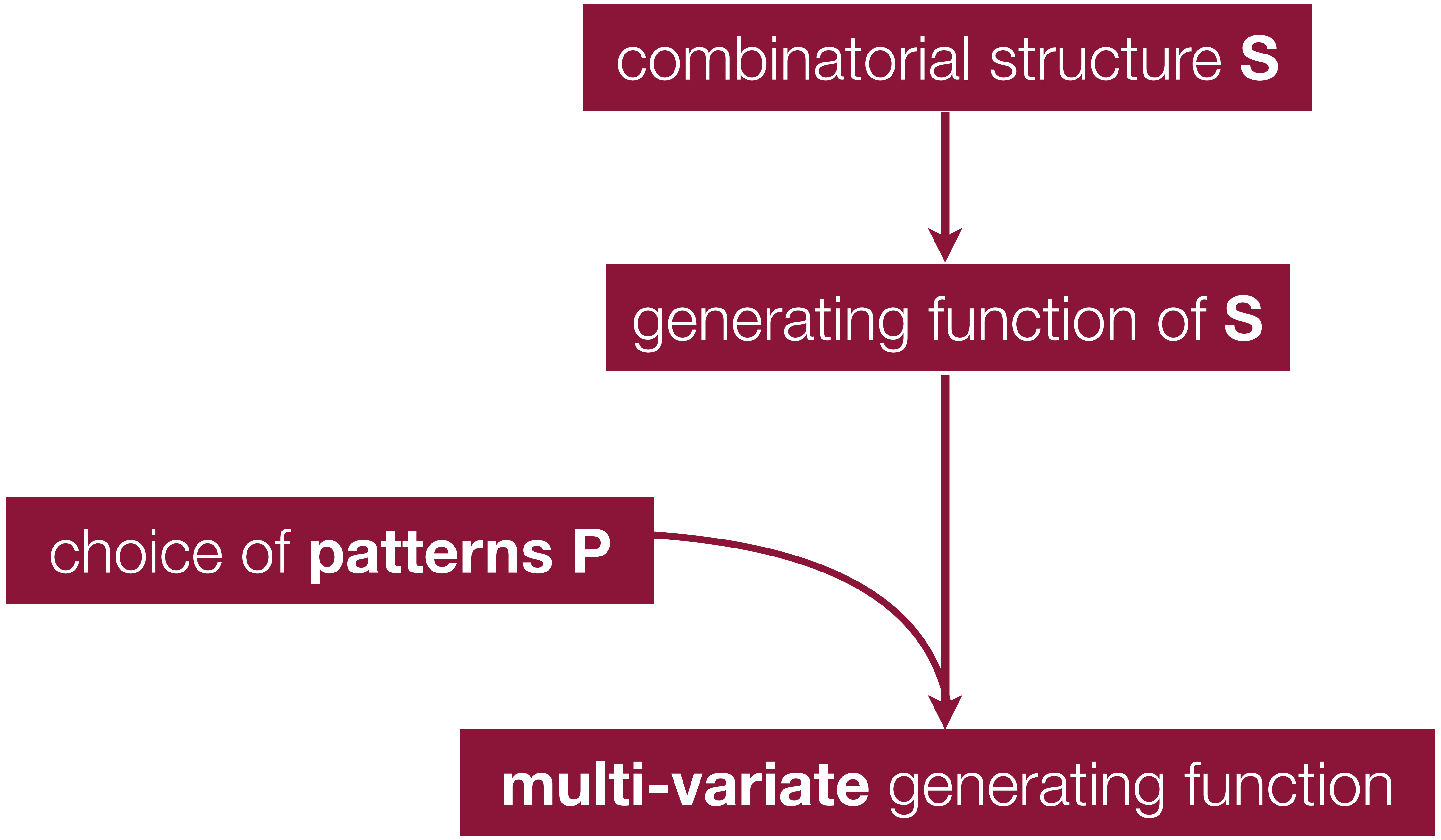
# Motivation

**Example:** planar rooted binary trees (PRBTs)

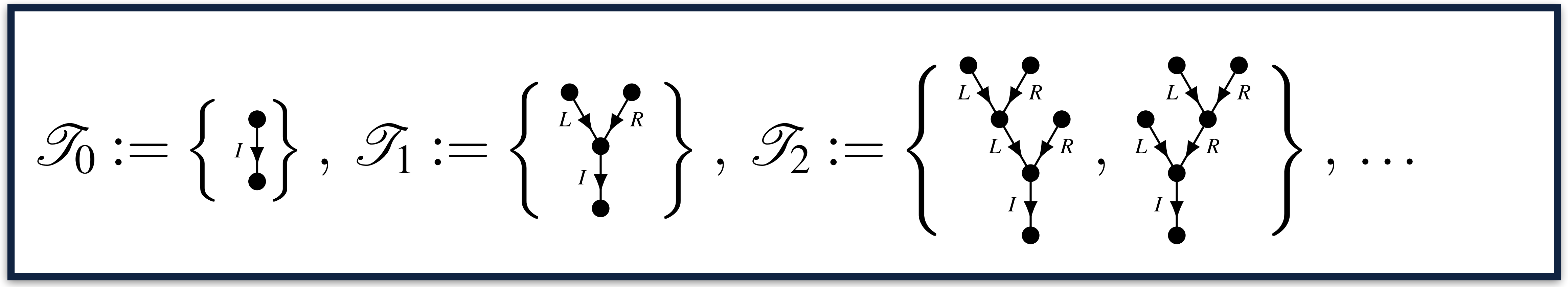


# Motivation




**Example:** planar rooted binary trees (PRBTs)




$$\mathcal{G}(\lambda) := \sum_{n\geq 0} \frac{\lambda^n}{n!}(\#\text{ of structures of size } n)$$

# Motivation

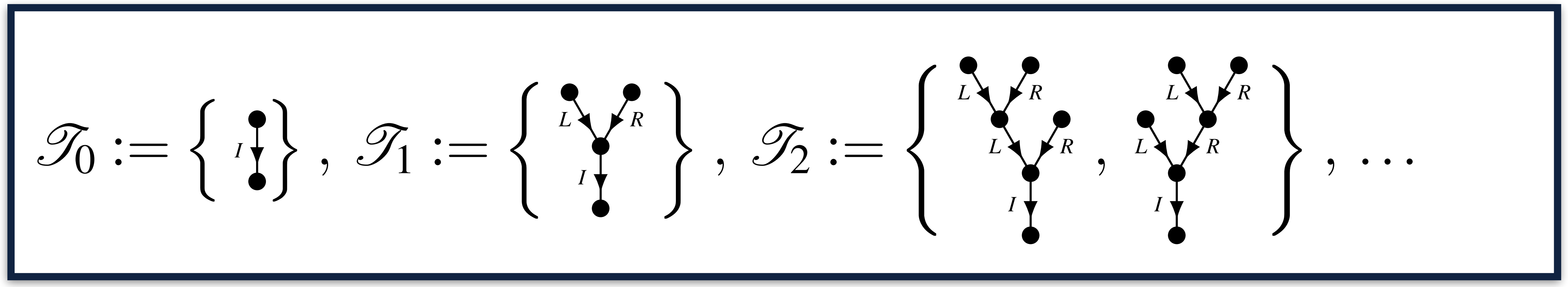
combinatorial structure **S**

generating function of **S**

choice of **patterns P**

**multi-variate** generating function

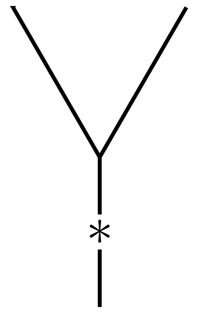
## **Example:** planar rooted binary trees (PRBTs)

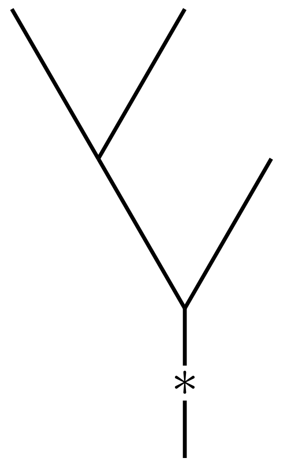


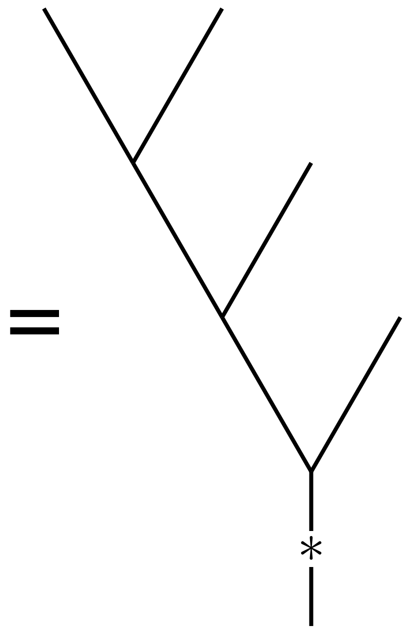

$$\mathcal{G}(\lambda) := \sum_{n\geq 0} \frac{\lambda^n}{n!}(\#\text{ of structures of size } n)$$

Choose some **patterns**:

$P_1 :=$ 

$P_2 :=$ 

$P_3 :=$ 

$P_4 :=$ 

# Motivation

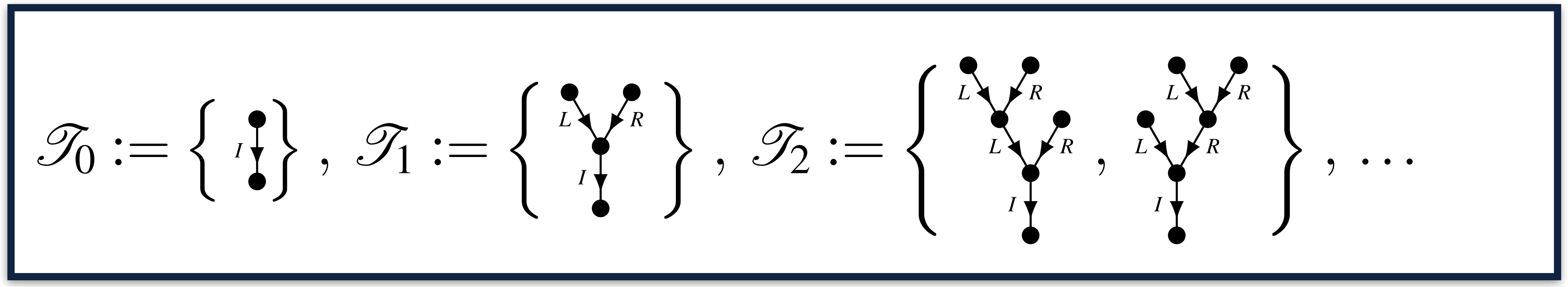
combinatorial structure **S**

generating function of **S**

choice of **patterns P**

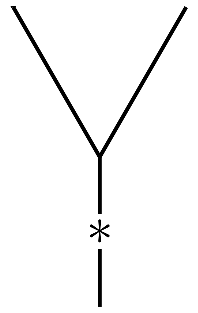
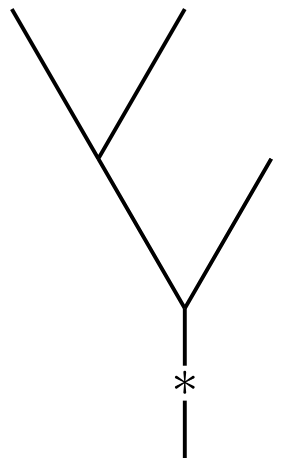
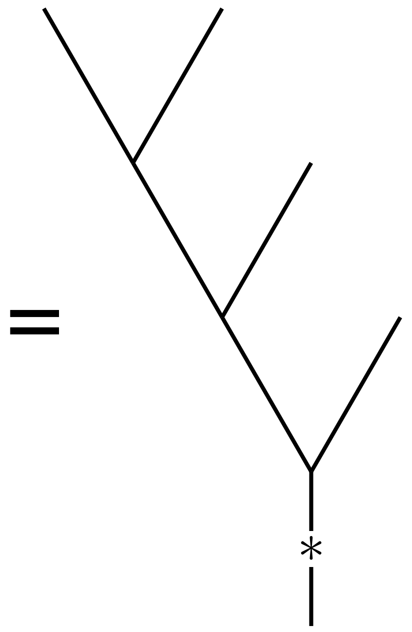
**multi-variate** generating function

**Example:** planar rooted binary trees (PRBTs)




$$\mathcal{G}(\lambda) := \sum_{n\geq 0} \frac{\lambda^n}{n!} (\#\text{ of structures of size } n)$$

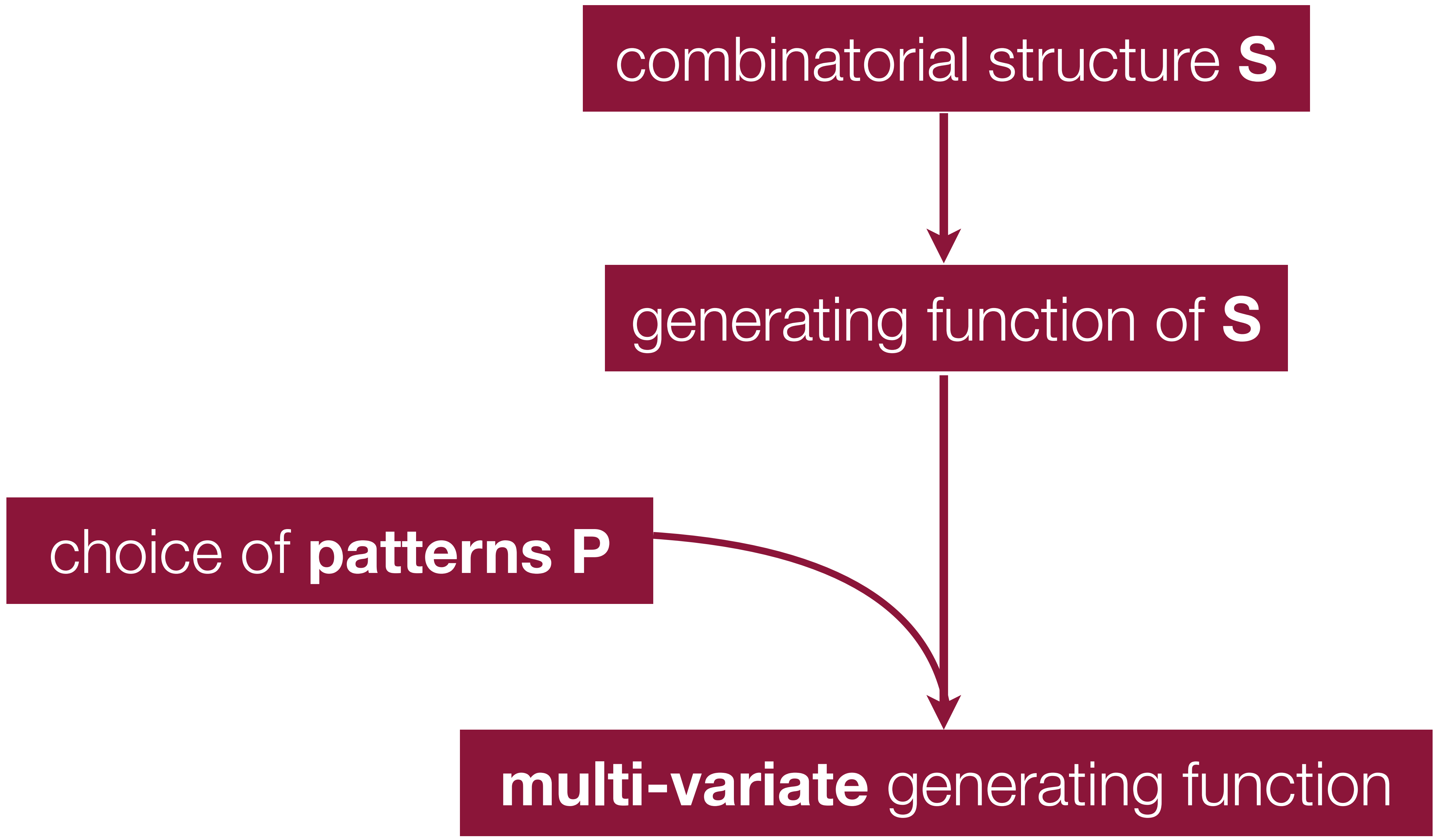
Choose some **patterns**:

$P_1 :=$  $P_2 :=$  $P_3 :=$  $P_4 :=$ 

$$\mathcal{G}(\lambda; \omega_1, \ldots, \omega_k) := \sum_{n\geq 0} \frac{\lambda^n}{n!} \sum_{p_1,\ldots,p_k\geq 0} \frac{\omega_1^{p_1}\cdots\omega_k^{p_k}}{p_1!\cdots p_k!} \left( \begin{array}{l} \#\text{ of structures of size } n \\ \quad \text{and with } p_i \text{ occurrences of pattern } P_i \ (\text{for } 1\leq i\leq k) \end{array} \right)$$

# This talk

An **alternative approach** to enumerative combinatorics based upon **rewriting theory:**

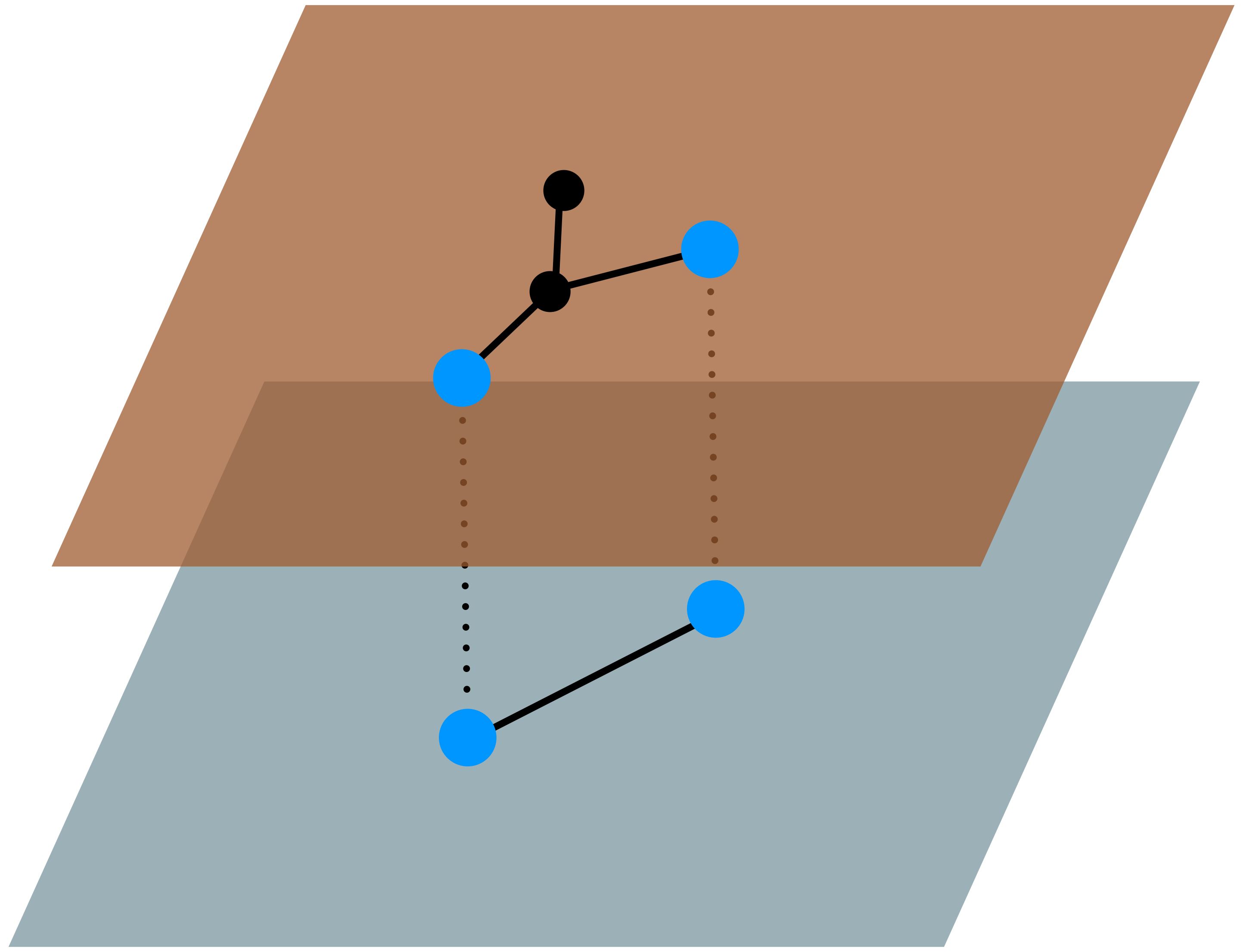



- **generate** structure **S** via applying **rewriting rules** to some **initial configuration** “in all possible ways”
- **count patterns** via applying special types of rewriting rules
- formulate **generating functions** via **linear operators** associated to rewriting rules

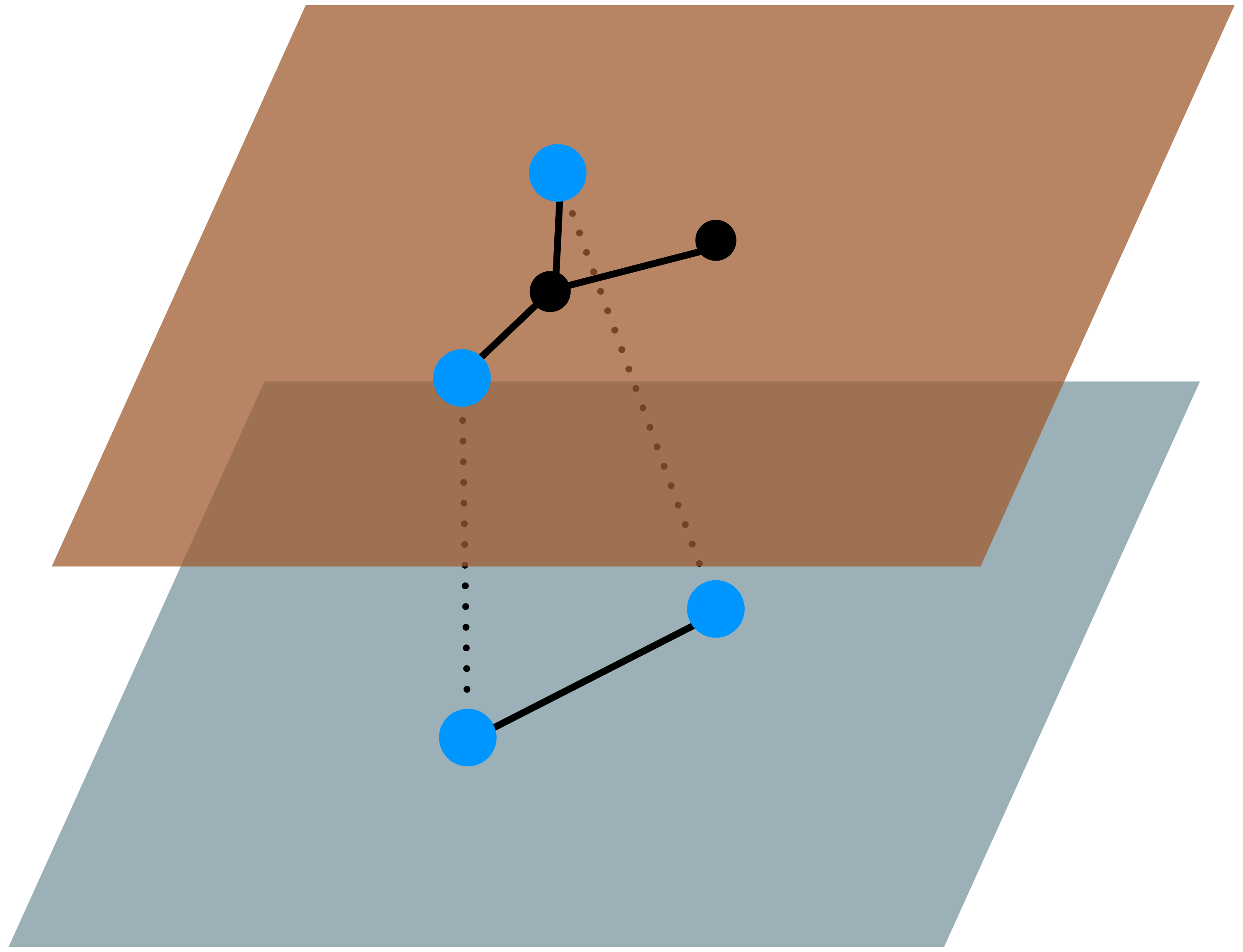
**Key tool**: the **rule-algebra** formalism!

# Example: generating **planar rooted binary trees (PRBTs)** uniformly

**The Rémy uniform generator (heuristics)**



or



# Example: generating **planar rooted binary trees (PRBTs)** uniformly

$\Rightarrow$ **combinatorics of partial observations**: rather than trying to reason about the full structure of the combinatorial species, we instead **pick** a (finite) **set of patterns** $P_1, \ldots, P_k$ and try to reason about their combinatorics within the species via **EGFs**

$$\mathcal{G}(\lambda; \omega_1, \ldots, \omega_k) := \sum_{n \geq 0} \frac{\lambda^n}{n!} \sum_{p_1, \ldots, p_k \geq 0} \frac{\omega_1^{p_1} \cdots \omega_k^{p_k}}{p_1! \cdots p_k!} \left( \begin{array}{l} \text{\# of structures of size } n \\ \quad \text{and with } p_i \text{ occurrences of pattern } P_i \text{ (for } 1 \leq i \leq k) \end{array} \right)$$

# Example: generating **planar rooted binary trees (PRBTs)** uniformly

$\Rightarrow$ **combinatorics of partial observations**: rather than trying to reason about the full structure of the combinatorial species, we instead **pick** a (finite) **set of patterns** $P_1, \ldots, P_k$ and try to reason about their combinatorics within the species via **EGFs**

$$\mathcal{G}(\lambda; \omega_1, \ldots, \omega_k) := \sum_{n \geq 0} \frac{\lambda^n}{n!} \sum_{p_1, \ldots, p_k \geq 0} \frac{\omega_1^{p_1} \cdots \omega_k^{p_k}}{p_1! \cdots p_k!} \left( \begin{array}{l} \text{\# of structures of size } n \\ \quad \text{and with } p_i \text{ occurrences of pattern } P_i \text{ (for } 1 \leq i \leq k\text{)} \end{array} \right)$$

Insight from **stochastic mechanics**: introduce so-called **observables** $\hat{O}_P$

$$\hat{O}_P \, | \, t \rangle := (\#_P(t)) \cdot | \, t \rangle$$

$P$ — a PBRT **pattern**

# of occurrences
of $P$ in the PBRT $t$

# A rule-algebraic **generating-functionology**

**Definition** Let $\langle\,|$ be defined via $\langle\,|t\rangle := 1_{\mathbb{R}}$ for arbitrary PRBT iso-class $t$ . (*Note:* this permits to implement the operation of **summation over coefficients**)

# A rule-algebraic **generating-functionology**

**Definition** Let $\langle\,|$ be defined via $\langle\,|t\rangle := 1_{\mathbb{R}}$ for arbitrary PRBT iso-class $t$ . (*Note:* this permits to implement the operation of **summation over coefficients**)

**Definition**

(12)

# A rule-algebraic **generating-functionology**

**Definition** Let $\langle\,|$ be defined via $\langle\,|t\rangle := 1_{\mathbb{R}}$ for arbitrary PRBT iso-class $t$ . (*Note:* this permits to implement the operation of **summation over coefficients**)

**Definition**

let $|\mathsf{X_0}\rangle \in \widehat{\mathbf{C}}$ denote the **initial state**

$$|X_0\rangle \tag{12}$$

# A rule-algebraic **generating-functionology**

**Definition** Let $\langle\,|$ be defined via $\langle\,|t\rangle := 1_{\mathbb{R}}$ for arbitrary PRBT iso-class $t$ . (*Note:* this permits to implement the operation of **summation over coefficients**)

**Definition** Let $\widehat{\mathsf{G}}$ be a linear operator (the **generator**),
let $|X_0\rangle \in \widehat{\mathbf{C}}$ denote the **initial state**

$$e^{\lambda \hat{G}}\,|X_0\rangle \tag{12}$$

# A rule-algebraic **generating-functionology**

**Definition** Let $\langle\,|$ be defined via $\langle\,|t\rangle := 1_{\mathbb{R}}$ for arbitrary PRBT iso-class $t$ . (*Note:* this permits to implement the operation of **summation over coefficients**)

**Definition** Let $\widehat{\mathsf{G}}$ be a linear operator (the **generator**), let $\widehat{\mathsf{O}}_1, \ldots, \widehat{\mathsf{O}}_m$ be a choice of (finitely many) **pattern observables**, and let $|X_0\rangle \in \widehat{\mathbf{C}}$ denote the **initial state**

$$e^{\underline{\omega}\cdot\underline{\hat{O}}}e^{\lambda\hat{G}}\,|X_0\rangle \tag{12}$$

# A rule-algebraic **generating-functionology**

**Definition** Let $\langle\,|$ be defined via $\langle\,|t\rangle := 1_{\mathbb{R}}$ for arbitrary PRBT iso-class $t$ . (*Note:* this permits to implement the operation of **summation over coefficients**)

**Definition** Let $\widehat{\mathsf{G}}$ be a linear operator (the **generator**), let $\widehat{\mathsf{O}}_1, \ldots, \widehat{\mathsf{O}}_m$ be a choice of (finitely many) **pattern observables**, and let $|X_0\rangle \in \widehat{\mathbf{C}}$ denote the **initial state**. Then the **exponential moment-generating function (EMGF)** $\mathcal{G}(\lambda; \underline{\omega})$ is defined as

$$\mathcal{G}(\lambda; \underline{\omega}) := \langle|\, e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda \hat{G}}\, |X_0\rangle \tag{12}$$

Here, we employed the shorthand notation $\underline{\omega} \cdot \underline{\widehat{\mathsf{O}}} := \sum_{j=1}^{m} \omega_j \widehat{\mathsf{O}}_j$, and $\lambda$ as well as $\omega_1, \ldots, \omega_m$ are **formal variables**.

# Combinatorial evolution equations

The **formal EMGF evolution equation** for $\mathcal{G}(\lambda;\underline{\omega})$ reads as follows:

$$\frac{\partial}{\partial\lambda}\mathcal{G}(\lambda;\underline{\omega}) = \langle| \left(e^{ad_{\underline{\omega}\cdot\underline{\hat{O}}}}\hat{G}\right) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}} |X_0\rangle \qquad (ad_A(B) := AB - BA) \tag{15}$$

# Combinatorial evolution equations

The **formal EMGF evolution equation** for $\mathcal{G}(\lambda;\underline{\omega})$ reads as follows:

$$\tfrac{\partial}{\partial\lambda}\mathcal{G}(\lambda;\underline{\omega}) = \langle| \left(e^{ad_{\underline{\omega}\cdot\underline{\hat{O}}}}\hat{G}\right) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}}\,|X_0\rangle \qquad (ad_A(B) := AB - BA) \tag{15}$$

Applying the version of the **jump-closure theorem** appropriate for the chosen rewriting semantics (DPO or SqPO), the above formal evolution equation may be converted into a proper **evolution equation on formal power series** if the following **polynomial jump-closure** holds:

$$(\mathrm{PJC}') \quad \forall q \in \mathbb{Z}_{\geq 0} : \exists \underline{N(n)} \in \mathbb{Z}^m_{\geq 0}, \gamma_q(\underline{\omega},\underline{k}) \in \mathbb{R} : \; \langle| \, ad^{\circ\, q}_{\underline{\omega}\cdot\underline{\hat{O}}}(\hat{G}) = \sum_{\underline{k}=\underline{0}}^{\underline{N(q)}} \gamma_{\underline{k}}(\underline{\omega},\underline{k}) \, \langle| \, \underline{\hat{O}}^{\underline{k}} \tag{16}$$

# Combinatorial evolution equations

The **formal EMGF evolution equation** for $\mathcal{G}(\lambda;\underline{\omega})$ reads as follows:

$$\frac{\partial}{\partial\lambda}\mathcal{G}(\lambda;\underline{\omega}) = \langle| \left(e^{ad_{\underline{\omega}\cdot\underline{\hat{O}}}}\hat{G}\right) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}} |X_0\rangle \qquad (ad_A(B) := AB - BA) \tag{15}$$

Applying the version of the **jump-closure theorem** appropriate for the chosen rewriting semantics (DPO or SqPO), the above formal evolution equation may be converted into a proper **evolution equation on formal power series** if the following **polynomial jump-closure** holds:

$$(\mathrm{PJC}') \quad \forall q \in \mathbb{Z}_{\geq 0} : \exists \underline{N}(n) \in \mathbb{Z}^m_{\geq 0}, \gamma_q(\underline{\omega},\underline{k}) \in \mathbb{R} : \ \langle| ad^{\circ q}_{\underline{\omega}\cdot\underline{\hat{O}}}(\hat{G}) = \sum_{\underline{k}=\underline{0}}^{\underline{N}(q)} \gamma_{\underline{k}}(\underline{\omega},\underline{k}) \langle| \underline{\hat{O}}^{\underline{k}} \tag{16}$$

# Combinatorial evolution equations

The **formal EMGF evolution equation** for $\mathcal{G}(\lambda;\underline{\omega})$ reads as follows:

$$\tfrac{\partial}{\partial\lambda}\mathcal{G}(\lambda;\underline{\omega}) = \langle| \left(e^{ad_{\underline{\omega}\cdot\underline{\hat{O}}}}\hat{G}\right) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}}\,|X_0\rangle \qquad (ad_A(B) := AB - BA) \tag{15}$$

Applying the version of the **jump-closure theorem** appropriate for the chosen rewriting semantics (DPO or SqPO), the above formal evolution equation may be converted into a proper **evolution equation on formal power series** if the following **polynomial jump-closure** holds:

$$(\mathrm{PJC}') \quad \forall q \in \mathbb{Z}_{\geq 0} : \exists \underline{N(n)} \in \mathbb{Z}^m_{\geq 0}, \gamma_q(\underline{\omega},\underline{k}) \in \mathbb{R} : \ \langle| ad^{\circ q}_{\underline{\omega}\cdot\underline{\hat{O}}}(\hat{G}) = \sum_{\underline{k}=\underline{0}}^{\underline{N(q)}} \gamma_{\underline{k}}(\underline{\omega},\underline{k})\,\langle| \underline{\hat{O}}^{\underline{k}} \tag{16}$$

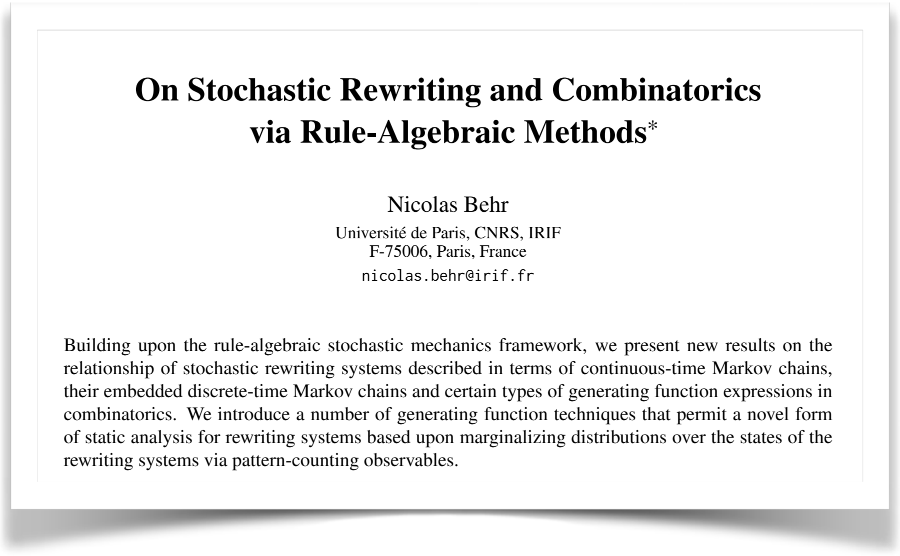
If a given set of observables satisfies $(\mathrm{PJC}')$, the **formal evolution equation** (12) for the EMGF $\mathcal{G}(\lambda;\underline{\omega})$ may be refined into

$$\tfrac{\partial}{\partial\lambda}\mathcal{G}(\lambda;\underline{\omega}) = \mathbb{G}(\underline{\omega},\underline{\partial\omega})\mathcal{G}(\lambda;\underline{\omega})\,, \quad \mathbb{G}(\underline{\omega},\underline{\partial\omega}) = \left(\langle| e^{ad_{\underline{\omega}\cdot\underline{\hat{O}}}}(\hat{G})\right)\Big|_{\underline{\hat{O}}\mapsto\underline{\partial\omega}}\,. \tag{17}$$

# Example: generating **planar rooted binary trees (PRBTs)** uniformly

$\hat{O}_E :=$ 

Pattern $E$ : an edge of any type

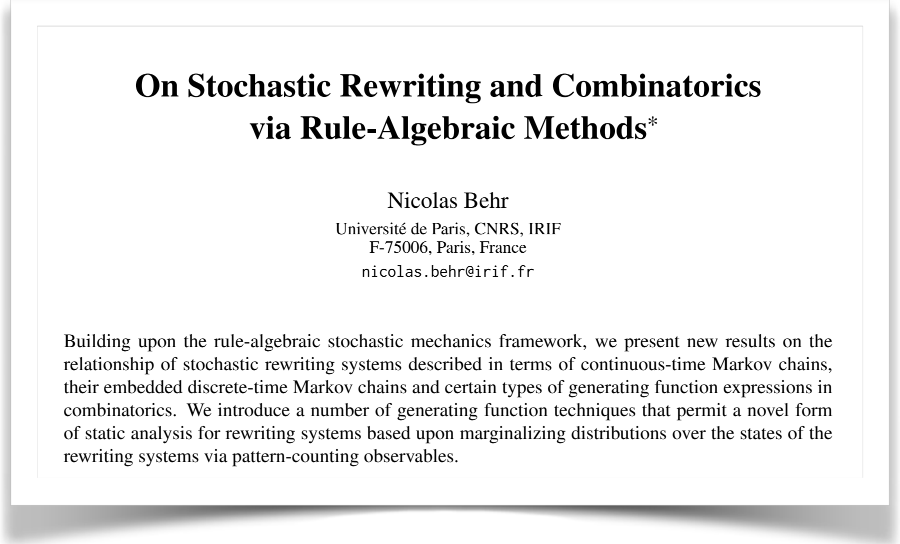


**On Stochastic Rewriting and Combinatorics via Rule-Algebraic Methods***

Nicolas Behr
Université de Paris, CNRS, IRIF
F-75006, Paris, France
nicolas.behr@irif.fr

Building upon the rule-algebraic stochastic mechanics framework, we present new results on the relationship of stochastic rewriting systems described in terms of continuous-time Markov chains, their embedded discrete-time Markov chains and certain types of generating function expressions in combinatorics. We introduce a number of generating function techniques that permit a novel form of static analysis for rewriting systems based upon marginalizing distributions over the states of the rewriting systems via pattern-counting observables.

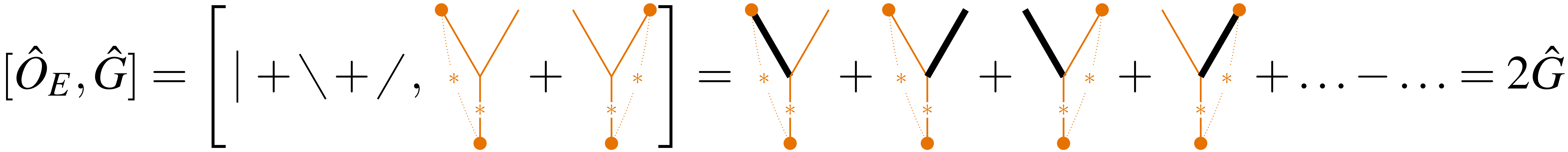
# Example: generating **planar rooted binary trees (PRBTs)** uniformly


On Stochastic Rewriting and Combinatorics
via Rule-Algebraic Methods*

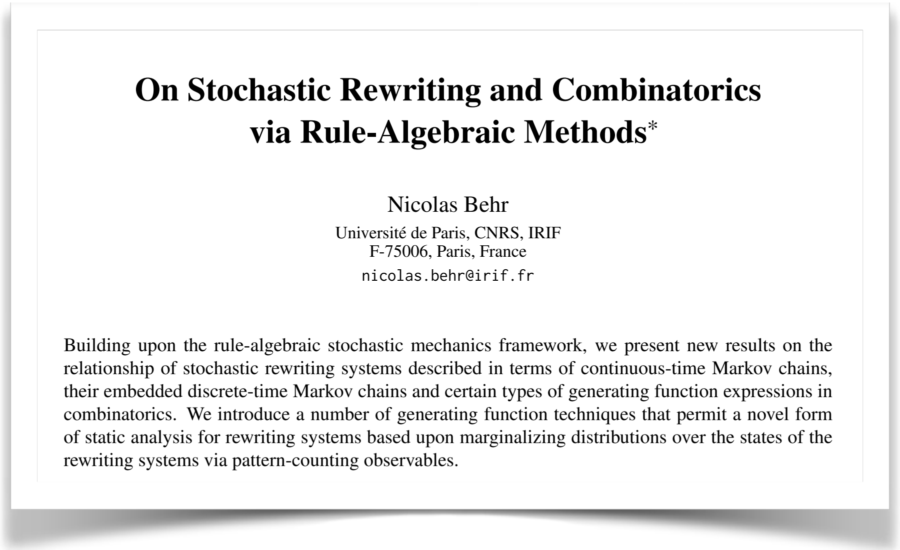
Nicolas Behr
Université de Paris, CNRS, IRIF
F-75006, Paris, France
nicolas.behr@irif.fr

Building upon the rule-algebraic stochastic mechanics framework, we present new results on the relationship of stochastic rewriting systems described in terms of continuous-time Markov chains, their embedded discrete-time Markov chains and certain types of generating function expressions in combinatorics. We introduce a number of generating function techniques that permit a novel form of static analysis for rewriting systems based upon marginalizing distributions over the states of the rewriting systems via pattern-counting observables.

$\hat{O}_E :=$  Pattern $E$ : an edge of any type



# Example: generating **planar rooted binary trees (PRBTs)** uniformly



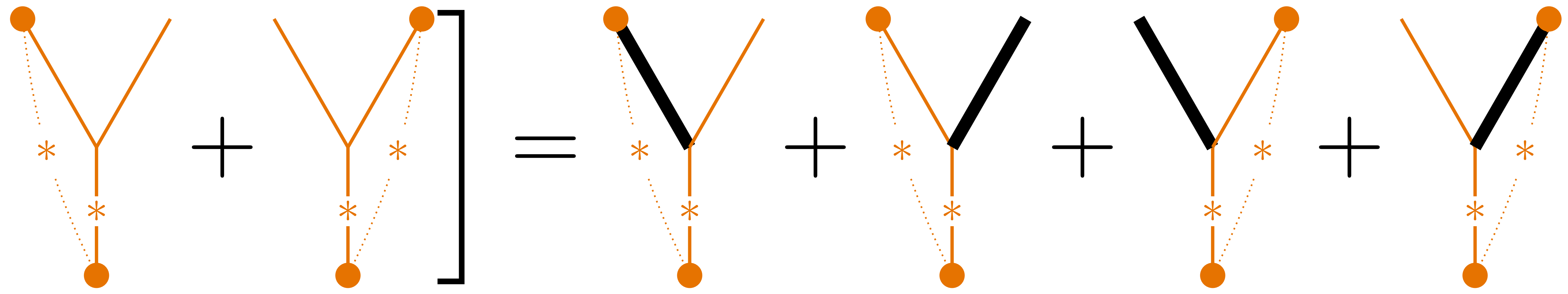
On Stochastic Rewriting and Combinatorics via Rule-Algebraic Methods*

Nicolas Behr
Université de Paris, CNRS, IRIF
F-75006, Paris, France
nicolas.behr@irif.fr

Building upon the rule-algebraic stochastic mechanics framework, we present new results on the relationship of stochastic rewriting systems described in terms of continuous-time Markov chains, their embedded discrete-time Markov chains and certain types of generating function expressions in combinatorics. We introduce a number of generating function techniques that permit a novel form of static analysis for rewriting systems based upon marginalizing distributions over the states of the rewriting systems via pattern-counting observables.

$\hat{O}_E :=$  Pattern $E$ : an edge of any type

$$[\hat{O}_E, \hat{G}] = \Big[\, | + \backslash + /,$$

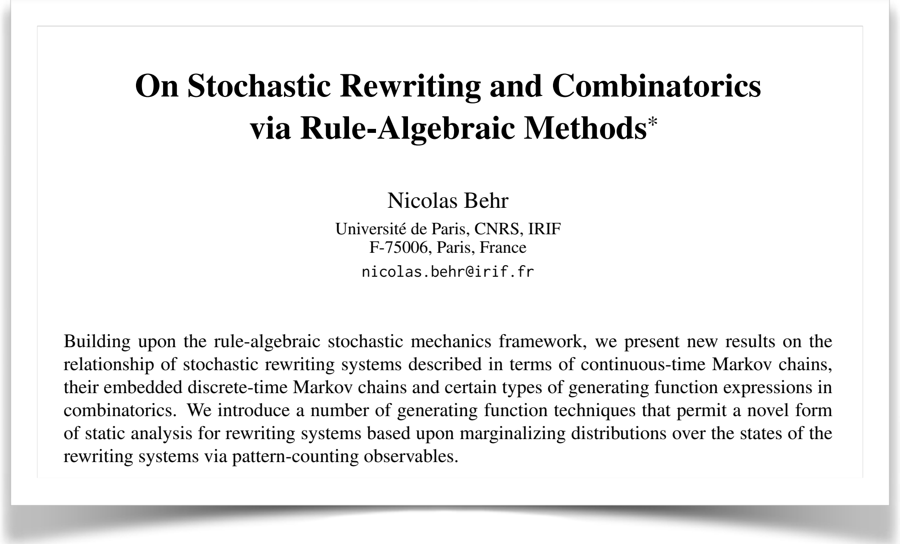


$$+ \ldots - \ldots = 2\hat{G}$$

$$\langle | \, \hat{G} = 2 \, \langle | \, \hat{O}_E$$

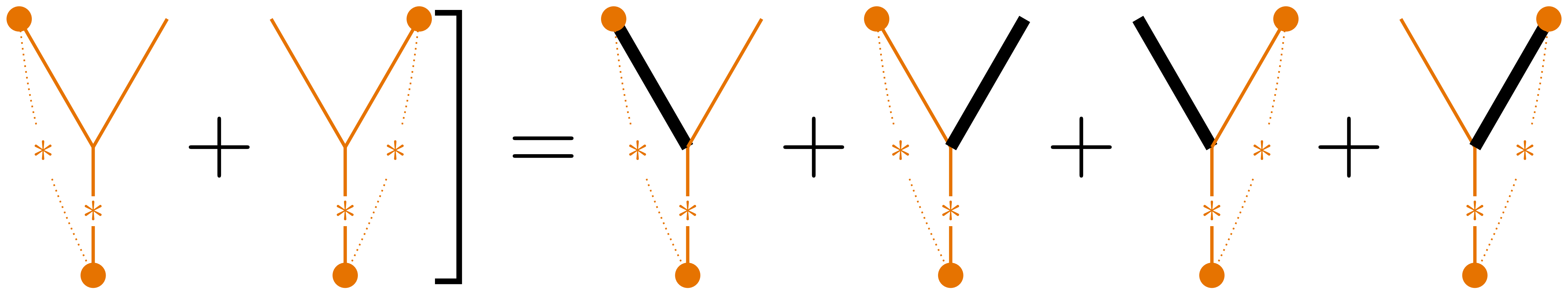
# Example: generating **planar rooted binary trees (PRBTs)** uniformly

$\hat{O}_E :=$  Pattern $E$ : an edge of any type


**On Stochastic Rewriting and Combinatorics via Rule-Algebraic Methods***

Nicolas Behr
Université de Paris, CNRS, IRIF
F-75006, Paris, France
nicolas.behr@irif.fr

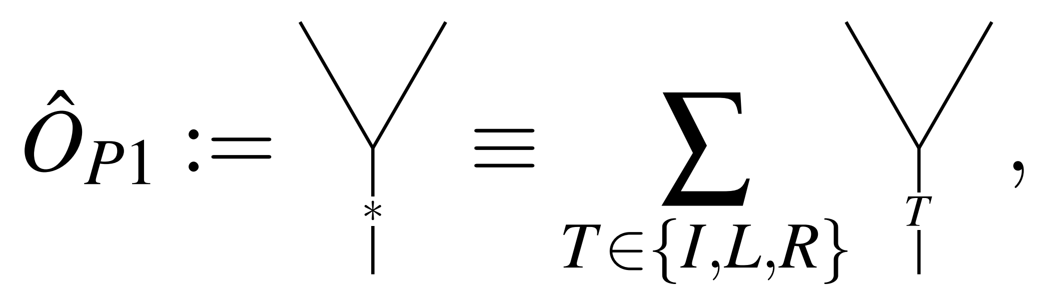
Building upon the rule-algebraic stochastic mechanics framework, we present new results on the relationship of stochastic rewriting systems described in terms of continuous-time Markov chains, their embedded discrete-time Markov chains and certain types of generating function expressions in combinatorics. We introduce a number of generating function techniques that permit a novel form of static analysis for rewriting systems based upon marginalizing distributions over the states of the rewriting systems via pattern-counting observables.

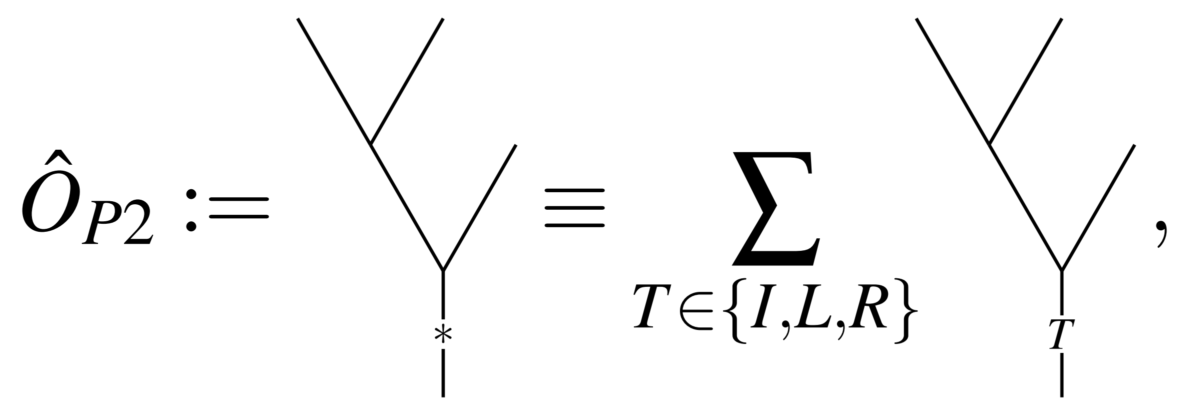
$[\hat{O}_E, \hat{G}] = \Big[\, | + \backslash + / ,$  $+ \ldots - \ldots = 2\hat{G}$

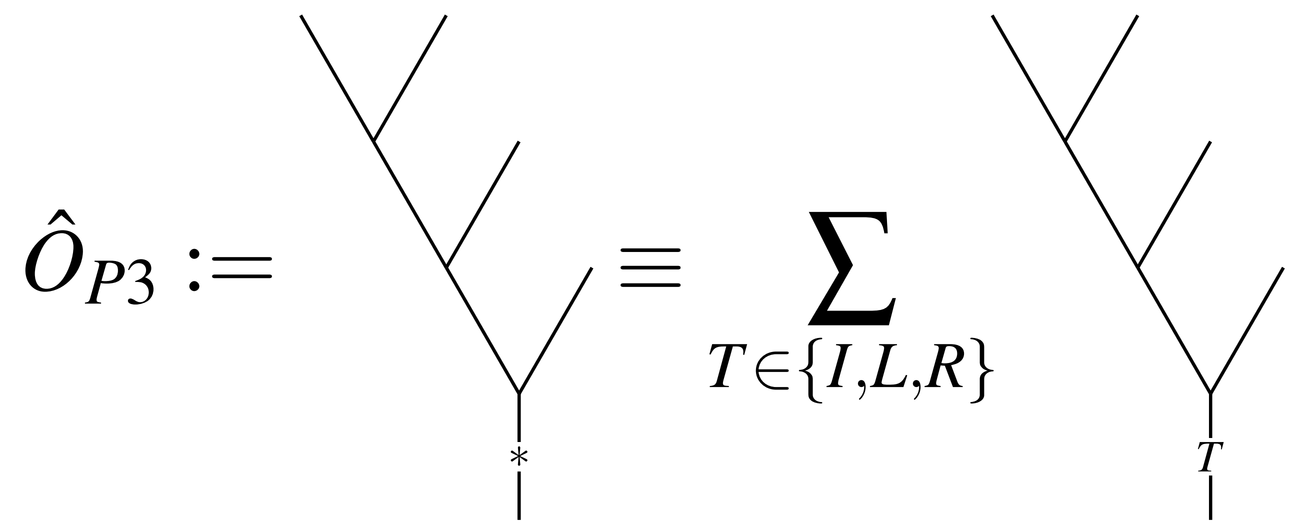
$$\langle | \,\hat{G} = 2\, \langle | \,\hat{O}_E$$

$$\begin{cases} \frac{\partial}{\partial\lambda}\mathscr{G}(\lambda;\varepsilon) = 2e^{2\varepsilon}\frac{\partial}{\partial\varepsilon}\mathscr{G}(\lambda;\varepsilon) \\ \mathscr{G}(0;\varepsilon) = \langle | \, e^{\varepsilon \hat{O}_E} \, | | \rangle = e^{\varepsilon} \end{cases} \Rightarrow \quad \mathscr{G}(\lambda;\varepsilon) = \frac{1}{\sqrt{e^{-2\varepsilon} - 4\lambda}} = \sum_{n\geq 0} \frac{\lambda^n}{n!}\left(\frac{(2n)!}{n!}e^{\varepsilon(2n+1)}\right)$$

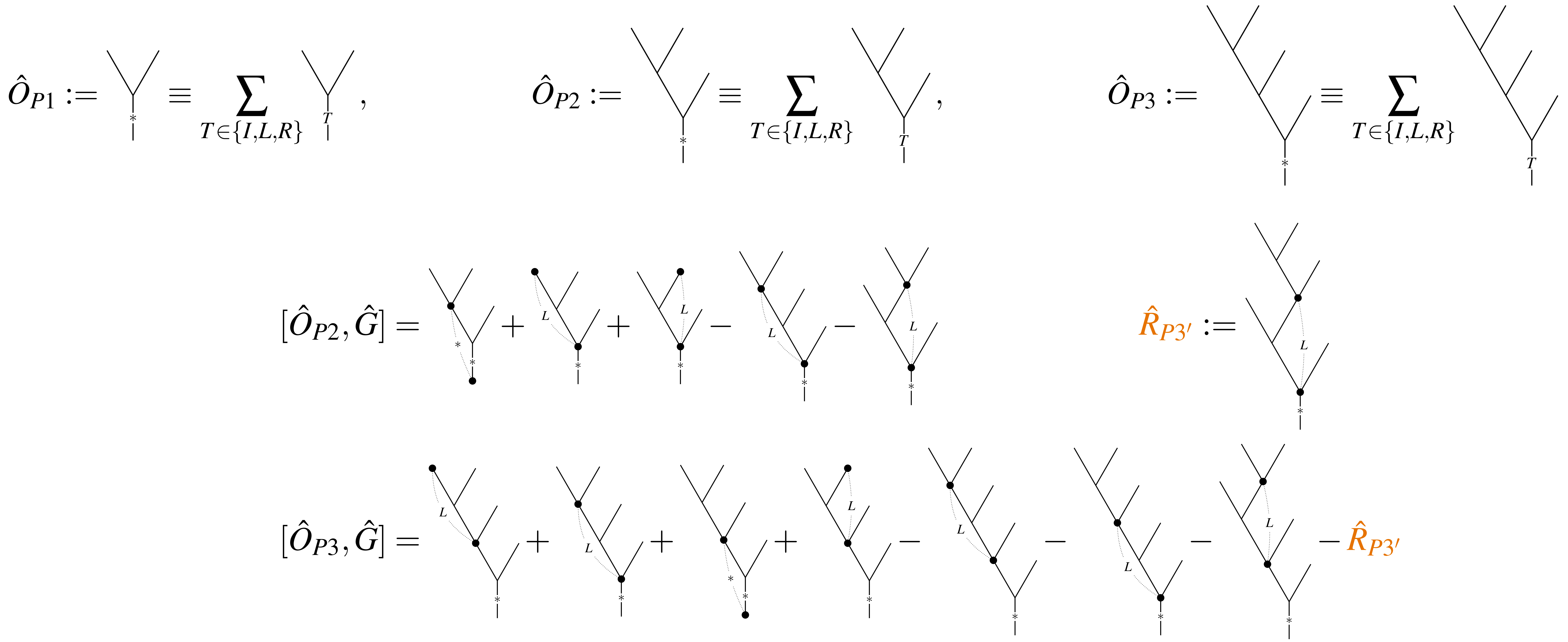
# Example: generating **planar rooted binary trees (PRBTs)** uniformly



# Example: generating **planar rooted binary trees (PRBTs)** uniformly

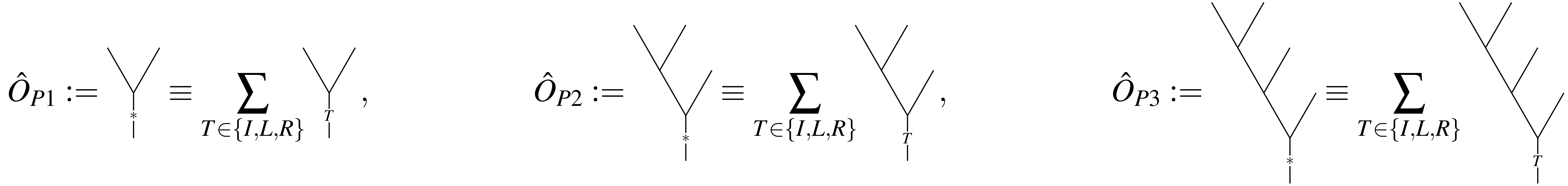


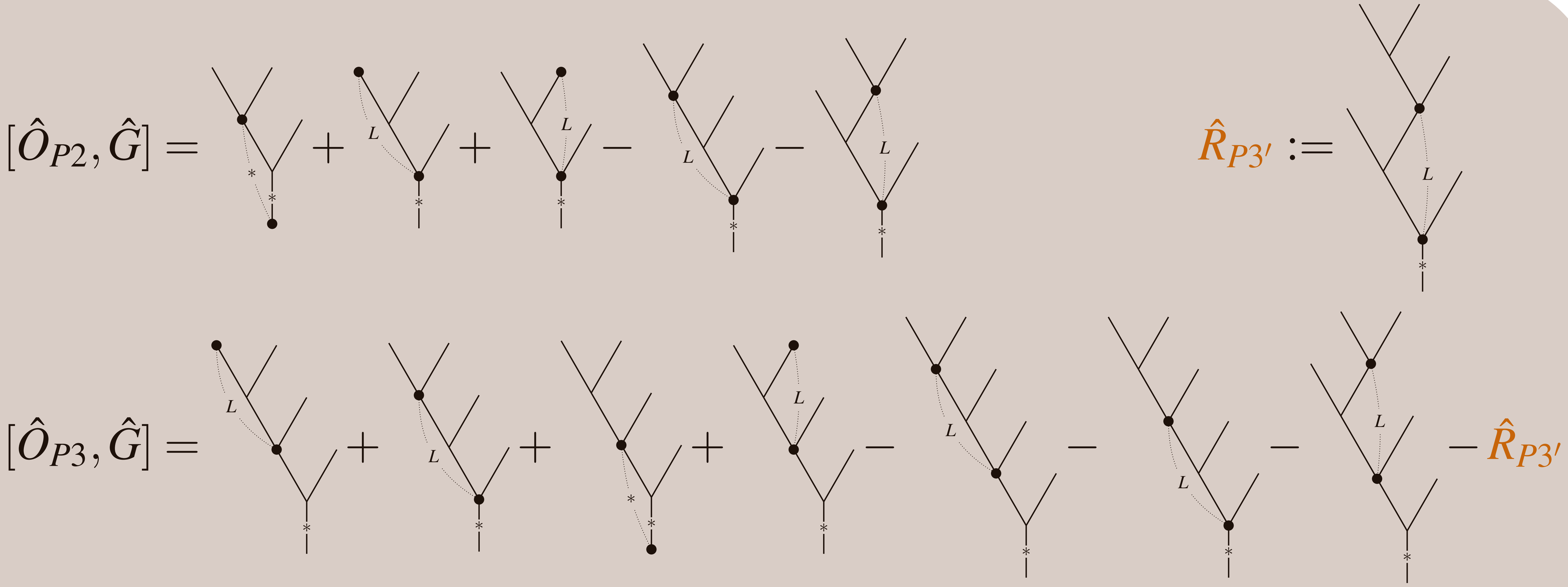

$$[\hat{O}_{P2},[\hat{O}_{P2},\hat{G}]] = [\hat{O}_{P2},\hat{G}]\,, \quad [\hat{O}_{P2},[\hat{O}_{P3},\hat{G}]] = [\hat{O}_{P3},\hat{G}] + \hat{R}_{P3}$$

$$[\hat{O}_{P3},[\hat{O}_{P3},\hat{G}]] = [\hat{O}_{P3},\hat{G}] + 2\hat{R}_{P3'}\,, \quad [\hat{O}_{P2},\hat{R}_{P3'}] = 0\,, \quad [\hat{O}_{P3},\hat{R}_{P3'}] = -\hat{R}_{P3'}$$

$$\langle|\,[\hat{O}_{P2},\hat{G}] = \langle|\,(3\hat{O}_{P1} - 2\hat{O}_{P2})\,, \quad \langle|\,[\hat{O}_{P3},\hat{G}] = \langle|\,(4\hat{O}_{P2} - 3\hat{O}_{P3})\,, \quad \langle|\,\hat{R}_{P3'} = \langle|\,\hat{O}_{P3}$$

# Example: generating **planar rooted binary trees (PRBTs)** uniformly



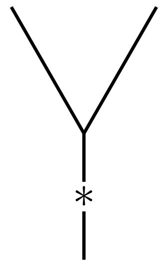
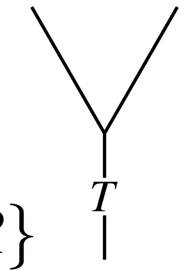

$$[\hat{O}_{P2},[\hat{O}_{P2},\hat{G}]] = [\hat{O}_{P2},\hat{G}]\,, \quad [\hat{O}_{P2},[\hat{O}_{P3},\hat{G}]] = [\hat{O}_{P3},\hat{G}] + \hat{R}_{P3}$$

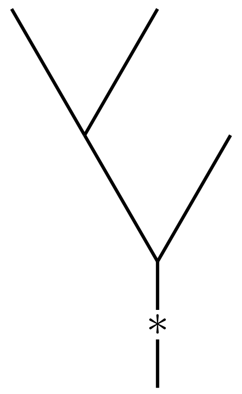
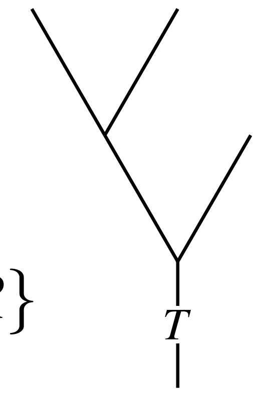
$$[\hat{O}_{P3},[\hat{O}_{P3},\hat{G}]] = [\hat{O}_{P3},\hat{G}] + 2\hat{R}_{P3'}\,, \quad [\hat{O}_{P2},\hat{R}_{P3'}] = 0\,, \quad [\hat{O}_{P3},\hat{R}_{P3'}] = -\hat{R}_{P3'}$$

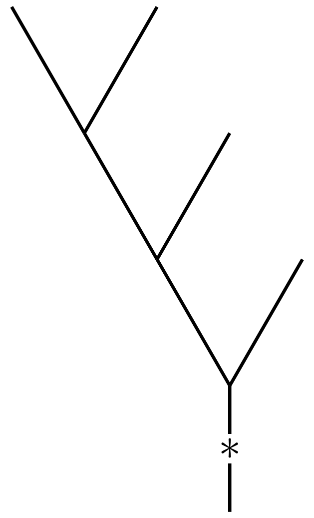
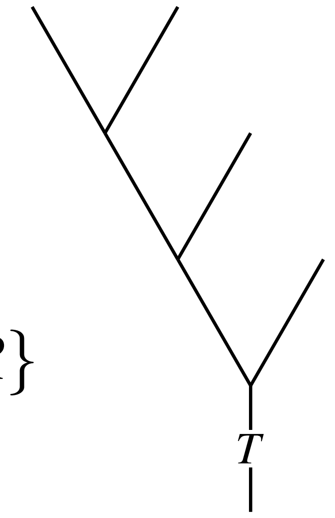
$$\langle|\,[\hat{O}_{P2},\hat{G}] = \langle|\,(3\hat{O}_{P1} - 2\hat{O}_{P2})\,, \quad \langle|\,[\hat{O}_{P3},\hat{G}] = \langle|\,(4\hat{O}_{P2} - 3\hat{O}_{P3})\,, \quad \langle|\,\hat{R}_{P3'} = \langle|\,\hat{O}_{P3}$$

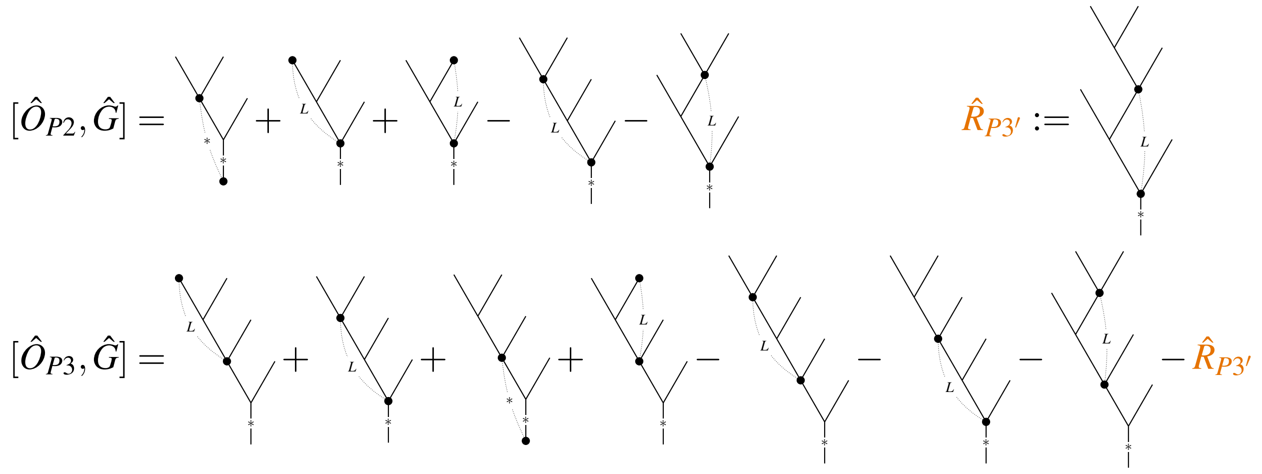
?

# Example: generating **planar rooted binary trees (PRBTs)** uniformly

$\hat{O}_{P1} :=$  $\equiv \sum_{T\in\{I,L,R\}}$  ,

$\hat{O}_{P2} :=$  $\equiv \sum_{T\in\{I,L,R\}}$  ,

$\hat{O}_{P3} :=$  $\equiv \sum_{T\in\{I,L,R\}}$ 




$$[\hat{O}_{P2},[\hat{O}_{P2},\hat{G}]] = [\hat{O}_{P2},\hat{G}]\,,\quad [\hat{O}_{P2},[\hat{O}_{P3},\hat{G}]] = [\hat{O}_{P3},\hat{G}] + \hat{R}_{P3}$$

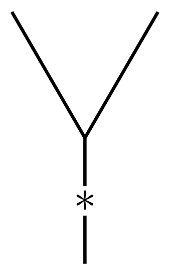
$$[\hat{O}_{P3},[\hat{O}_{P3},\hat{G}]] = [\hat{O}_{P3},\hat{G}] + 2\hat{R}_{P3'}\,,\quad [\hat{O}_{P2},\hat{R}_{P3'}] = 0\,,\quad [\hat{O}_{P3},\hat{R}_{P3'}] = -\hat{R}_{P3'}$$

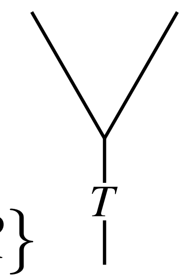
$$\langle|\,[\hat{O}_{P2},\hat{G}] = \langle|\,(3\hat{O}_{P1} - 2\hat{O}_{P2})\,,\quad \langle|\,[\hat{O}_{P3},\hat{G}] = \langle|\,(4\hat{O}_{P2} - 3\hat{O}_{P3})\,,\quad \langle|\,\hat{R}_{P3'} = \langle|\,\hat{O}_{P3}$$

$$\mathscr{G}(\lambda;\underline{\omega}) := \langle|\, e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}}\,|\,\rangle\,,\quad \underline{\omega}\cdot\underline{\hat{O}} := \varepsilon\hat{O}_E + \gamma\hat{O}_{P1} + \mu\hat{O}_{P2} + \nu\hat{O}_{P3}$$

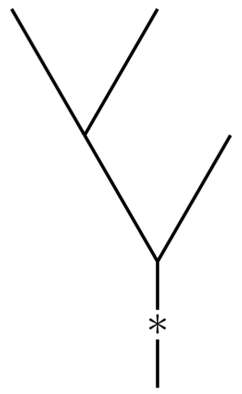
$$\begin{aligned}
\frac{\partial}{\partial\lambda}\mathscr{G}(\lambda;\underline{\omega}) &= \langle|\left(e^{ad_{\underline{\omega}\cdot\underline{\hat{O}}}}(\hat{G})\right) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}}\,|\,\rangle \overset{(*)}{=} \langle|\left(e^{ad_{\nu\hat{O}_{P3}}}\left(e^{ad_{\mu\hat{O}_{P2}}}\left(e^{ad_{\varepsilon\hat{O}_E+\gamma\hat{O}_{P1}}}(\hat{G})\right)\right)\right) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}}\,|\,\rangle\\
&= e^{2\varepsilon+\gamma}\langle|\left(e^{ad_{\nu\hat{O}_{P3}}}\left(e^{ad_{\mu\hat{O}_{P2}}}(\hat{G})\right)\right) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}}\,|\,\rangle\\
&= e^{2\varepsilon+\gamma}\langle|\left(e^{ad_{\nu\hat{O}_{P3}}}\left(\hat{G} + (e^{\mu}-1)[\hat{O}_{P2},\hat{G}]\right)\right) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}}\,|\,\rangle\\
&= e^{2\varepsilon+\gamma}\langle|\big(\hat{G} + (e^{\mu}-1)[\hat{O}_{P2},\hat{G}]\\
&\qquad + e^{\mu}(e^{\nu}-1)[\hat{O}_{P3},\hat{G}] + (e^{\nu}-1)(e^{\mu}-e^{-\nu})\hat{R}_{P3'}\big) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}}\,|\,\rangle\\
&= e^{2\varepsilon+\gamma}\langle|\big(2\hat{O}_E + 3(e^{\mu}-1)\hat{O}_{P1} + (4e^{\mu+\nu} - 6e^{\mu} + 2)\hat{O}_{P2}\\
&\qquad + (3e^{\mu} + e^{-\nu} - 3e^{\mu+\nu} - 1)\hat{O}_{P3}\big) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}}\,|\,\rangle\\
&= e^{2\varepsilon+\gamma}\langle|\big(2\tfrac{\partial}{\partial\varepsilon} + 3(e^{\mu}-1)\tfrac{\partial}{\partial\gamma} + (4e^{\mu+\nu} - 6e^{\mu} + 2)\tfrac{\partial}{\partial\mu}\\
&\qquad + (3e^{\mu} + e^{-\nu} - 3e^{\mu+\nu} - 1)\tfrac{\partial}{\partial\nu}\big) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}}\,|\,\rangle
\end{aligned}$$

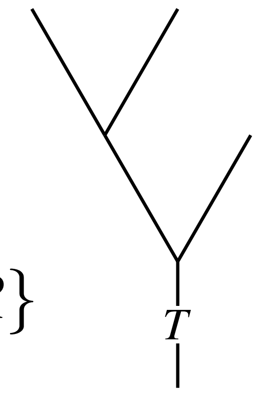
# Example: generating **planar rooted binary trees (PRBTs)** uniformly

$\hat{O}_{P1} :=$ 

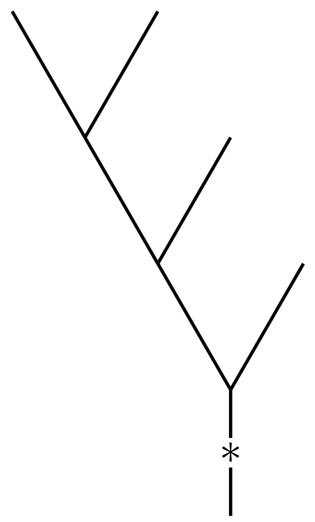
$\equiv \sum_{T\in\{I,L,R\}}$ 

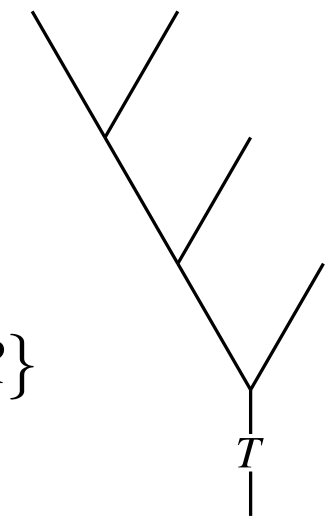
,

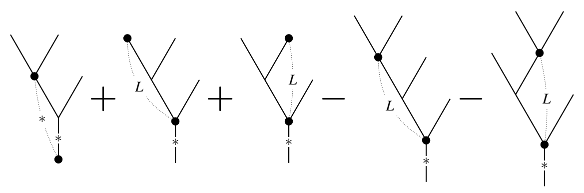
$\hat{O}_{P2} :=$ 

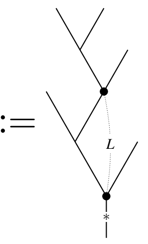
$\equiv \sum_{T\in\{I,L,R\}}$ 

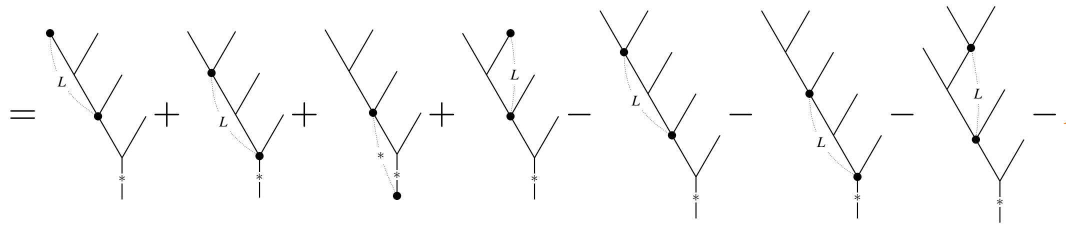
,

$\hat{O}_{P3} :=$ 

$\equiv \sum_{T\in\{I,L,R\}}$ 


$[\hat{O}_{P2},\hat{G}] =$ 

$\hat{R}_{P3'} :=$ 

$[\hat{O}_{P3},\hat{G}] =$  $-\hat{R}_{P3'}$

$$[\hat{O}_{P2},[\hat{O}_{P2},\hat{G}]] = [\hat{O}_{P2},\hat{G}]\,,\quad [\hat{O}_{P2},[\hat{O}_{P3},\hat{G}]] = [\hat{O}_{P3},\hat{G}] + \hat{R}_{P3}$$
$$[\hat{O}_{P3},[\hat{O}_{P3},\hat{G}]] = [\hat{O}_{P3},\hat{G}] + 2\hat{R}_{P3'}\,,\quad [\hat{O}_{P2},\hat{R}_{P3'}] = 0\,,\quad [\hat{O}_{P3},\hat{R}_{P3'}] = -\hat{R}_{P3'}$$
$$\langle|\,[\hat{O}_{P2},\hat{G}] = \langle|\,(3\hat{O}_{P1}-2\hat{O}_{P2})\,,\quad \langle|\,[\hat{O}_{P3},\hat{G}] = \langle|\,(4\hat{O}_{P2}-3\hat{O}_{P3})\,,\quad \langle|\,\hat{R}_{P3'} = \langle|\,\hat{O}_{P3}$$

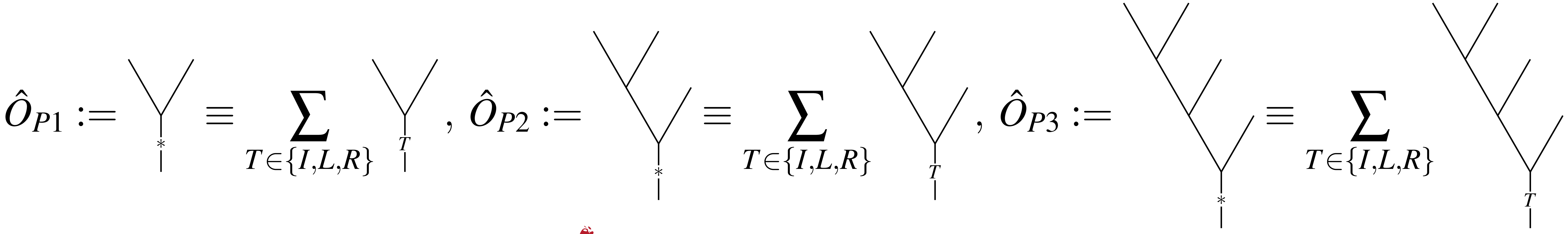
$$\mathscr{G}(\lambda;\underline{\omega}) := \langle|\, e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}}\,||\rangle\,,\quad \underline{\omega}\cdot\underline{\hat{O}} := \varepsilon\hat{O}_E + \gamma\hat{O}_{P1} + \mu\hat{O}_{P2} + \nu\hat{O}_{P3}$$

$$\begin{aligned}
\tfrac{\partial}{\partial\lambda}\mathscr{G}(\lambda;\underline{\omega}) &= \langle|\left(e^{ad_{\underline{\omega}\cdot\underline{\hat{O}}}}(\hat{G})\right) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}}\,||\rangle \overset{(*)}{=} \langle|\left(e^{ad_{\nu\hat{O}_{P3}}}\left(e^{ad_{\mu\hat{O}_{P2}}}\left(e^{ad_{\varepsilon\hat{O}_E+\gamma\hat{O}_{P1}}}(\hat{G})\right)\right)\right) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}}\,||\rangle\\
&= e^{2\varepsilon+\gamma}\langle|\left(e^{ad_{\nu\hat{O}_{P3}}}\left(e^{ad_{\mu\hat{O}_{P2}}}(\hat{G})\right)\right) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}}\,||\rangle\\
&= e^{2\varepsilon+\gamma}\langle|\left(e^{ad_{\nu\hat{O}_{P3}}}\left(\hat{G}+(e^{\mu}-1)[\hat{O}_{P2},\hat{G}]\right)\right) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}}\,||\rangle\\
&= e^{2\varepsilon+\gamma}\langle|\big(\hat{G}+(e^{\mu}-1)[\hat{O}_{P2},\hat{G}]\\
&\qquad +e^{\mu}(e^{\nu}-1)[\hat{O}_{P3},\hat{G}]+(e^{\nu}-1)(e^{\mu}-e^{-\nu})\hat{R}_{P3'}\big) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}}\,||\rangle\\
&= e^{2\varepsilon+\gamma}\langle|\big(2\hat{O}_E+3(e^{\mu}-1)\hat{O}_{P1}+(4e^{\mu+\nu}-6e^{\mu}+2)\hat{O}_{P2}\\
&\qquad +(3e^{\mu}+e^{-\nu}-3e^{\mu+\nu}-1)\hat{O}_{P3}\big) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}}\,||\rangle\\
&= e^{2\varepsilon+\gamma}\langle|\big(2\tfrac{\partial}{\partial\varepsilon}+3(e^{\mu}-1)\tfrac{\partial}{\partial\gamma}+(4e^{\mu+\nu}-6e^{\mu}+2)\tfrac{\partial}{\partial\mu}\\
&\qquad +(3e^{\mu}+e^{-\nu}-3e^{\mu+\nu}-1)\tfrac{\partial}{\partial\nu}\big) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}}\,||\rangle
\end{aligned}$$

Granted that the derivation of the evolution equation for $\mathscr{G}(\lambda;\underline{\omega})$ is somewhat involved, one may extract from it a very interesting insight via a transformation of variables $\omega_i \to \ln x_i$ (which entails that $\frac{\partial}{\partial\omega_i} \to x_i\frac{\partial}{\partial x_i}$), and collecting coefficients for the operators $\hat{n}_i := x_i\frac{\partial}{\partial x_i}$:

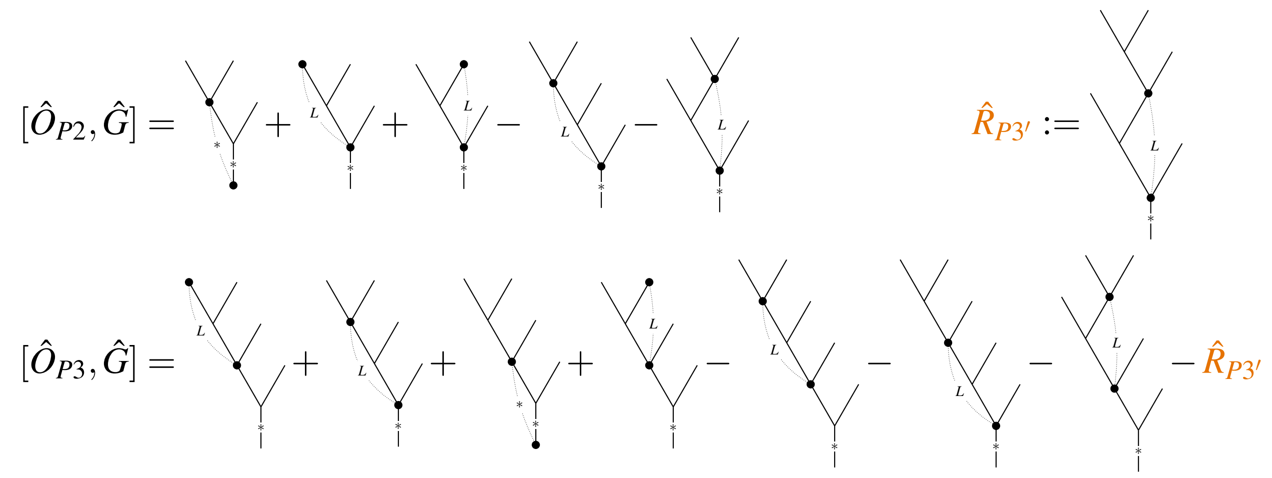
$$\begin{aligned}
&\tfrac{\partial}{\partial\lambda}\mathscr{G}(\lambda;\underline{\ln x}) = \hat{\mathsf{D}}\mathscr{G}(\lambda;\underline{\ln x})\\
&\hat{\mathsf{D}} = x_{\varepsilon}^2x_{\nu}(2\hat{n}_{\varepsilon}-3\hat{n}_{\gamma}+2\hat{n}_{\mu}-\hat{n}_{\nu})+x_{\varepsilon}^2x_{\nu}x_{\mu}(3\hat{n}_{\gamma}-6\hat{n}_{\mu}+3\hat{n}_{\nu})+x_{\varepsilon}^2x_{\nu}x_{\mu}^2(4\hat{n}_{\mu}-3\hat{n}_{\nu})+x_{\varepsilon}^2\hat{n}_{\nu}
\end{aligned}\tag{58}$$

# Example: generating **planar rooted binary trees (PRBTs)** uniformly




$$\mathscr{G}(\lambda;\underline{\omega}) := \langle| e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}} ||\rangle\,, \quad \underline{\omega}\cdot\underline{\hat{O}} := \varepsilon\hat{O}_E + \gamma\hat{O}_{P1} + \mu\hat{O}_{P2} + \nu\hat{O}_{P3}$$

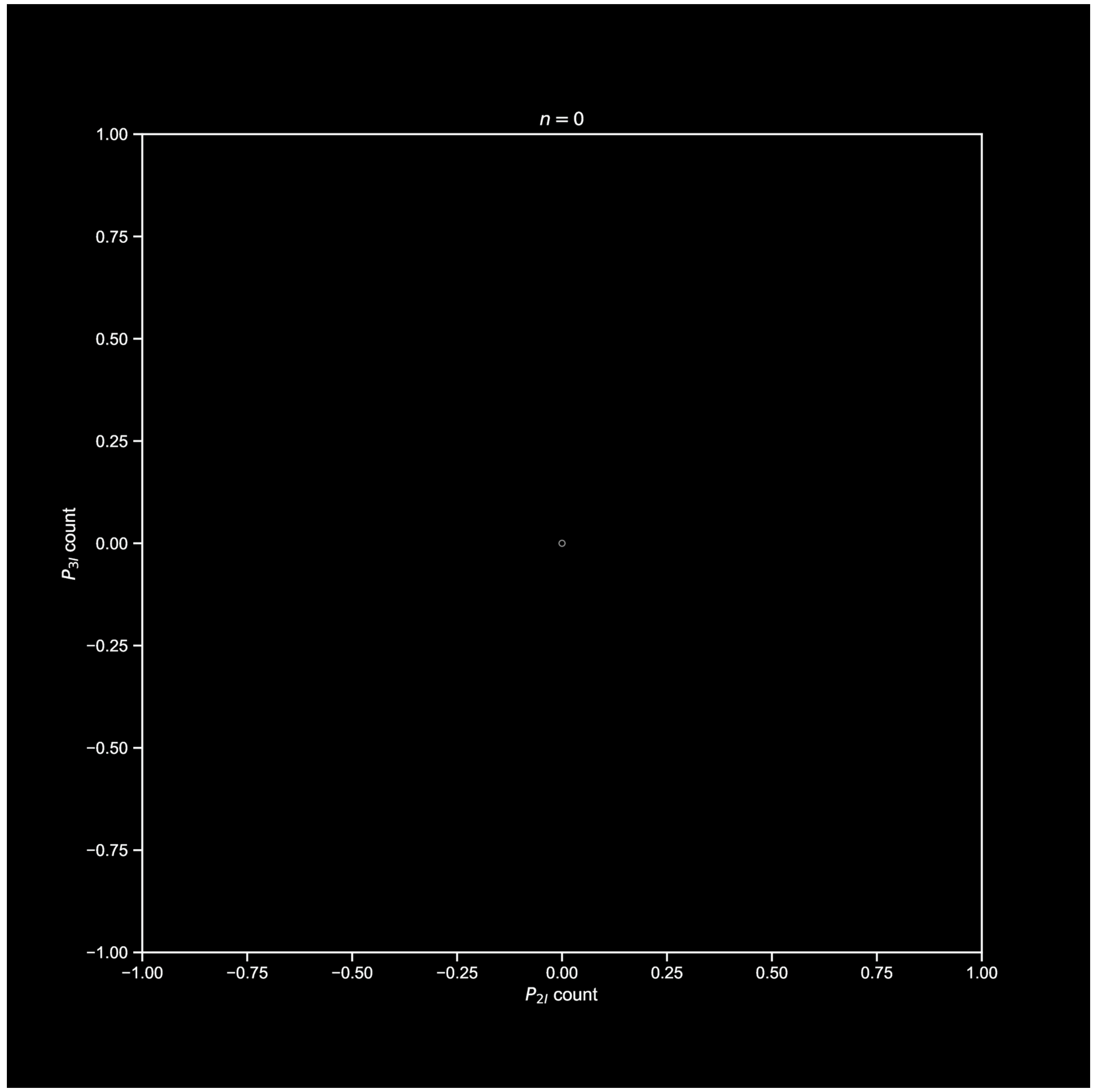
$$\begin{aligned}
\tfrac{\partial}{\partial\lambda}\mathscr{G}(\lambda;\underline{\omega}) &= \langle| \left(e^{ad_{\underline{\omega}\cdot\underline{\hat{O}}}}(\hat{G})\right) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}} ||\rangle \overset{(*)}{=} \langle| \left(e^{ad_{\nu\hat{O}_{P3}}}\left(e^{ad_{\mu\hat{O}_{P2}}}\left(e^{ad_{\varepsilon\hat{O}_E+\gamma\hat{O}_{P1}}}(\hat{G})\right)\right)\right) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}} ||\rangle \\
&= e^{2\varepsilon+\gamma}\langle| \left(e^{ad_{\nu\hat{O}_{P3}}}\left(e^{ad_{\mu\hat{O}_{P2}}}(\hat{G})\right)\right) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}} ||\rangle \\
&= e^{2\varepsilon+\gamma}\langle| \left(e^{ad_{\nu\hat{O}_{P3}}}\left(\hat{G} + (e^{\mu}-1)[\hat{O}_{P2},\hat{G}]\right)\right) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}} ||\rangle \\
&= e^{2\varepsilon+\gamma}\langle| \big(\hat{G} + (e^{\mu}-1)[\hat{O}_{P2},\hat{G}] \\
&\qquad + e^{\mu}(e^{\nu}-1)[\hat{O}_{P3},\hat{G}] + (e^{\nu}-1)(e^{\mu}-e^{-\nu})\hat{R}_{P3'}\big) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}} ||\rangle \\
&= e^{2\varepsilon+\gamma}\langle| \big(2\hat{O}_E + 3(e^{\mu}-1)\hat{O}_{P1} + (4e^{\mu+\nu}-6e^{\mu}+2)\hat{O}_{P2} \\
&\qquad + (3e^{\mu}+e^{-\nu}-3e^{\mu+\nu}-1)\hat{O}_{P3}\big) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}} ||\rangle \\
&= e^{2\varepsilon+\gamma}\langle| \big(2\tfrac{\partial}{\partial\varepsilon} + 3(e^{\mu}-1)\tfrac{\partial}{\partial\gamma} + (4e^{\mu+\nu}-6e^{\mu}+2)\tfrac{\partial}{\partial\mu} \\
&\qquad + (3e^{\mu}+e^{-\nu}-3e^{\mu+\nu}-1)\tfrac{\partial}{\partial\nu}\big) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}} ||\rangle
\end{aligned}$$



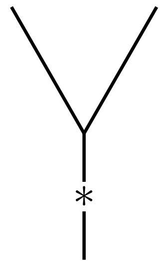
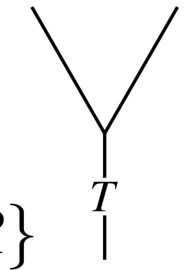
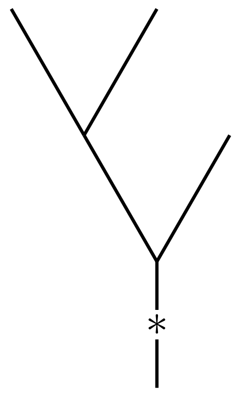
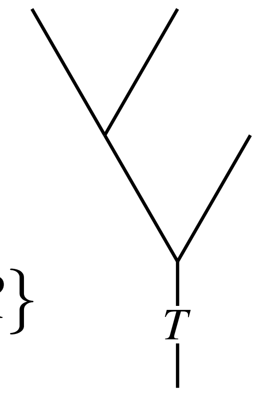
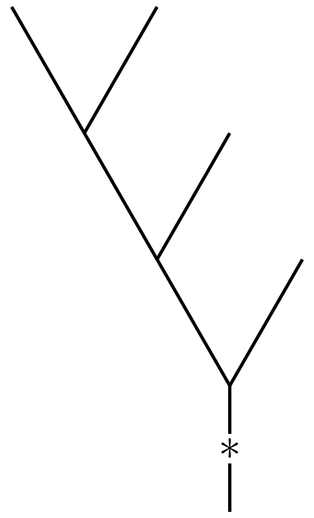
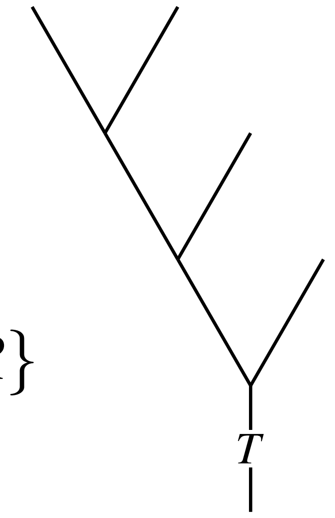

$$[\hat{O}_{P2},[\hat{O}_{P2},\hat{G}]] = [\hat{O}_{P2},\hat{G}]\,, \quad [\hat{O}_{P2},[\hat{O}_{P3},\hat{G}]] = [\hat{O}_{P3},\hat{G}] + \hat{R}_{P3}$$

$$[\hat{O}_{P3},[\hat{O}_{P3},\hat{G}]] = [\hat{O}_{P3},\hat{G}] + 2\hat{R}_{P3'}\,, \quad [\hat{O}_{P2},\hat{R}_{P3'}] = 0\,, \quad [\hat{O}_{P3},\hat{R}_{P3'}] = -\hat{R}_{P3'}$$

$$\langle|\,[\hat{O}_{P2},\hat{G}] = \langle|\,(3\hat{O}_{P1} - 2\hat{O}_{P2})\,, \quad \langle|\,[\hat{O}_{P3},\hat{G}] = \langle|\,(4\hat{O}_{P2} - 3\hat{O}_{P3})\,, \quad \langle|\,\hat{R}_{P3'} = \langle|\,\hat{O}_{P3}$$

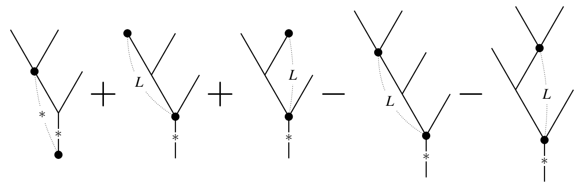
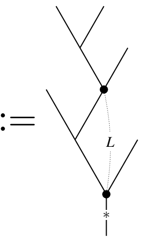


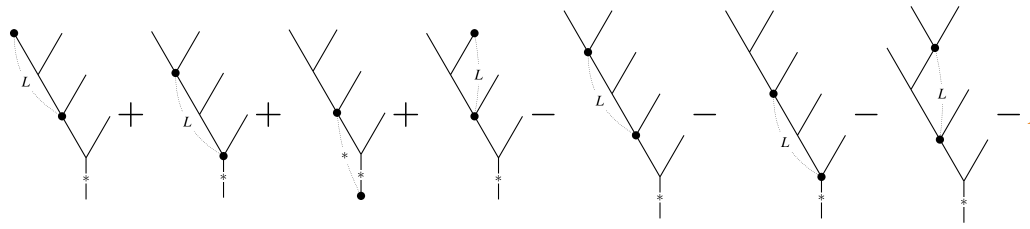
# Example: generating **planar rooted binary trees (PRBTs)** uniformly

$\hat{O}_{P1} :=$  $\equiv \sum_{T\in\{I,L,R\}}$  , $\hat{O}_{P2} :=$  $\equiv \sum_{T\in\{I,L,R\}}$  , $\hat{O}_{P3} :=$  $\equiv \sum_{T\in\{I,L,R\}}$ 

$$\mathscr{G}(\lambda;\underline{\omega}) := \langle| e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}} ||\rangle\,, \quad \underline{\omega}\cdot\underline{\hat{O}} := \varepsilon\hat{O}_E + \gamma\hat{O}_{P1} + \mu\hat{O}_{P2} + \nu\hat{O}_{P3}$$

$$\begin{aligned}
\tfrac{\partial}{\partial\lambda}\mathscr{G}(\lambda;\underline{\omega}) &= \langle| \left(e^{ad_{\underline{\omega}\cdot\underline{\hat{O}}}}(\hat{G})\right) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}} ||\rangle \overset{(*)}{=} \langle| \left(e^{ad_{\nu\hat{O}_{P3}}}\left(e^{ad_{\mu\hat{O}_{P2}}}\left(e^{ad_{\varepsilon\hat{O}_E+\gamma\hat{O}_{P1}}}(\hat{G})\right)\right)\right) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}} ||\rangle \\
&= e^{2\varepsilon+\gamma} \langle| \left(e^{ad_{\nu\hat{O}_{P3}}}\left(e^{ad_{\mu\hat{O}_{P2}}}(\hat{G})\right)\right) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}} ||\rangle \\
&= e^{2\varepsilon+\gamma} \langle| \left(e^{ad_{\nu\hat{O}_{P3}}}\left(\hat{G} + (e^{\mu}-1)[\hat{O}_{P2},\hat{G}]\right)\right) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}} ||\rangle \\
&= e^{2\varepsilon+\gamma} \langle| \big(\hat{G} + (e^{\mu}-1)[\hat{O}_{P2},\hat{G}] \\
&\qquad + e^{\mu}(e^{\nu}-1)[\hat{O}_{P3},\hat{G}] + (e^{\nu}-1)(e^{\mu}-e^{-\nu})\hat{R}_{P3'}\big) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}} ||\rangle \\
&= e^{2\varepsilon+\gamma} \langle| \big(2\hat{O}_E + 3(e^{\mu}-1)\hat{O}_{P1} + (4e^{\mu+\nu} - 6e^{\mu} + 2)\hat{O}_{P2} \\
&\qquad + (3e^{\mu} + e^{-\nu} - 3e^{\mu+\nu} - 1)\hat{O}_{P3}\big) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}} ||\rangle \\
&= e^{2\varepsilon+\gamma} \langle| \big(2\tfrac{\partial}{\partial\varepsilon} + 3(e^{\mu}-1)\tfrac{\partial}{\partial\gamma} + (4e^{\mu+\nu} - 6e^{\mu} + 2)\tfrac{\partial}{\partial\mu} \\
&\qquad + (3e^{\mu} + e^{-\nu} - 3e^{\mu+\nu} - 1)\tfrac{\partial}{\partial\nu}\big) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}} ||\rangle
\end{aligned}$$

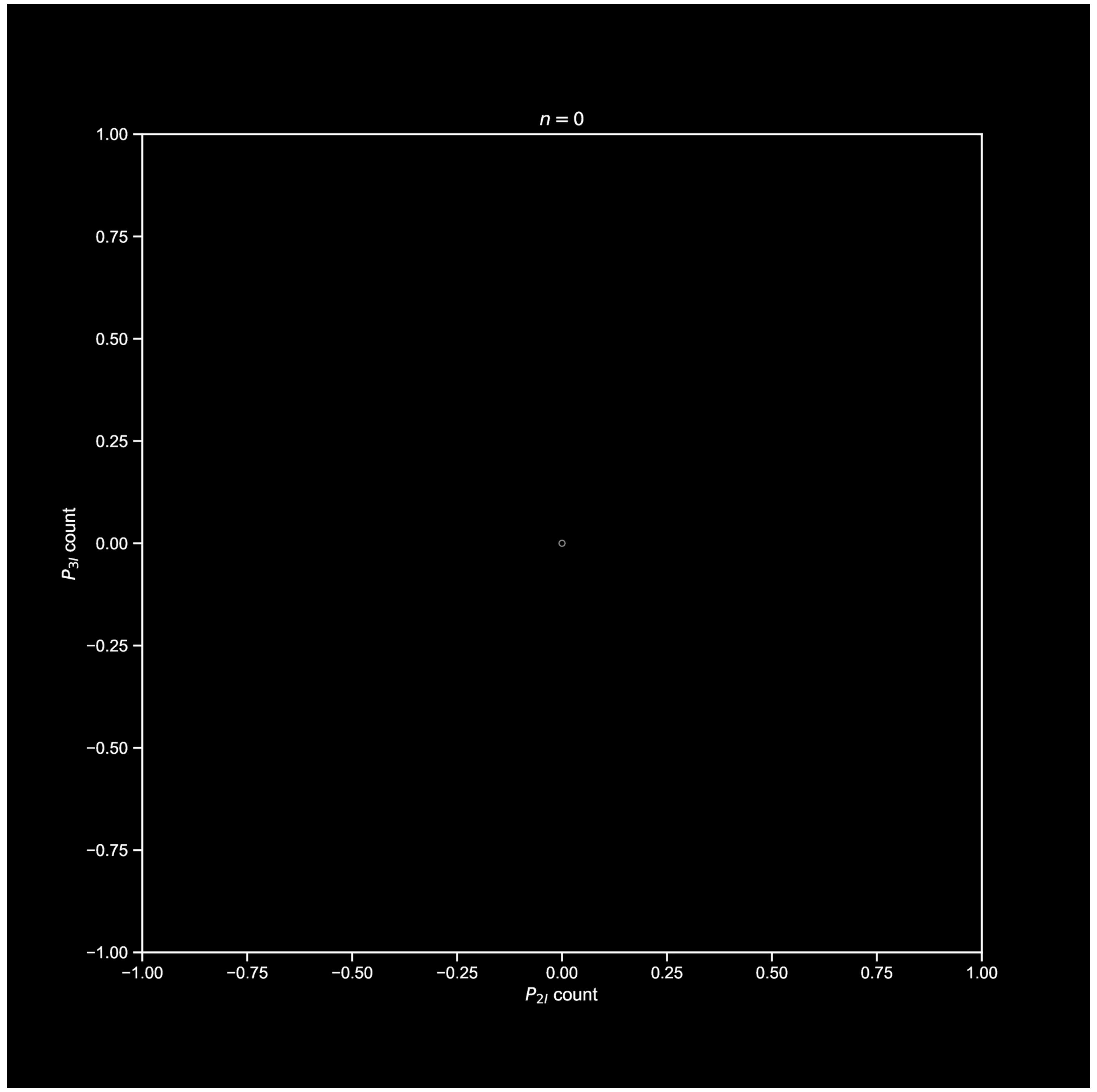
$[\hat{O}_{P2},\hat{G}] =$  $\quad \hat{R}_{P3'} :=$ 

$[\hat{O}_{P3},\hat{G}] =$  $- \hat{R}_{P3'}$

$$[\hat{O}_{P2},[\hat{O}_{P2},\hat{G}]] = [\hat{O}_{P2},\hat{G}]\,, \quad [\hat{O}_{P2},[\hat{O}_{P3},\hat{G}]] = [\hat{O}_{P3},\hat{G}] + \hat{R}_{P3}$$

$$[\hat{O}_{P3},[\hat{O}_{P3},\hat{G}]] = [\hat{O}_{P3},\hat{G}] + 2\hat{R}_{P3'}\,, \quad [\hat{O}_{P2},\hat{R}_{P3'}] = 0\,, \quad [\hat{O}_{P3},\hat{R}_{P3'}] = -\hat{R}_{P3'}$$

$$\langle|[\hat{O}_{P2},\hat{G}] = \langle|(3\hat{O}_{P1} - 2\hat{O}_{P2})\,, \quad \langle|[\hat{O}_{P3},\hat{G}] = \langle|(4\hat{O}_{P2} - 3\hat{O}_{P3})\,, \quad \langle|\hat{R}_{P3'} = \langle|\hat{O}_{P3}$$



# Combinatorial evolution equations

The **formal EMGF evolution equation** for $\mathcal{G}(\lambda;\underline{\omega})$ reads as follows:

$$\tfrac{\partial}{\partial\lambda}\mathcal{G}(\lambda;\underline{\omega}) = \langle| \left(e^{ad_{\underline{\omega}\cdot\underline{\hat{O}}}}\hat{G}\right) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}}\,|X_0\rangle \qquad (ad_A(B) := AB - BA) \tag{15}$$

Applying the version of the **jump-closure theorem** appropriate for the chosen rewriting semantics (DPO or SqPO), the above formal evolution equation may be converted into a proper **evolution equation on formal power series** if the following **polynomial jump-closure** holds:

$$(\mathsf{PJC}') \quad \forall q \in \mathbb{Z}_{\geq 0} : \exists \underline{N(n)} \in \mathbb{Z}^m_{\geq 0}, \gamma_q(\underline{\omega}, \underline{k}) \in \mathbb{R} : \ \langle| \, ad^{\circ\, q}_{\underline{\omega}\cdot\underline{\hat{O}}}(\hat{G}) = \sum_{\underline{k}=\underline{0}}^{\underline{N(q)}} \gamma_{\underline{k}}(\underline{\omega}, \underline{k}) \, \langle| \, \underline{\hat{O}}^{\underline{k}} \tag{16}$$

If a given set of observables satisfies $(\mathsf{PJC}')$, the **formal evolution equation** (12) for the EMGF $\mathcal{G}(\lambda;\underline{\omega})$ may be refined into

$$\tfrac{\partial}{\partial\lambda}\mathcal{G}(\lambda;\underline{\omega}) = \mathbb{G}(\underline{\omega}, \underline{\partial\omega})\mathcal{G}(\lambda;\underline{\omega})\,, \quad \mathbb{G}(\underline{\omega}, \underline{\partial\omega}) = \left(\langle| \, e^{ad_{\underline{\omega}\cdot\underline{\hat{O}}}}(\hat{G})\right)\Big|_{\underline{\hat{O}}\mapsto\underline{\partial\omega}}\,. \tag{17}$$

# Combinatorial evolution equations

The **formal EMGF evolution equation** for $\mathcal{G}(\lambda;\underline{\omega})$ reads as follows:

$$\frac{\partial}{\partial\lambda}\mathcal{G}(\lambda;\underline{\omega}) = \langle | \left(e^{ad_{\underline{\omega}\cdot\underline{\hat{O}}}}\hat{G}\right) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}} |X_0\rangle \qquad (ad_A(B) := AB - BA) \tag{15}$$

Applying the version of the **jump-closure theorem** appropriate for the chosen rewriting semantics (DPO or SqPO), the above formal evolution equation may be converted into a proper **evolution equation on formal power series** if the following **polynomial jump-closure** holds:

$$(\mathrm{PJC}') \quad \forall q \in \mathbb{Z}_{\geq 0} : \exists \underline{N(n)} \in \mathbb{Z}^m_{\geq 0}, \gamma_q(\underline{\omega},\underline{k}) \in \mathbb{R} : \ \langle | \, ad^{\circ q}_{\underline{\omega}\cdot\underline{\hat{O}}}(\hat{G}) = \sum_{\underline{k}=\underline{0}}^{\underline{N(q)}} \gamma_{\underline{k}}(\underline{\omega},\underline{k}) \, \langle | \, \underline{\hat{O}}^{\underline{k}} \tag{16}$$

If a given set of observables satisfies $(\mathrm{PJC}')$, the **formal evolution equation** (12) for the EMGF $\mathcal{G}(\lambda;\underline{\omega})$ may be refined into

$$\frac{\partial}{\partial\lambda}\mathcal{G}(\lambda;\underline{\omega}) = \mathbb{G}(\underline{\omega},\underline{\partial\omega})\mathcal{G}(\lambda;\underline{\omega}) \,, \quad \mathbb{G}(\underline{\omega},\underline{\partial\omega}) = \left(\langle | \, e^{ad_{\underline{\omega}\cdot\underline{\hat{O}}}}(\hat{G})\right)\Big|_{\underline{\hat{O}}\mapsto\underline{\partial\omega}} \,. \tag{17}$$

# Plan of the talk:

I. A Case Study in Applied Category Theory: from Categorical Rewriting to Rule-algebraic Combinatorics

II. The `coreact.wiki` Initiative

# CoREACT

## Coq-based Rewriting: towards Executable Applied Category Theory

Consortium: IRIF (UP), LIP (ENS-Lyon), LIX (École Polytechnique), Sophia-Antipolis (Inria)

| Partner | Last name | First name |
|---|---|---|
| Université de Paris | BEHR | Nicolas |
| | GALLEGO | Emilio |
| | GHEERBRANT | Amélie |
| | HERBELIN | Hugo |
| | MELLIÈS | Paul-André |
| | ROGOVA | Alexandra |
| ENS-Lyon | HARMER | Russell |
| | HIRSCHOWITZ | Tom |
| | POUS | Damien |
| École Polytechnique | MIMRAM | Samuel |
| | WERNER | Benjamin |
| | ZEILBERGER | Noam |
| Inria Sophia-Antipolis | BERTOT | Yves |
| | COHEN | Cyril |
| | TASSI | Enrico |


coreact.wiki

# Main objectives of the CoREACT/GReTA ExACT initiative

- Development of a methodology for **diagrammatic reasoning in Coq**
- **Formalization** (in Coq) and **certification** of a representative collection of axioms and theorems for **compositional categorical rewriting theory**
- Development of a **Coq-enabled interactive database and wiki system**
- Development of a CoREACT wiki-based **"proof-by-pointing" engine**
- Executable **reference prototype algorithms** from categorical structures in Coq (via the use of SMT solvers/theorem provers such as Z3)

# Main objectives of the CoREACT/GReTA ExACT initiative

- Development of a methodology for **diagrammatic reasoning in Coq**
- **Formalization** (in Coq) and **certification** of a representative collection of axioms and theorems for **compositional categorical rewriting theory**
- Development of a **Coq-enabled interactive database and wiki system**
- Development of a CoREACT wiki-based **“proof-by-pointing” engine**
- Executable **reference prototype algorithms** from categorical structures in Coq (via the use of SMT solvers/theorem provers such as Z3)

# A (very non-exhaustive!) view on **wiki systems in mathematics/ (A)CT**

nLab

Home Page All Pages

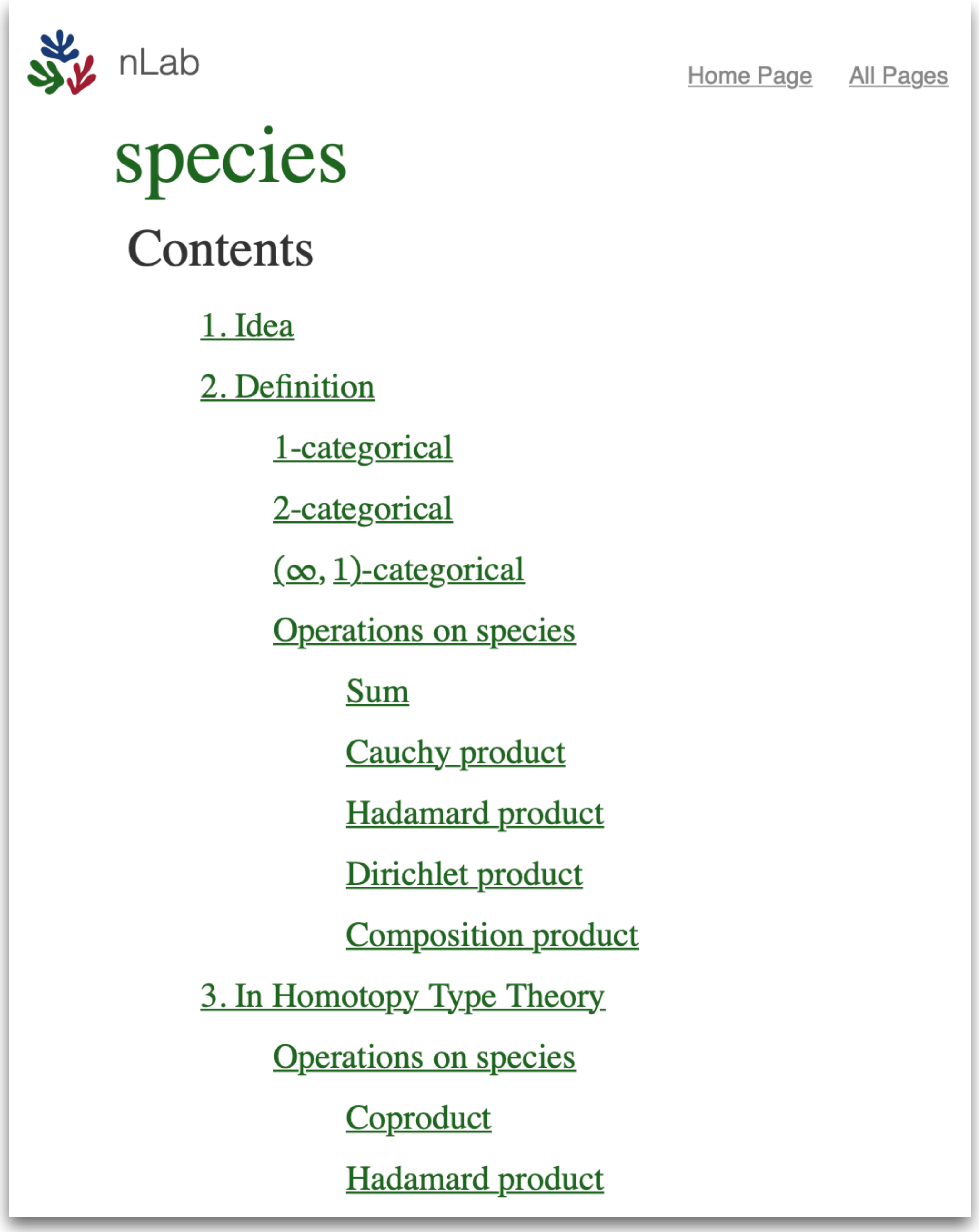
## species

### Contents

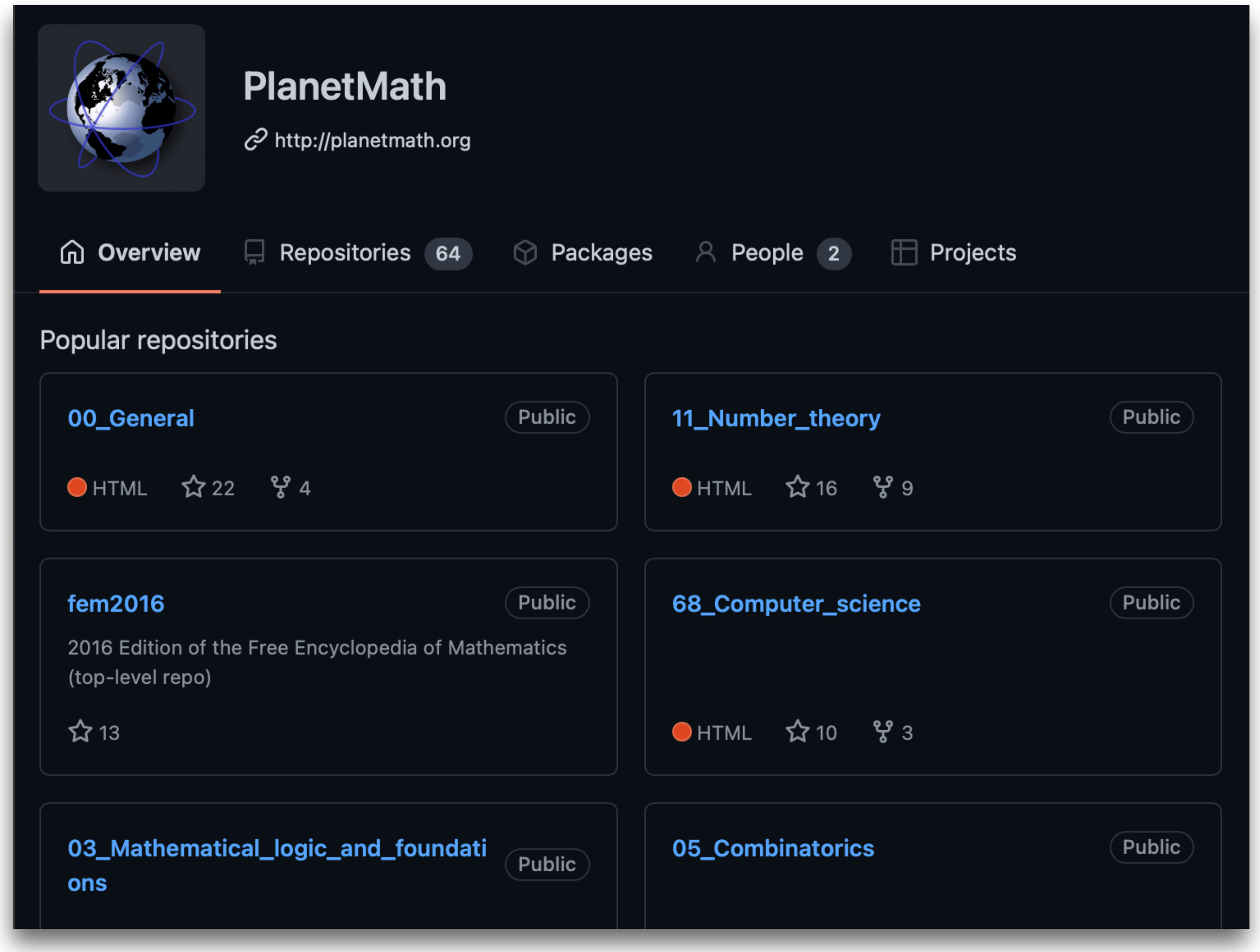


`https://ncatlab.org`

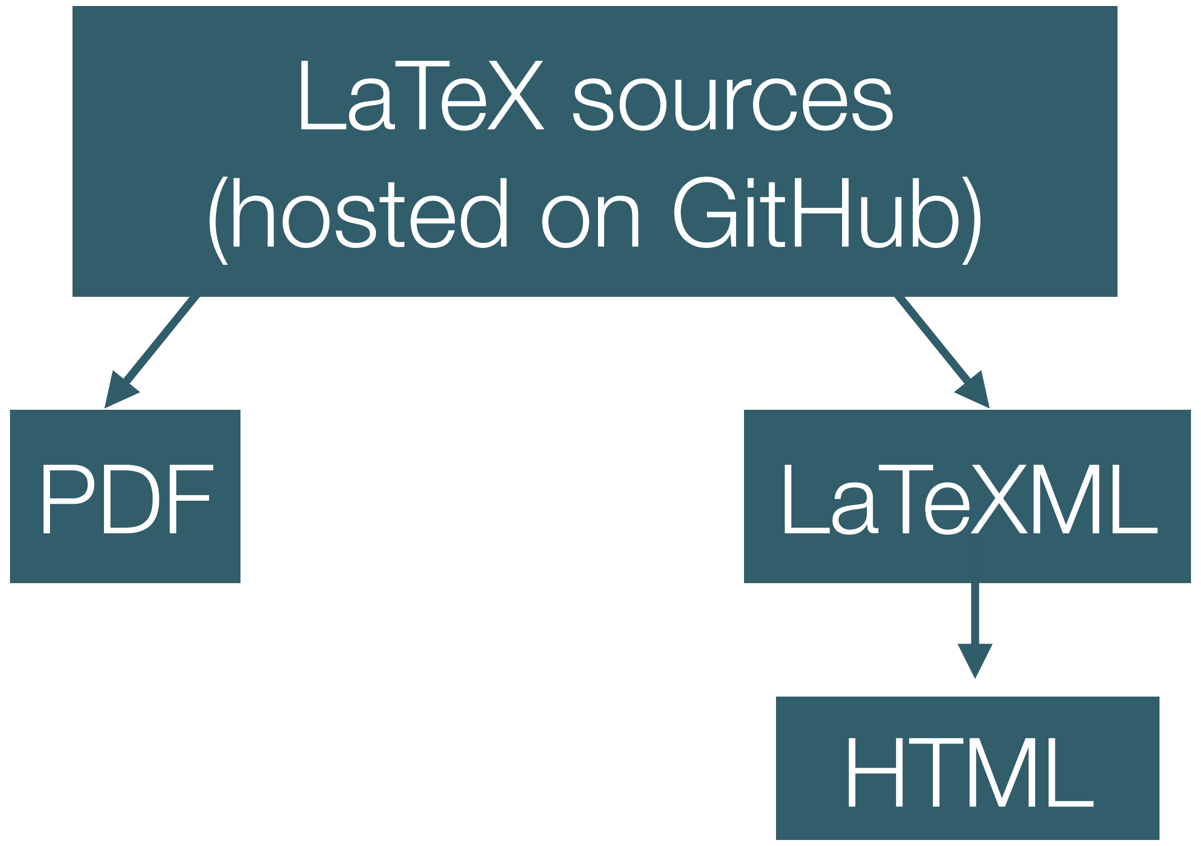
# A (very non-exhaustive!) view on **wiki systems in mathematics/ (A)CT**




`https://ncatlab.org`

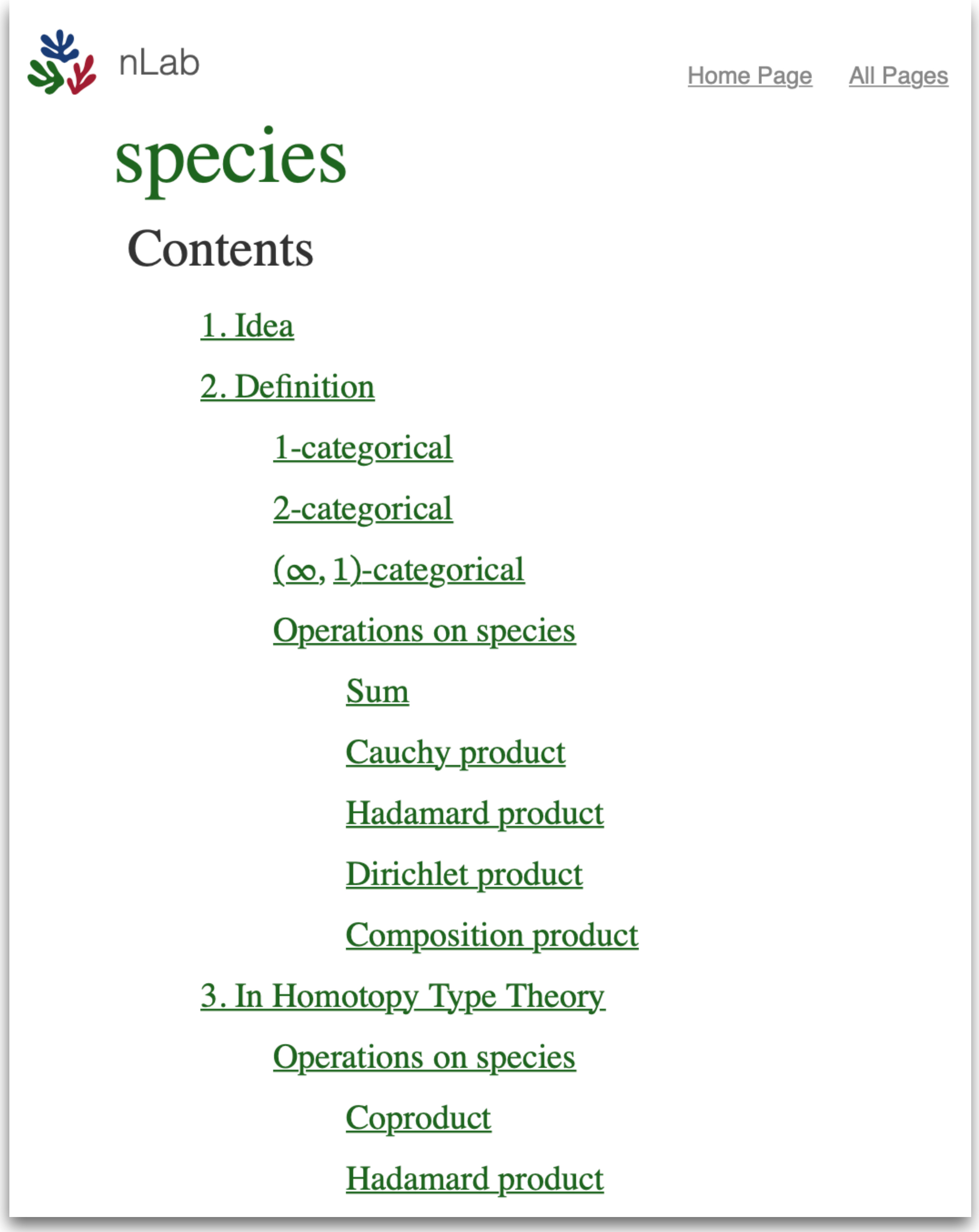



`https://planetmath.org`

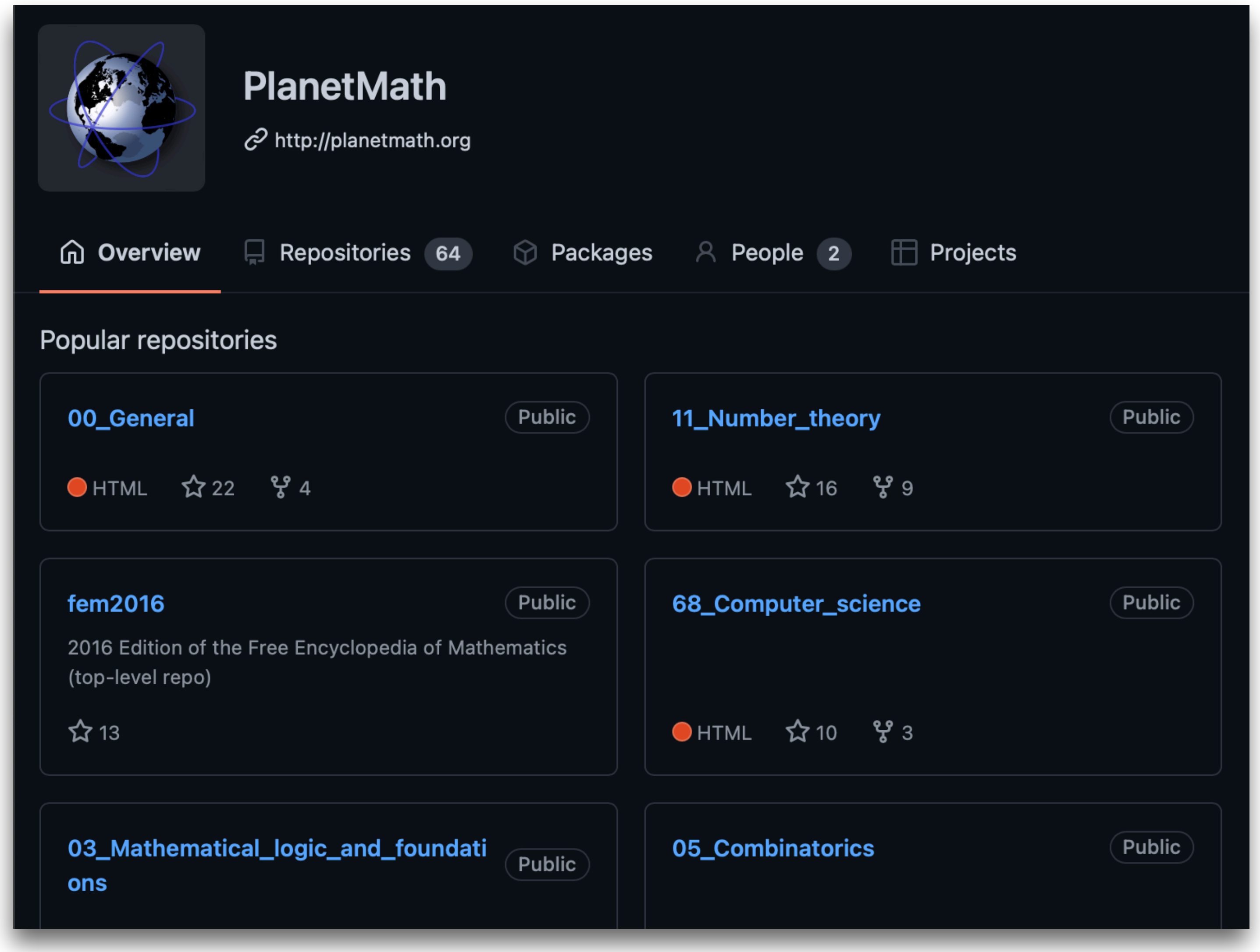



**(semi-) automatic cross-linking**
provided via `NNexus` system

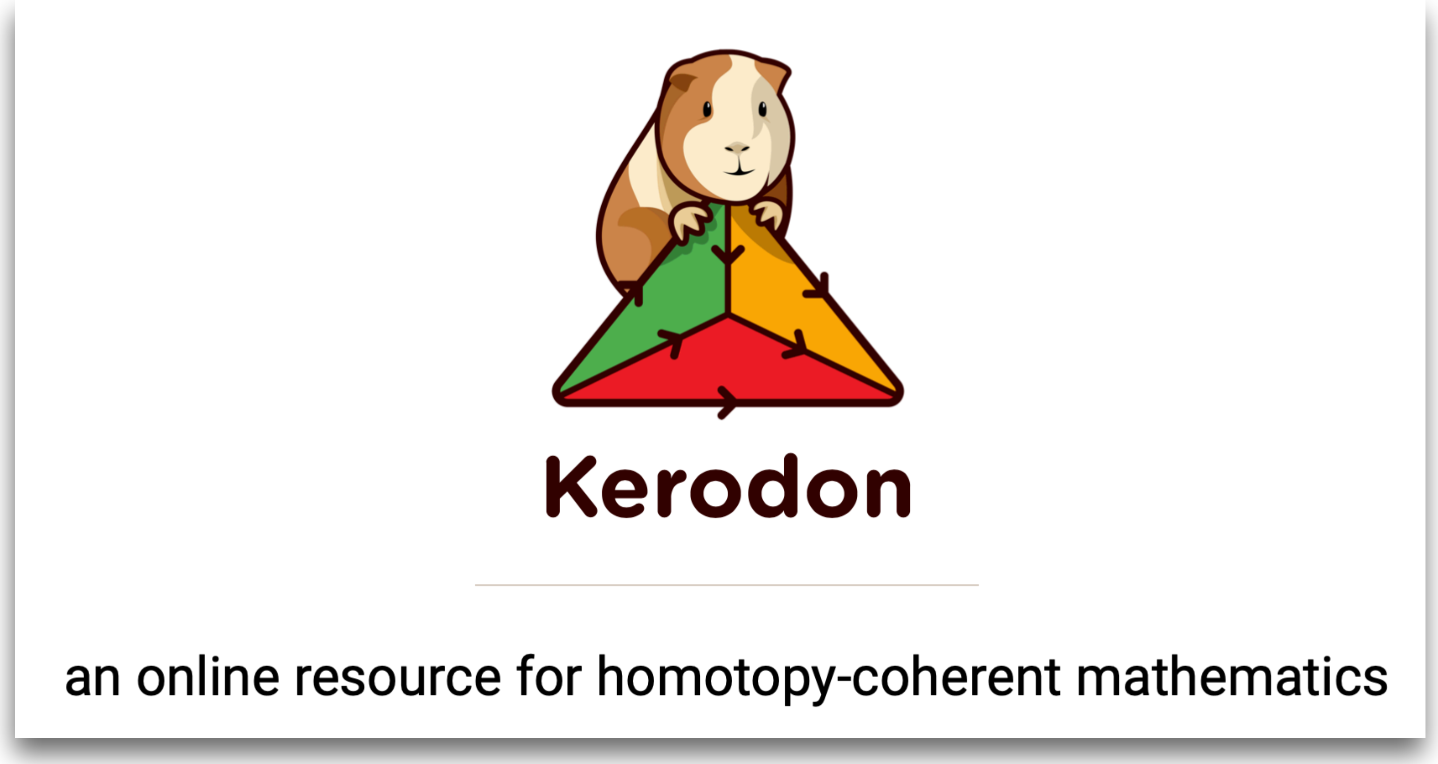
# A (very non-exhaustive!) view on **wiki systems in mathematics/ (A)CT**



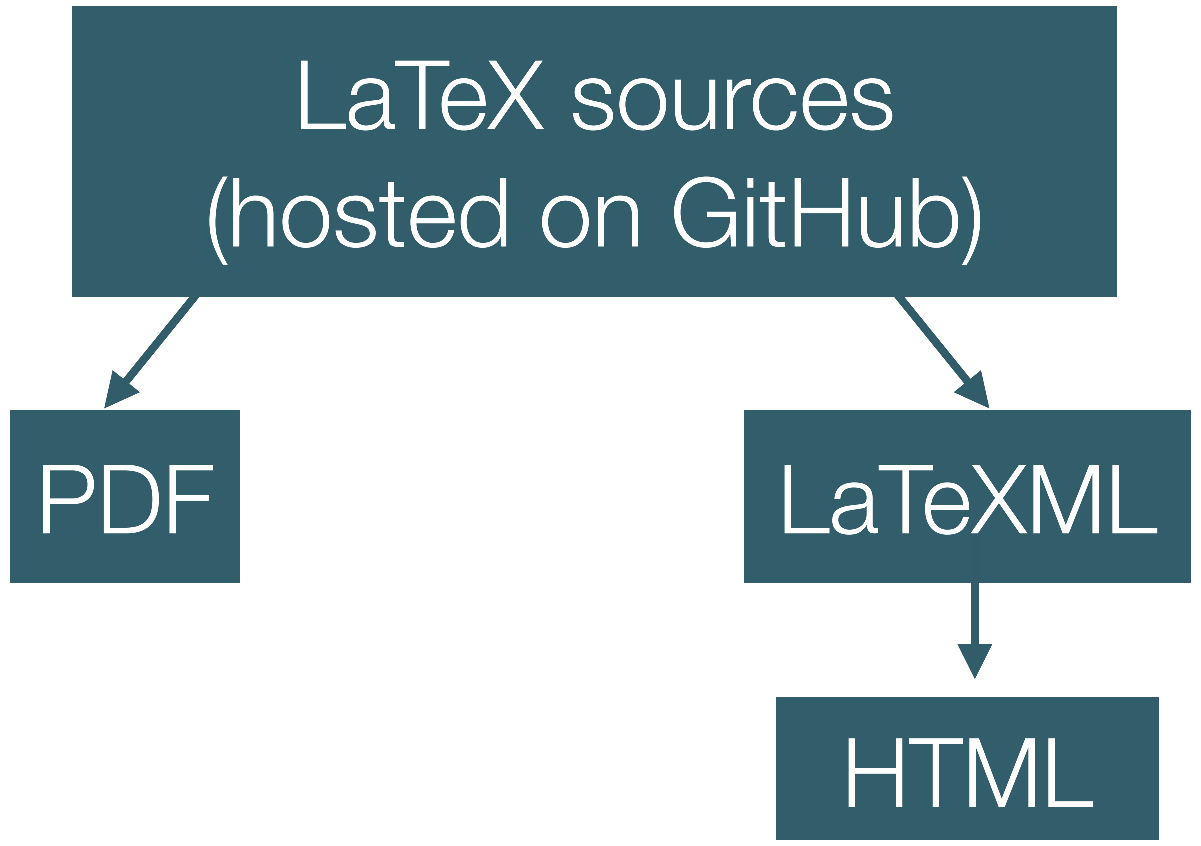

https://ncatlab.org




https://planetmath.org

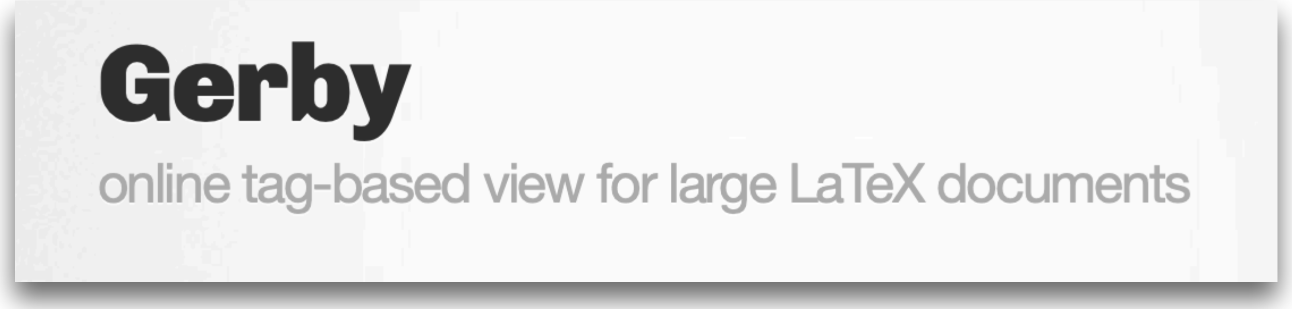



https://kerodon.net




**(semi-) automatic cross-linking** provided via `NNexus` system

- J. Lurie's **online textbook** on categorical homotopy theory
- technology based upon **online tags view** via the `Gerby` system




https://gerby-project.github.io

# A (very non-exhaustive!) view on **proof assistants in mathematics**

Agda

https://github.com/agda

**Hu Jason**, CPP21: “Formalizing Category Theory in Agda”

https://youtu.be/a2txkoybw2M

The Coq Proof Assistant

https://coq.inria.fr

EPIT Spring School on HoTT: **Bas Spitters** Part 1 (Introduction to Coq and HoTT)

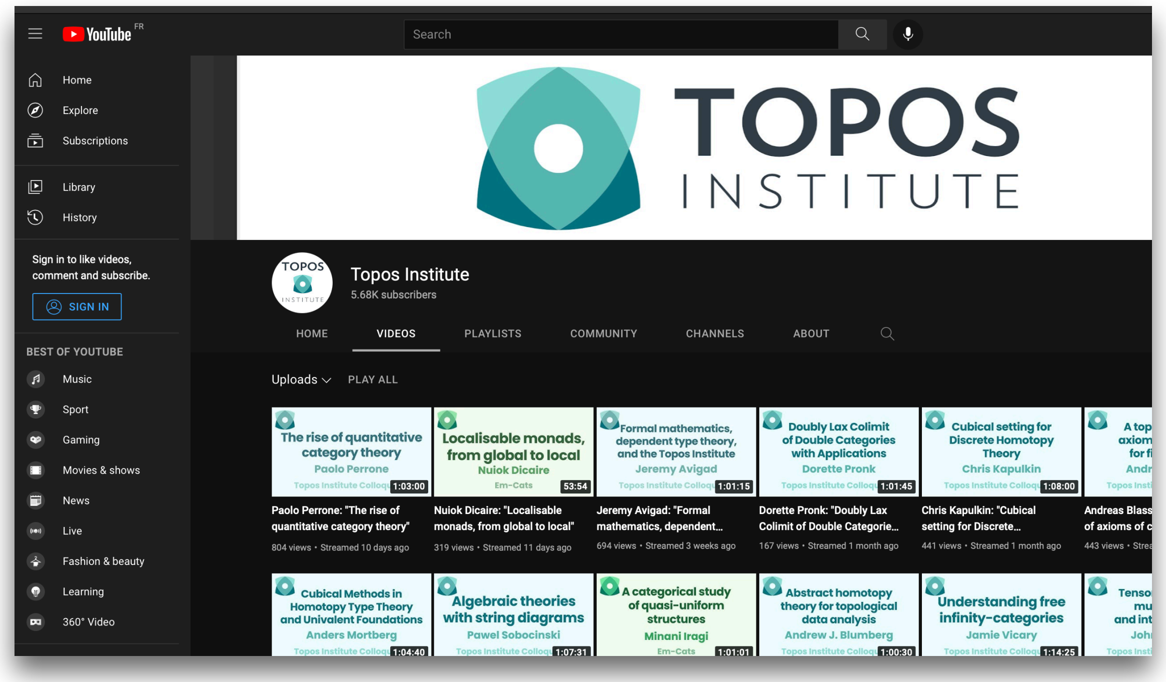
https://youtu.be/k8T9L0qR38o

LEAN

THEOREM PROVER

Microsoft Research

https://leanprover.github.io




**Jeremy Avigad**: "Formal mathematics, dependent type theory, and the Topos Institute"

https://youtu.be/Kpa8cCUZLms

**Kevin Buzzard**: "What is the point of Lean's maths library?"

https://youtu.be/alByz_LoANE

# A (very non-exhaustive!) view on **proof assistants in mathematics**

Agda

https://github.com/agda

**Hu Jason**, CPP21: “Formalizing Category Theory in Agda”

https://youtu.be/a2txkoybw2M

The Coq Proof Assistant

https://coq.inria.fr

EPIT Spring School on HoTT: **Bas Spitters** Part 1 (Introduction to Coq and HoTT)

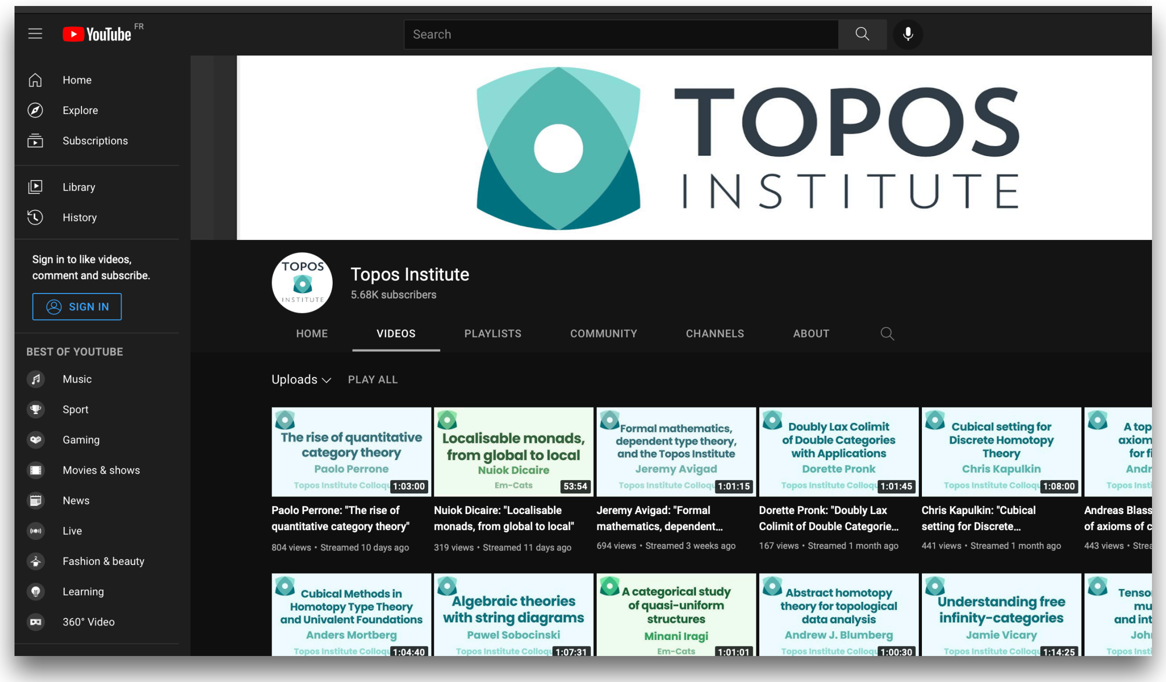
https://youtu.be/k8T9L0qR38o

LEAN

THEOREM PROVER

Microsoft Research

https://leanprover.github.io




**Jeremy Avigad**: "Formal mathematics, dependent type theory, and the Topos Institute"

https://youtu.be/Kpa8cCUZLms

**Kevin Buzzard**: "What is the point of Lean's maths library?"

https://youtu.be/alByz_LoANE

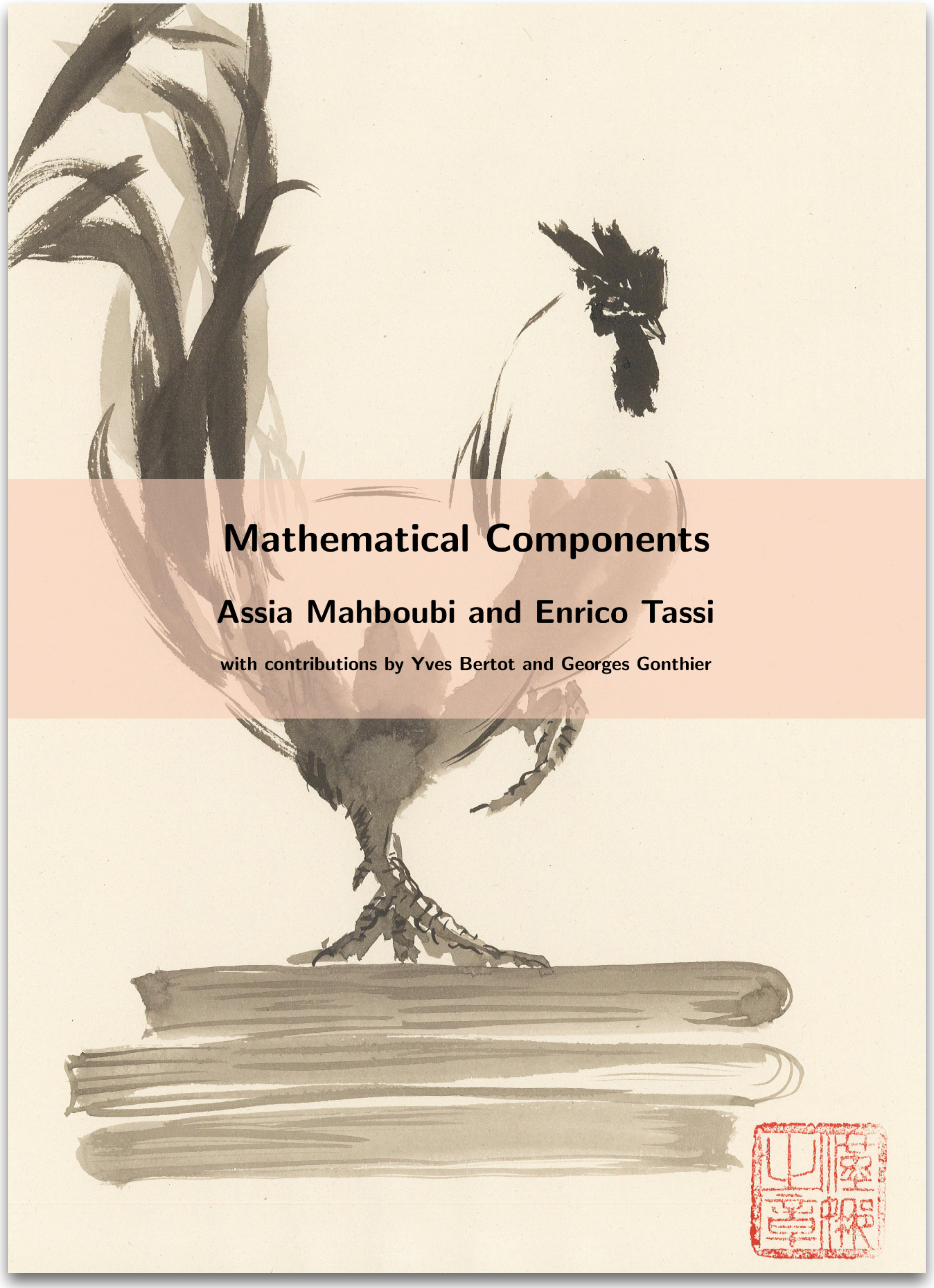
# Coq — overview

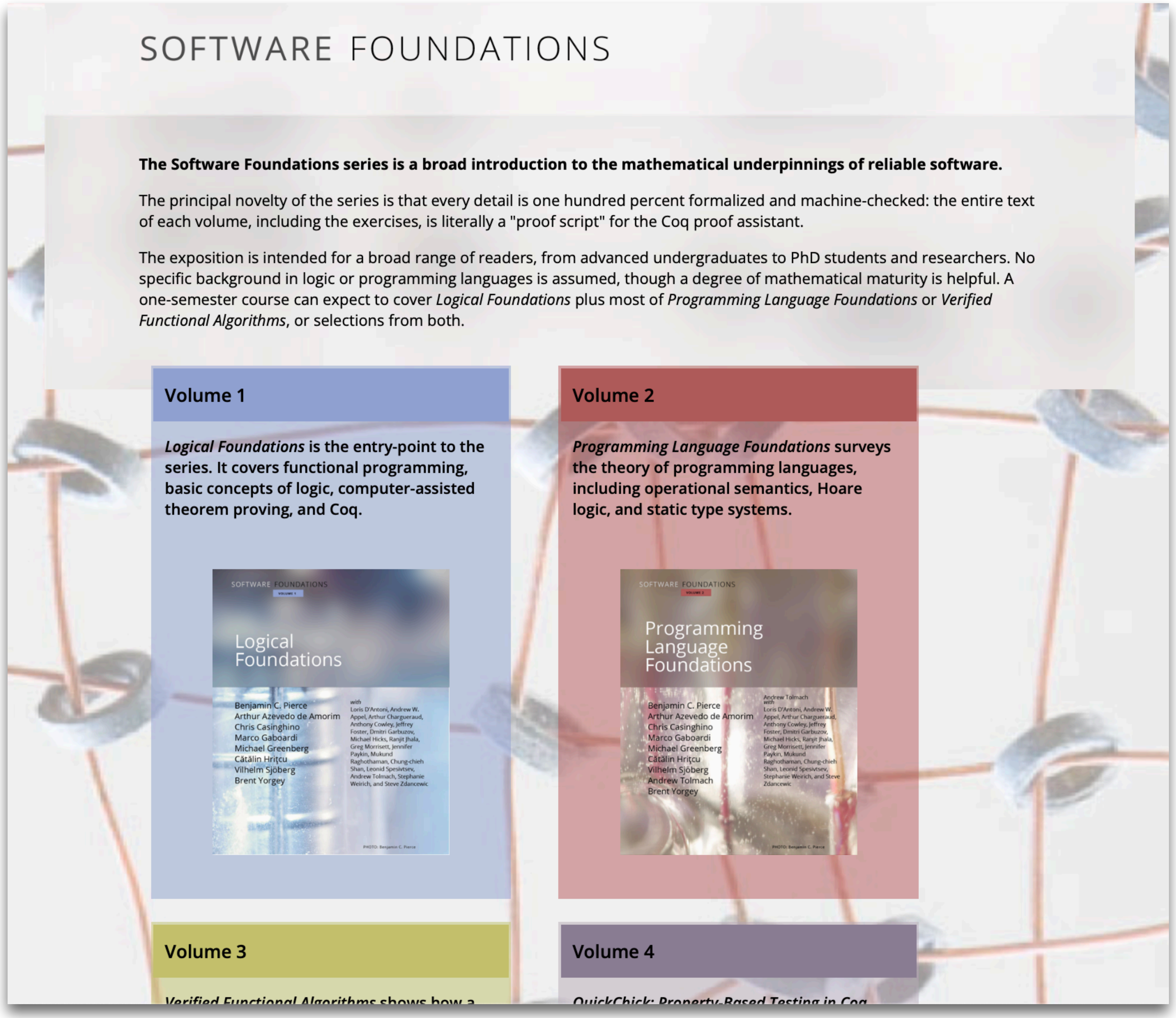
The Coq Proof Assistant

`https://coq.inria.fr`

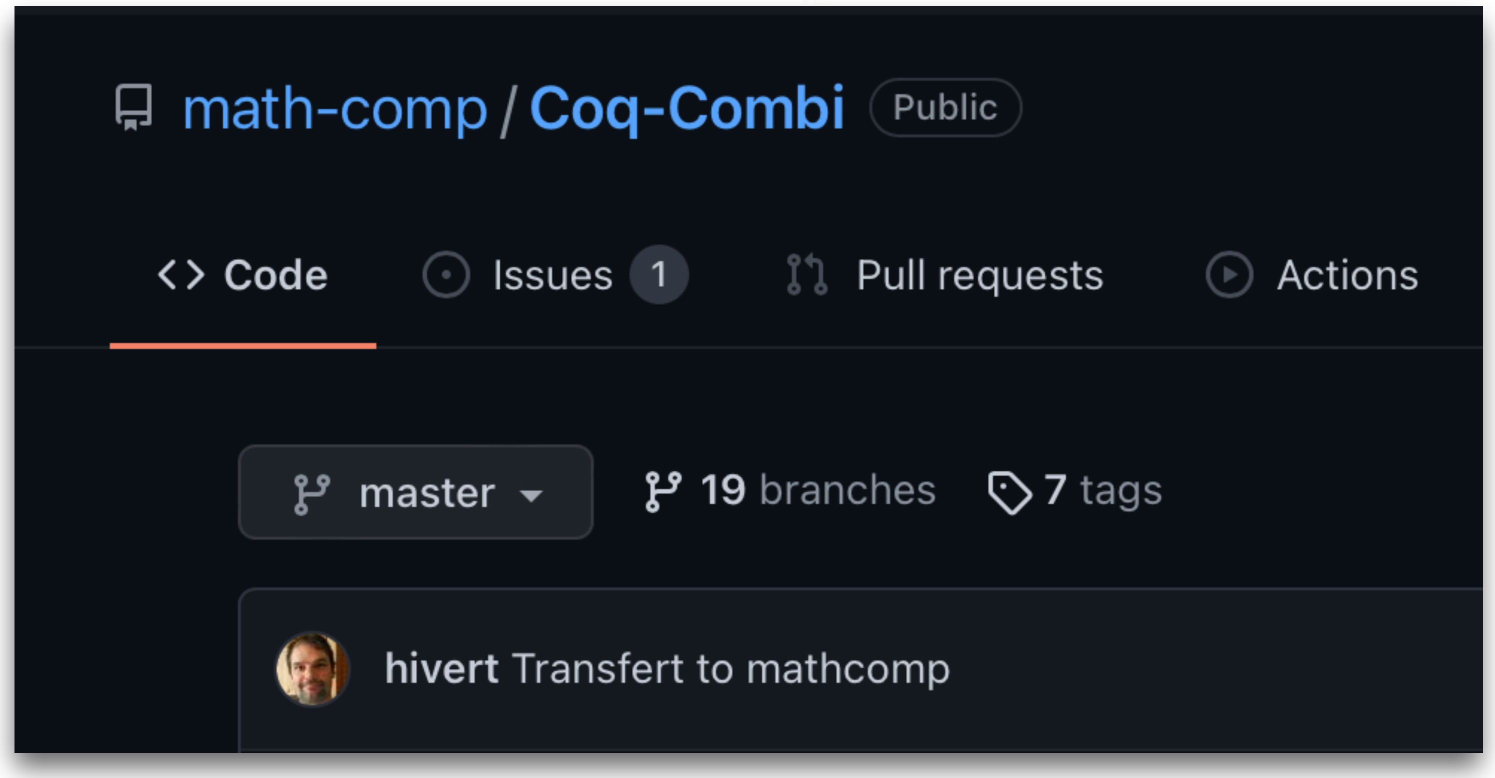
- **1984** implementation of the **Calculus of Constructions** at INRIA-Rocquencourt by **Thierry Coquand** and **Gérard Huet**
- **1991 Calculus of Inductive Constructions (CIC)** by **Christine Paulin-Mohring**
- **2002** completion of the **four color theorem proof in Coq** by **Georges Gonthier** and **Benjamin Werner** (start of the `SSReflect` development)
- **(…)**
- **2012** completion of the **Feit–Thompson theorem** by **Georges Gonthier et al.**
- **(…)**
- more than **200 people** contributed over the past >30 years (most recent stable version: **8.14**)

# Coq — famous milestones

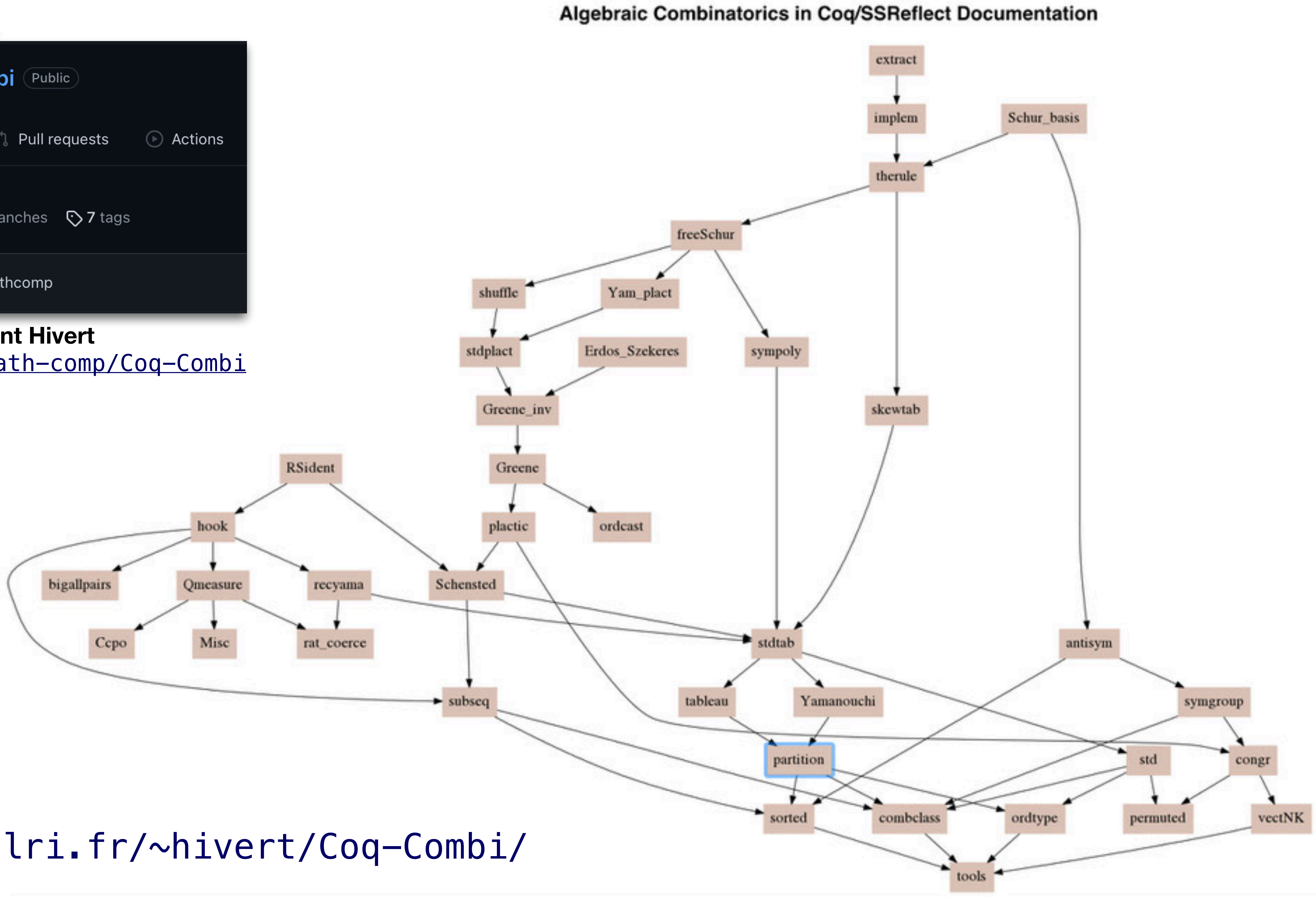


# Coq-combi

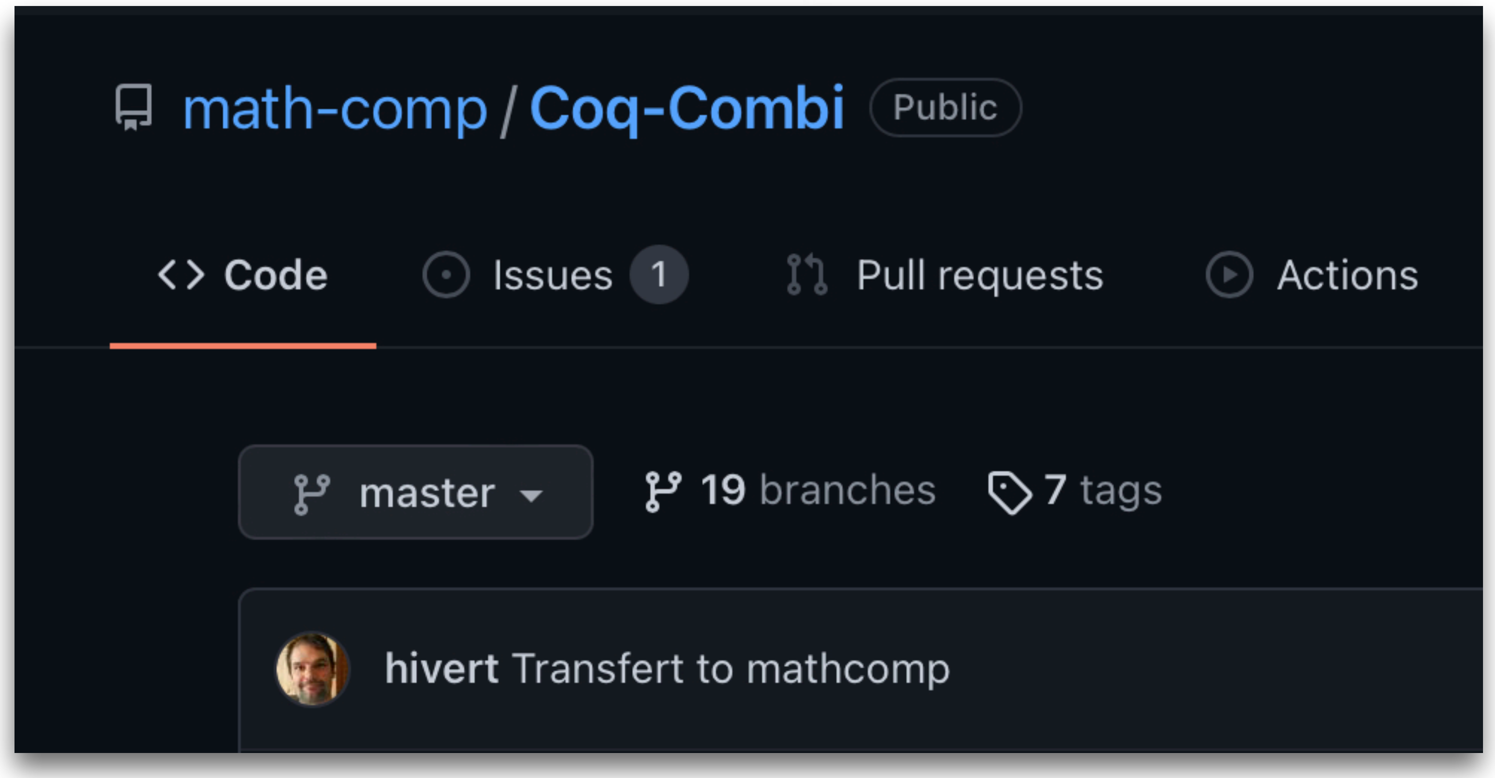



Author: **Florent Hivert**
https://github.com/math-comp/Coq-Combi

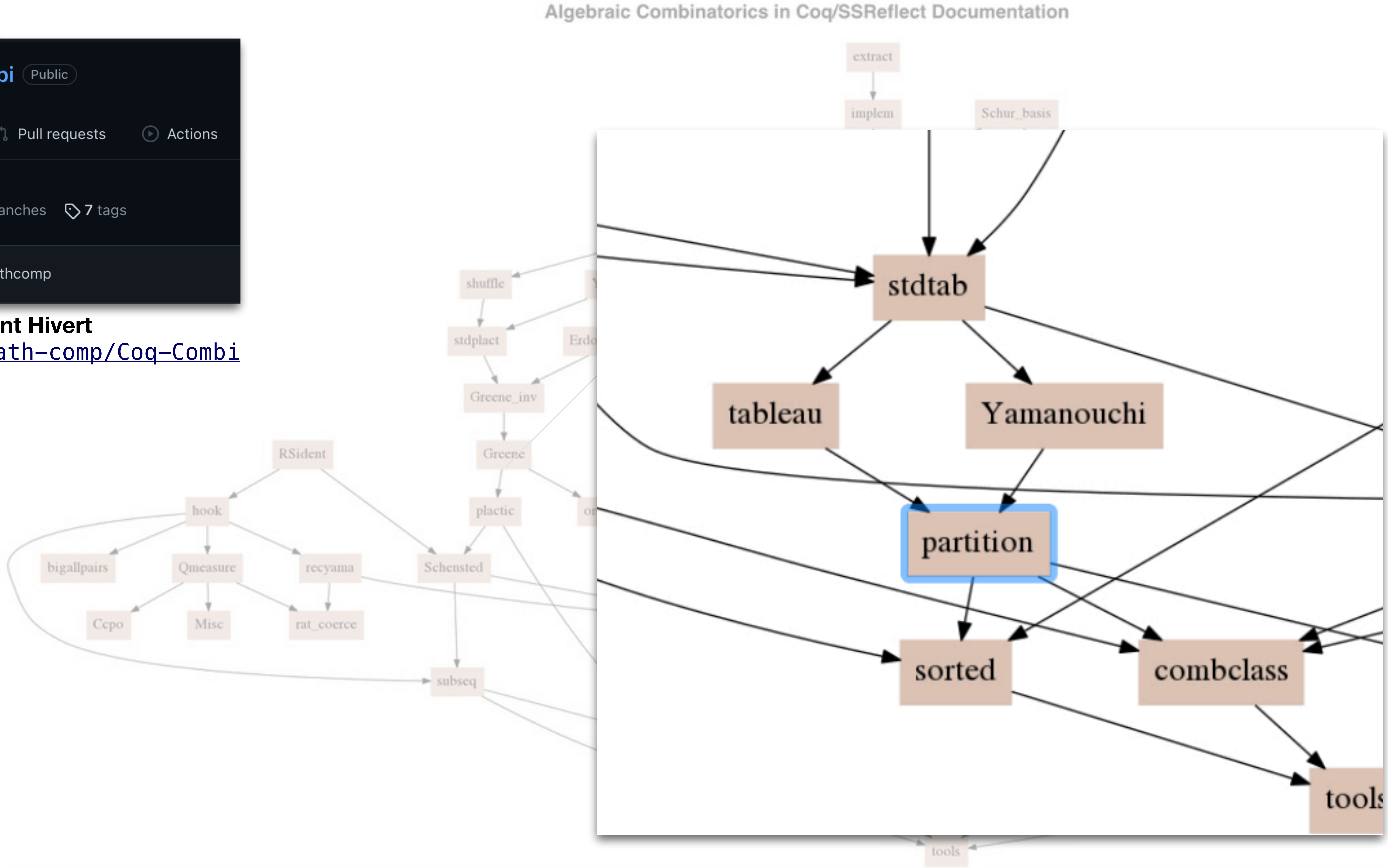



https://www.lri.fr/~hivert/Coq-Combi/

# Coq-combi




Author: **Florent Hivert**
https://github.com/math-comp/Coq-Combi



# Coq-combi

## Combi.Combi.partition: Integer Partitions

- Shapes and Integer Partitions
- Shapes
  - A finite type `finType` for coordinate of boxes inside a shape
  - Rewriting bigops running along the boxes of a shape
  - Adding a box to a shape
- Integer Partitions
  - Definitions and basic properties
  - Corners, adding and removing corners
  - Conjugate of a partition
  - Partial sum of partitions
- Inclusion of Partitions and Skew Partitions
- Sigma Types for Partitions
- Counting functions
- TODO: Generalize and move in finOrdType

# Coq-combi

## Shapes and Integer Partitions

Partitions (and more generally shapes) are stored by terms of type `seq (seq nat)`. We define the following predicates and operations on `seq (seq nat)`: `(r, c)` is in `sh` if `r < sh[i]`

- `is_in_shape sh r c` == the box with coordinate (r, c) belongs to the shape sh, that is: `c < sh[r]`.
- `is_box_in_shape (r, c)` == uncurried version: same as `is_in_shape sh r c`.
- `box_in sh` == a sigma type for boxes in sh : `{ b | is_box_in_shape sh b }` is is canonically a `subFinType`.
- `enum_box_in sh` == a full duplicate free list of the boxes in sh.

Integer Partitions:

- `is_part sh` == `sh` is a partition
- `rem_trail0 sh` == remove the trailing zeroes of a shape
- `is_add_corner sh i` == i is the row of an addable corner of sh
- `is_rem_corner sh i` == i is the row of a removable corner of sh
- `incr_nth sh i` == the shape obtained by adding a box at the end of the i-th row. This gives a partition if i is an addable corner of sh (Lemma `is_part_incr_nth`)
- `decr_nth sh i` == the shape obtained by removing a box at the end of the i-th row. This gives a partition if i is an removable corner of sh

## Sigma Types for Partitions

```
Section PartCombClass.

Structure intpart : Type := IntPart {pval :> seq nat; _ : is_part pval}.
Canonical intpart_subType := Eval hnf in [subType for pval].
Definition intpart_eqMixin := Eval hnf in [eqMixin of intpart by <:].
Canonical intpart_eqType := Eval hnf in EqType intpart intpart_eqMixin.
Definition intpart_choiceMixin := Eval hnf in [choiceMixin of intpart by <:].
Canonical intpart_choiceType := Eval hnf in ChoiceType intpart intpart_choiceMixin.
Definition intpart_countMixin := Eval hnf in [countMixin of intpart by <:].
Canonical intpart_countType := Eval hnf in CountType intpart intpart_countMixin.

Lemma intpartP (p : intpart) : is_part p.

Hint Resolve intpartP.

Canonical conj_intpart p := IntPart (is_part_conj (intpartP p)).

Lemma conj_intpartK : involutive conj_intpart.

Lemma intpart_sum_inj (s t : intpart) :
  (∀ k, part_sum s k = part_sum t k) → s = t.

Fixpoint enum_partnsk sm sz mx : (seq (seq nat)) :=
  if sz is sz.+1 then
    flatten [seq [seq i :: p | p <- enum_partnsk (sm - i) sz i] | i <- iota 1 (minn sm mx)]
  else if sm is sm.+1 then [::] else [:: [::]].
Definition enum_partns sm sz := enum_partnsk sm sz sm.
Definition enum_partn sm := flatten [seq enum_partns sm sz | sz <- iota 0 sm.+1 ].
```

# Major usability concerns (?)

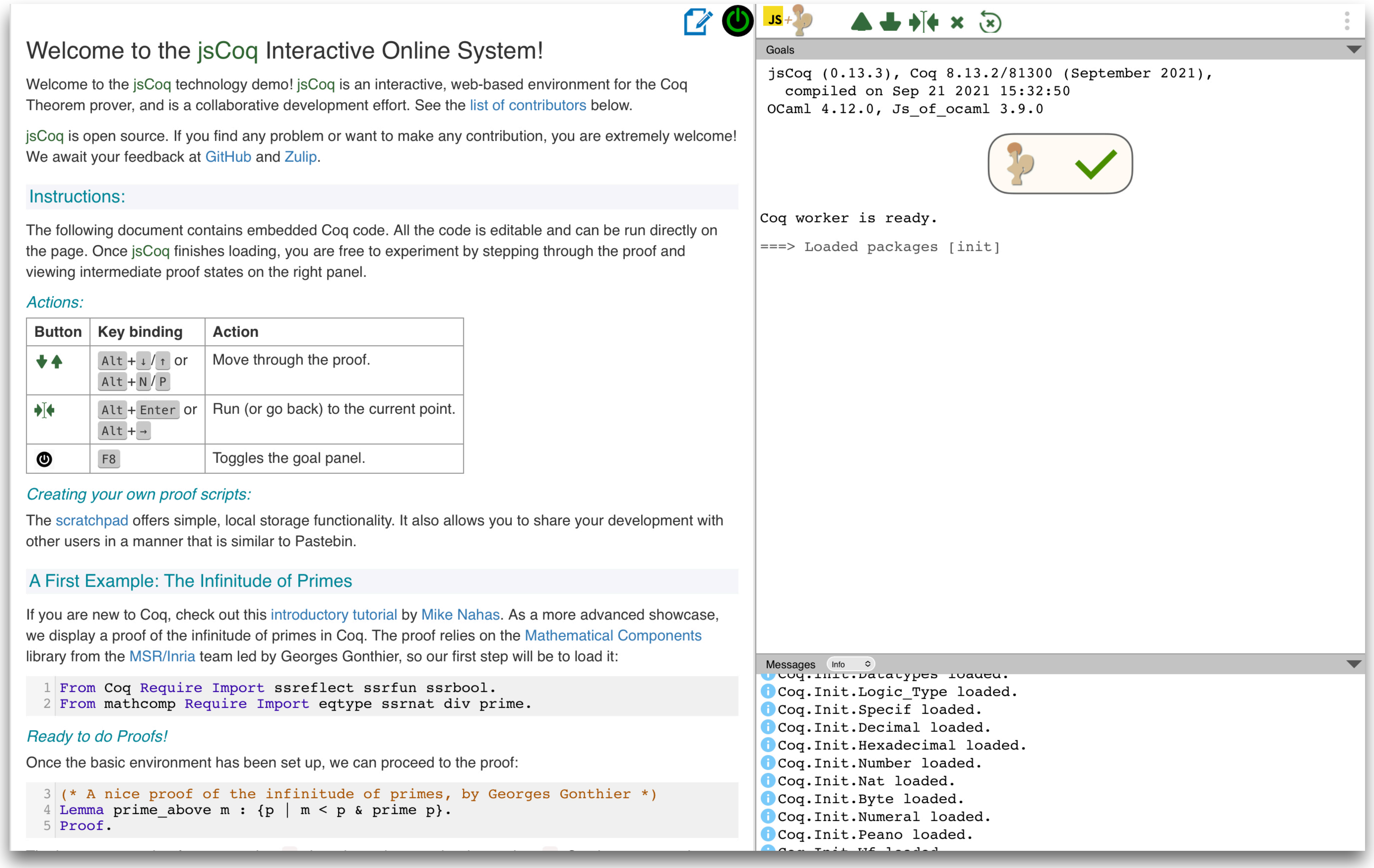
- Difficult and work-intensive to **install/compile from source** on some systems
- As both a **proof assistant** *and* a **programming language**, understanding the theory behind Coq and acquiring a working knowledge of the semantics/technical peculiarities of the Coq system is quite work-intensive
- Finding and analyzing proofs is mostly a **manual** (if assisted) **process** — **standardization** and **searchability**?
- **Curation**, **quality control** and medium- to long-term **maintenance** of collections of proofs is challenging
- **"Burden of interdisciplinarity"** — documenting a given piece of mathematical knowledge in a wiki system requires substantial amounts of **human-readable** and potentially highly technical text, potentially with very involved **mathematical examples** for illustration, while designing corresponding proofs in Coq requires a form of programming which has to be aided by some form of **Coq API** documentation and ideally a library of **code examples**
- **…**

# Major usability concerns (?)

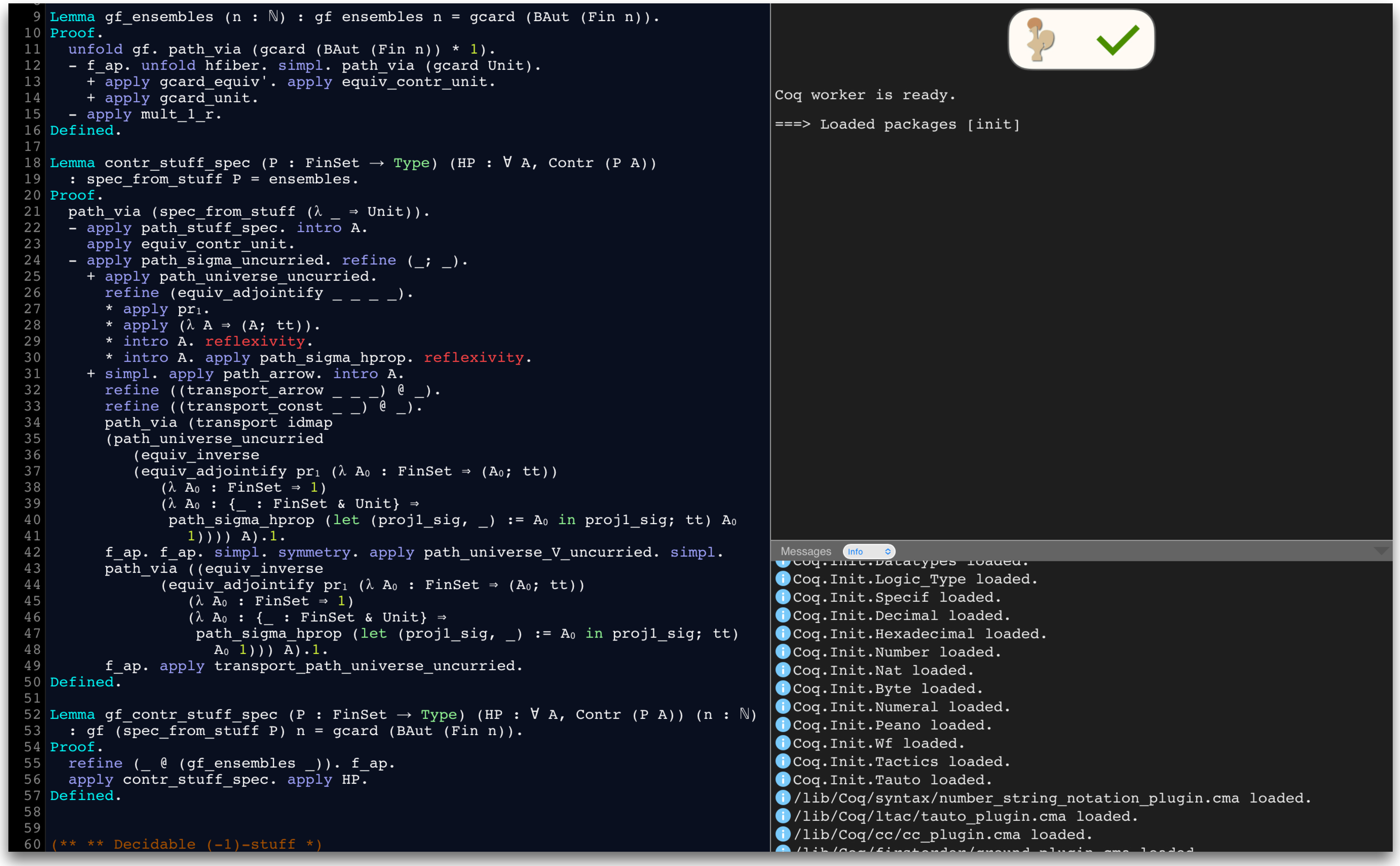
- Difficult and work-intensive to **install/compile from source** on some systems
- As both a **proof assistant** *and* a **programming language**, understanding the theory behind Coq and acquiring a working knowledge of the semantics/technical peculiarities of the Coq system is quite work-intensive
- Finding and analyzing proofs is mostly a **manual** (if assisted) **process** — **standardization** and **searchability**?
- **Curation**, **quality control** and medium- to long-term **maintenance** of collections of proofs is challenging
- **"Burden of interdisciplinarity"** — documenting a given piece of mathematical knowledge in a wiki system requires substantial amounts of **human-readable** and potentially highly technical text, potentially with very involved **mathematical examples** for illustration, while designing corresponding proofs in Coq requires a form of programming which has to be aided by some form of **Coq API** documentation and ideally a library of **code examples**
- …


https://youtu.be/4084o1hk1Qs

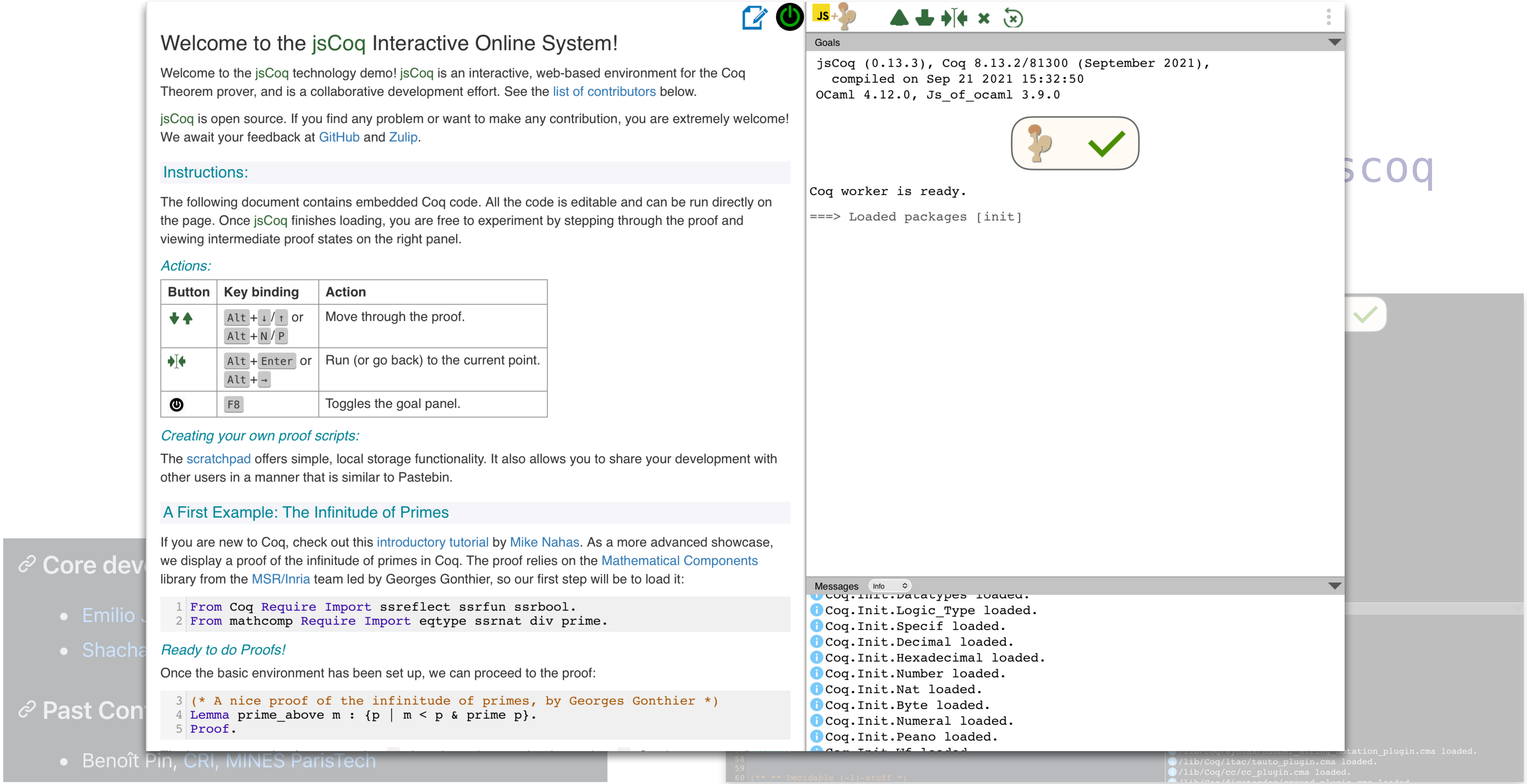
# jsCoq




https://github.com/jscoq

# jsCoq



Welcome to the jsCoq Interactive Online System!
Welcome to the jsCoq technology demo! jsCoq is an interactive, web-based environment for the Coq Theorem prover, and is a collaborative development effort. See the list of contributors below.
jsCoq is open source. If you find any problem or want to make any contribution, you are extremely welcome! We await your feedback at GitHub and Zulip.
Instructions:
The following document contains embedded Coq code. All the code is editable and can be run directly on the page. Once jsCoq finishes loading, you are free to experiment by stepping through the proof and viewing intermediate proof states on the right panel.
Actions:
Button | Key binding | Action
Alt + ↓ / ↑ or Alt + N / P | Move through the proof.
Alt + Enter or Alt + → | Run (or go back) to the current point.
F8 | Toggles the goal panel.
Creating your own proof scripts:
The scratchpad offers simple, local storage functionality. It also allows you to share your development with other users in a manner that is similar to Pastebin.
A First Example: The Infinitude of Primes
If you are new to Coq, check out this introductory tutorial by Mike Nahas. As a more advanced showcase, we display a proof of the infinitude of primes in Coq. The proof relies on the Mathematical Components library from the MSR/Inria team led by Georges Gonthier, so our first step will be to load it:
From Coq Require Import ssreflect ssrfun ssrbool.
From mathcomp Require Import eqtype ssrnat div prime.
Ready to do Proofs!
Once the basic environment has been set up, we can proceed to the proof:
(* A nice proof of the infinitude of primes, by Georges Gonthier *)
Lemma prime_above m : {p | m < p & prime p}.
Proof.
Goals
jsCoq (0.13.3), Coq 8.13.2/81300 (September 2021),
compiled on Sep 21 2021 15:32:50
OCaml 4.12.0, Js_of_ocaml 3.9.0
Coq worker is ready.
===> Loaded packages [init]
Messages
Info
Coq.Init.Logic_Type loaded.
Coq.Init.Specif loaded.
Coq.Init.Decimal loaded.
Coq.Init.Hexadecimal loaded.
Coq.Init.Number loaded.
Coq.Init.Nat loaded.
Coq.Init.Byte loaded.
Coq.Init.Numeral loaded.
Coq.Init.Peano loaded.
Core dev
Emilio
Shacha
Past Con
Benoît Pin, CRI, MINES ParisTech

# jsCoq

```
Lemma gf_ensembles (n : ℕ) : gf ensembles n = gcard (BAut (Fin n)).
Proof.
  unfold gf. path_via (gcard (BAut (Fin n)) * 1).
  - f_ap. unfold hfiber. simpl. path_via (gcard Unit).
    + apply gcard_equiv'. apply equiv_contr_unit.
    + apply gcard_unit.
  - apply mult_1_r.
Defined.

Lemma contr_stuff_spec (P : FinSet → Type) (HP : ∀ A, Contr (P A))
  : spec_from_stuff P = ensembles.
Proof.
  path_via (spec_from_stuff (λ _ ⇒ Unit)).
  - apply path_stuff_spec. intro A.
    apply equiv_contr_unit.
  - apply path_sigma_uncurried. refine (_; _).
    + apply path_universe_uncurried.
      refine (equiv_adjointify _ _ _ _).
      * apply pr1.
      * apply (λ A ⇒ (A; tt)).
      * intro A. reflexivity.
      * intro A. apply path_sigma_hprop. reflexivity.
    + simpl. apply path_arrow. intro A.
      refine ((transport_arrow _ _ _) @ _).
      refine ((transport_const _ _) @ _).
      path_via (transport idmap
      (path_universe_uncurried
         (equiv_inverse
         (equiv_adjointify pr1 (λ A0 : FinSet ⇒ (A0; tt))
            (λ A0 : FinSet ⇒ 1)
            (λ A0 : {_ : FinSet & Unit} ⇒
             path_sigma_hprop (let (proj1_sig, _) := A0 in proj1_sig; tt) A0
              1)))) A).1.
      f_ap. f_ap. simpl. symmetry. apply path_universe_V_uncurried. simpl.
      path_via ((equiv_inverse
            (equiv_adjointify pr1 (λ A0 : FinSet ⇒ (A0; tt))
               (λ A0 : FinSet ⇒ 1)
               (λ A0 : {_ : FinSet & Unit} ⇒
                path_sigma_hprop (let (proj1_sig, _) := A0 in proj1_sig; tt)
                  A0 1))) A).1.
      f_ap. apply transport_path_universe_uncurried.
Defined.

Lemma gf_contr_stuff_spec (P : FinSet → Type) (HP : ∀ A, Contr (P A)) (n : ℕ)
  : gf (spec_from_stuff P) n = gcard (BAut (Fin n)).
Proof.
  refine (_ @ (gf_ensembles _)). f_ap.
  apply contr_stuff_spec. apply HP.
Defined.


(** ** Decidable (-1)-stuff *)
```

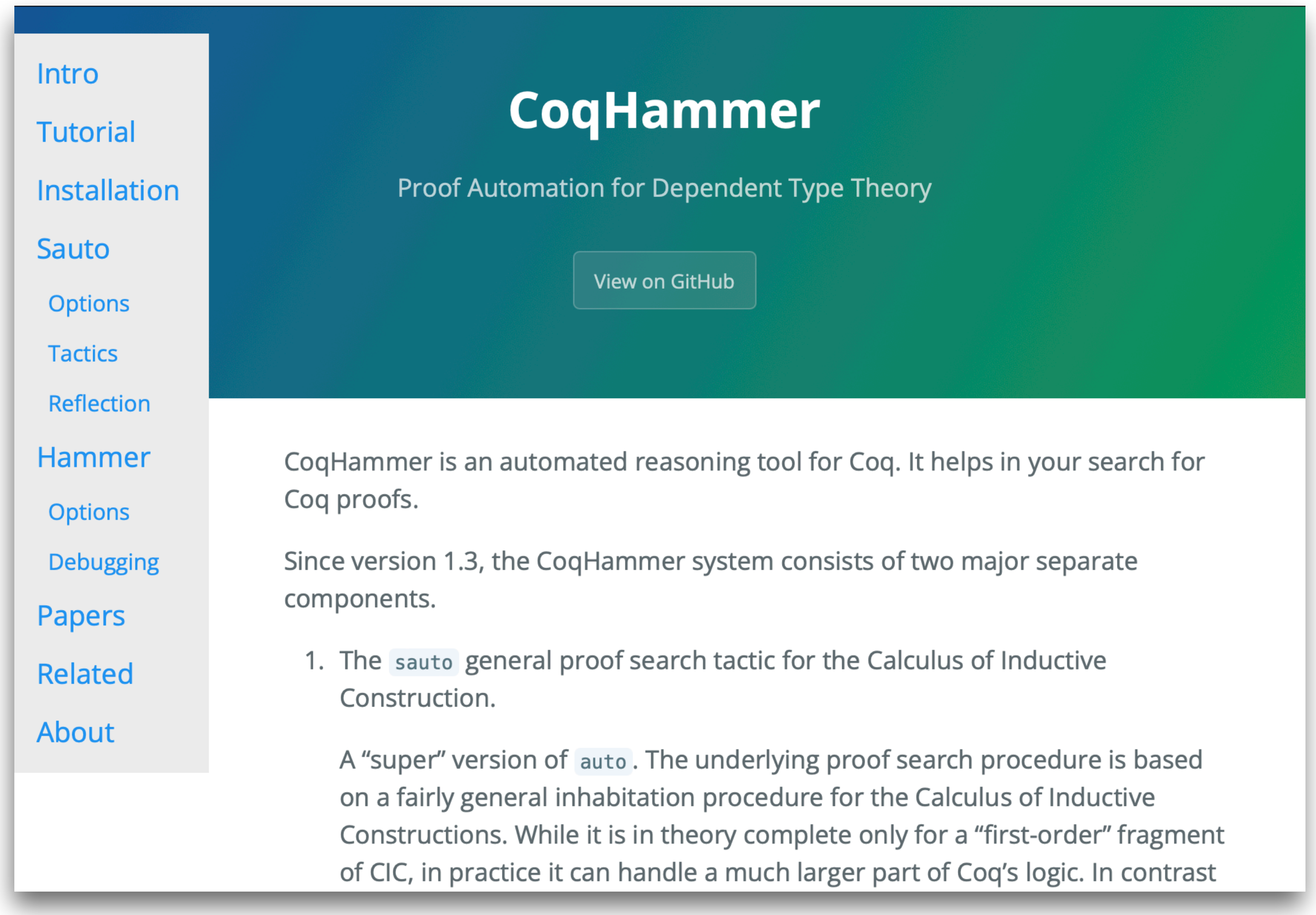
Coq worker is ready.

===> Loaded packages [init]

Messages Info

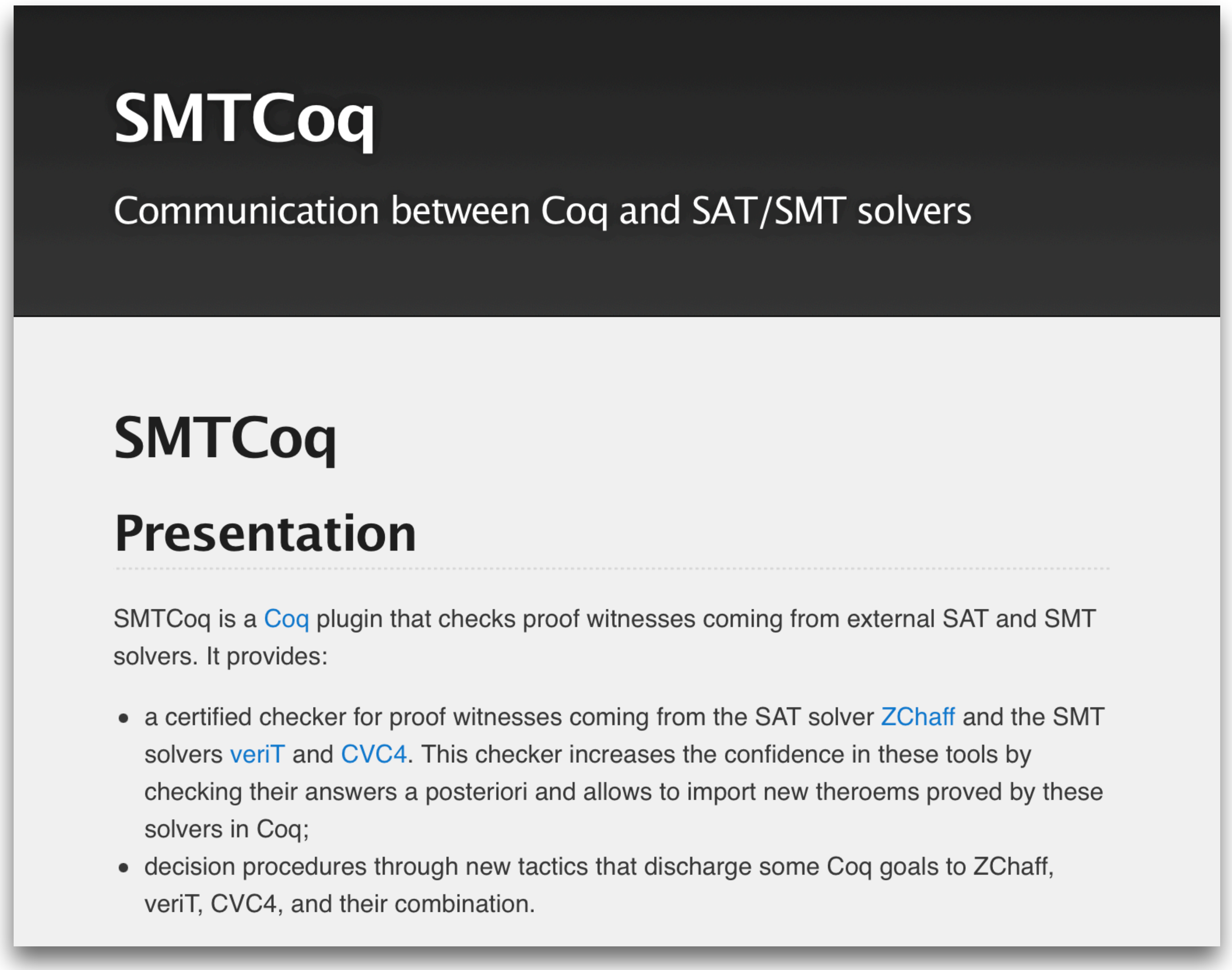
```
Coq.Init.Datatypes loaded.
Coq.Init.Logic_Type loaded.
Coq.Init.Specif loaded.
Coq.Init.Decimal loaded.
Coq.Init.Hexadecimal loaded.
Coq.Init.Number loaded.
Coq.Init.Nat loaded.
Coq.Init.Byte loaded.
Coq.Init.Numeral loaded.
Coq.Init.Peano loaded.
Coq.Init.Wf loaded.
Coq.Init.Tactics loaded.
Coq.Init.Tauto loaded.
/lib/Coq/syntax/number_string_notation_plugin.cma loaded.
/lib/Coq/ltac/tauto_plugin.cma loaded.
/lib/Coq/cc/cc_plugin.cma loaded.
/lib/Coq/firstorder/ground_plugin.cma loaded.
```

# Towards **automated** theorem-proving and **tactics-learning**

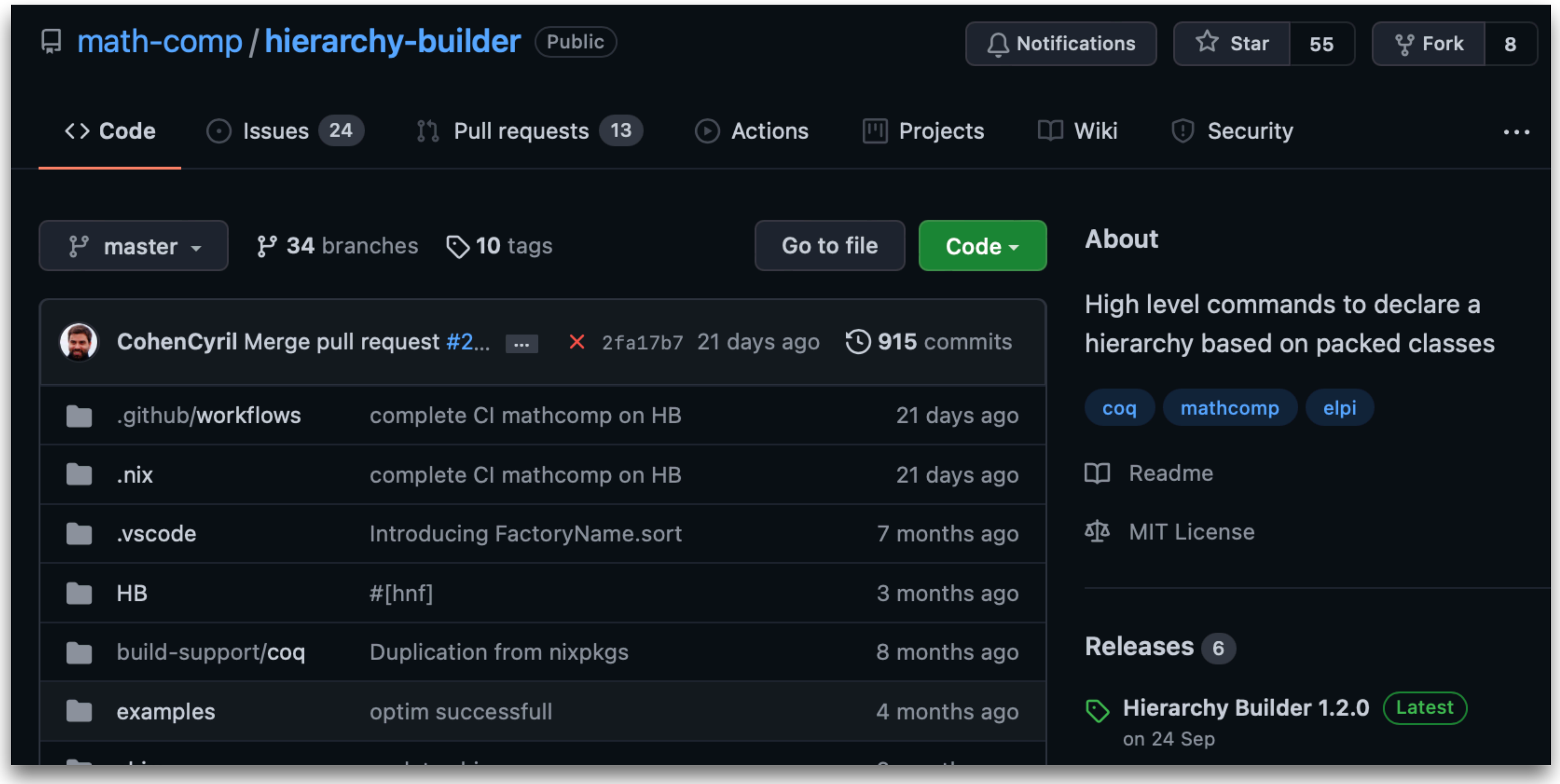



https://coqhammer.github.io


https://coq-tactician.github.io

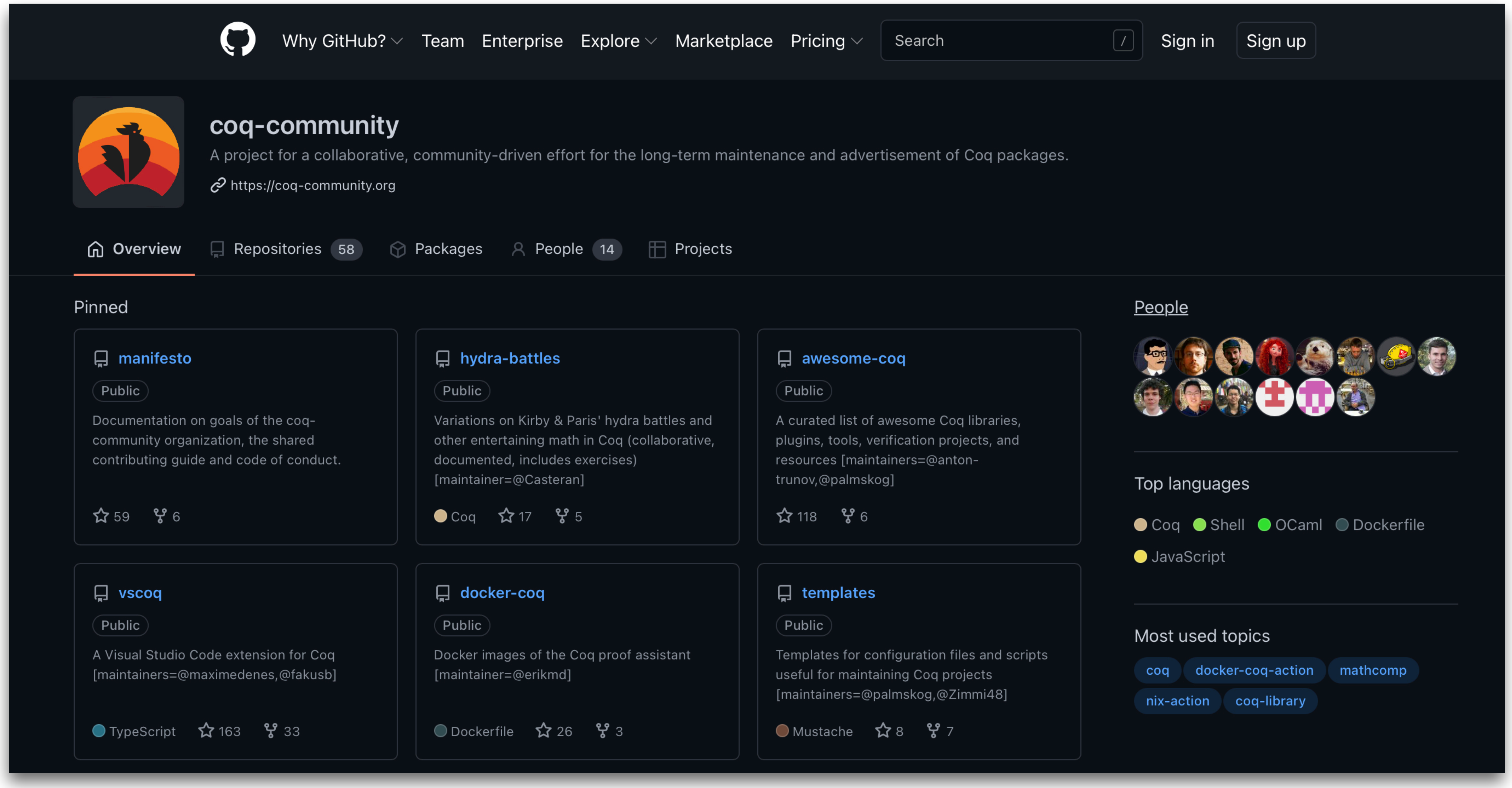



https://smtcoq.github.io




https://github.com/math-comp/hierarchy-builder

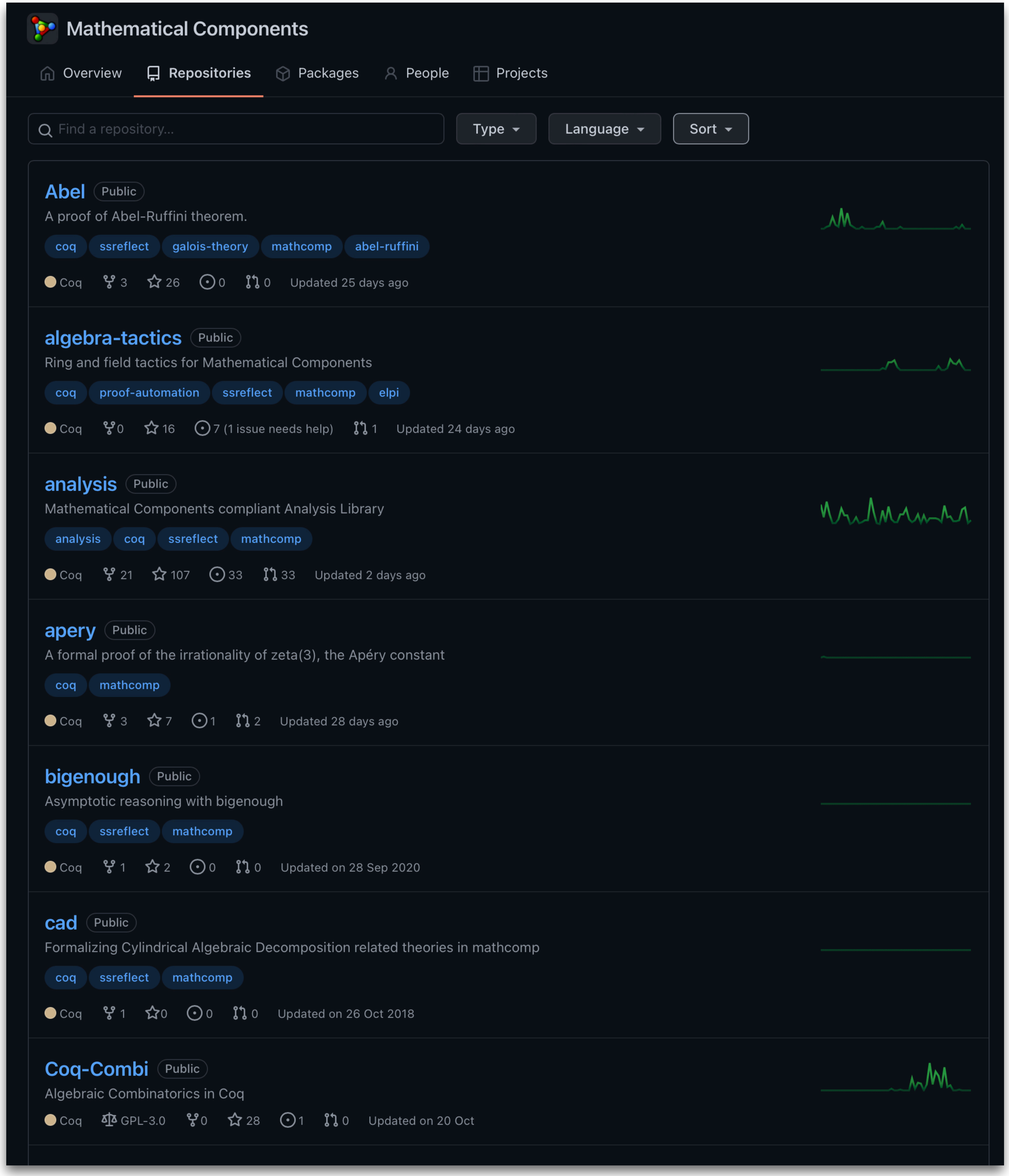
# Coq-community



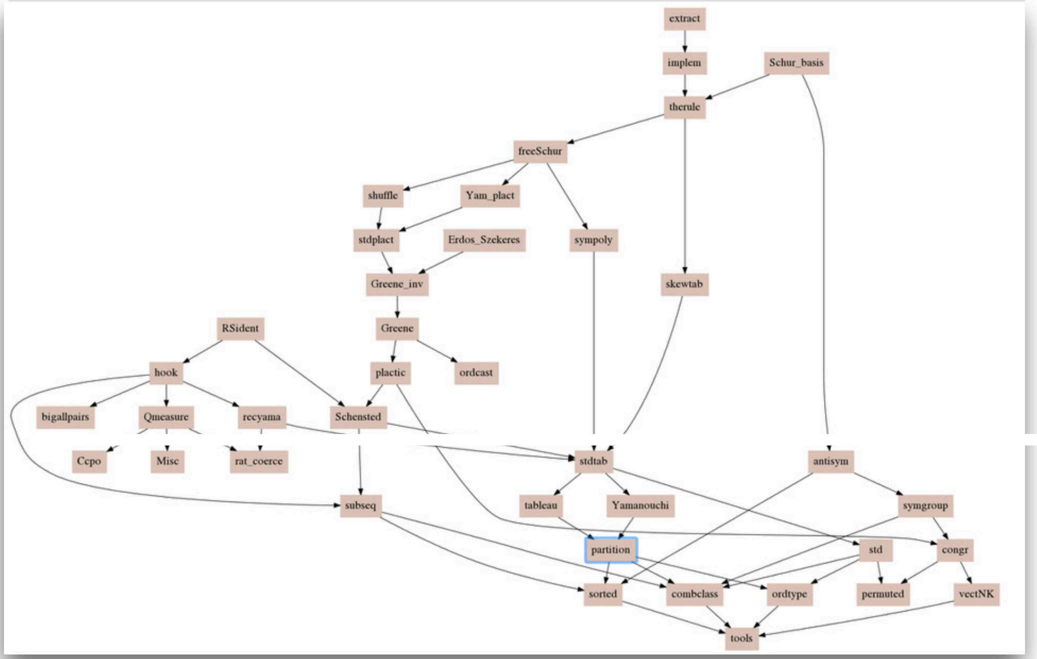

https://github.com/coq-community

*"A project for a **collaborative**, **community-driven** effort for the **long-term maintenance** and **advertisement** of Coq packages."*

- **58 repositories**



# The **coreact.wiki** proposal



**knowledge graphs**

(graph-) **database**

**Maintenance:** collaborate with/ adopt standards of **Coq-community**

**PDF & HTML textbook**

**wiki entry** (e.g., a Lemma)

**#hash** (auto-generated)
**tags**
**list of cross-references**
**bibliographic references**
**code origin references**

**Human-readable text**

`LaTeX`-based, with annotations permitting generation of cross-references via `NNexus`

**Machine-readable Coq-formalization**

Including compatible Coq version and possibly different variants for (1) different Coq versions and/or (2) different implementation strategies/frameworks/theories.

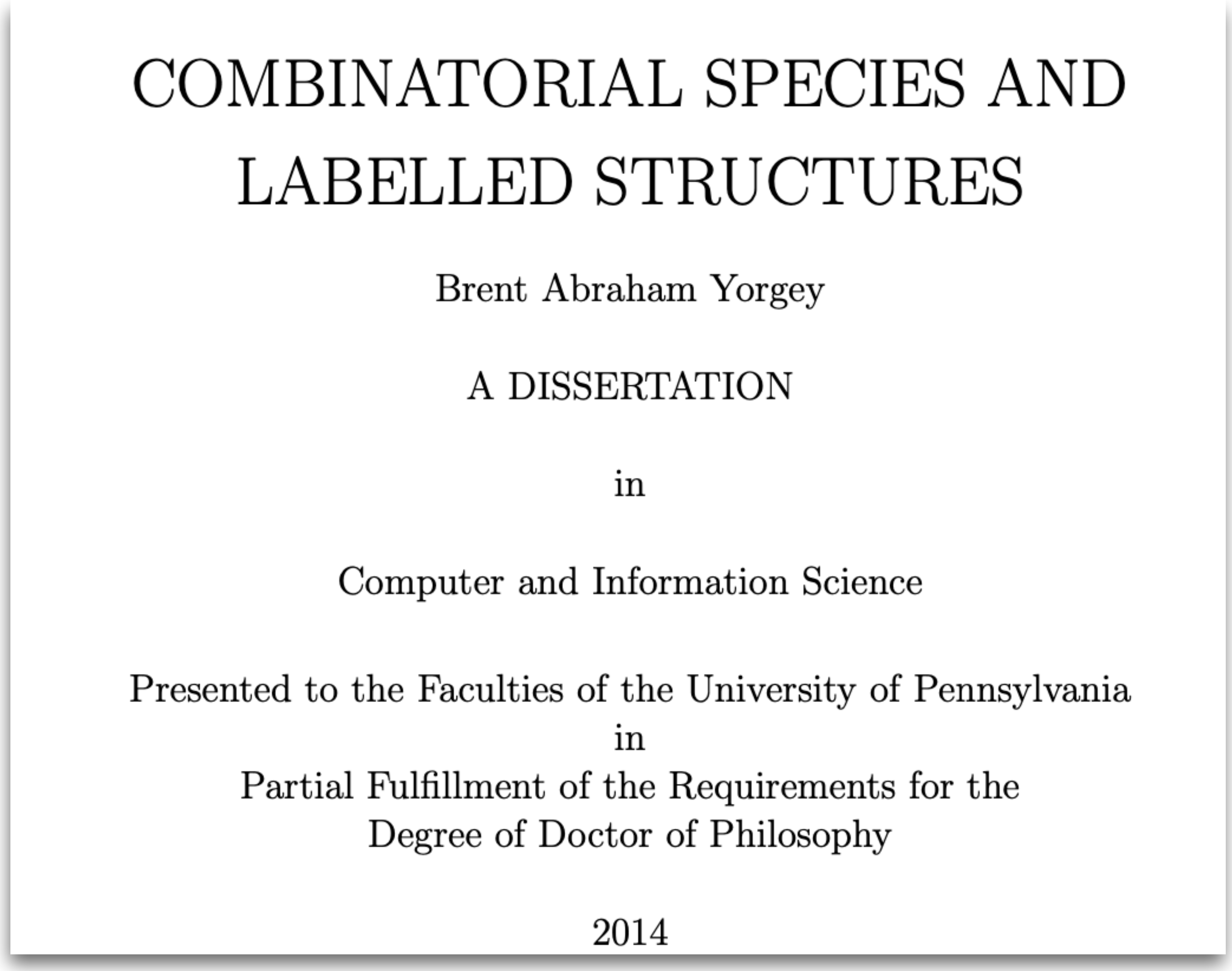
**Examples** (**both** maths & Coq)

Curated in `jsCoq`, directly executable from within the wiki entry in the form of a literate web document and/or as a bundle of a Coq file with instructions for a particular Docker image for Coq.

**Proof tactics and performance data**

Machine-learned tactics data, cross-evaluation of performance of different variants of implementations, user annotations on different Coq versions/libraries used

# A proposal for a **wiki-topic case study: HoTT Species**



COMBINATORIAL SPECIES AND LABELLED STRUCTURES

Brent Abraham Yorgey

A DISSERTATION

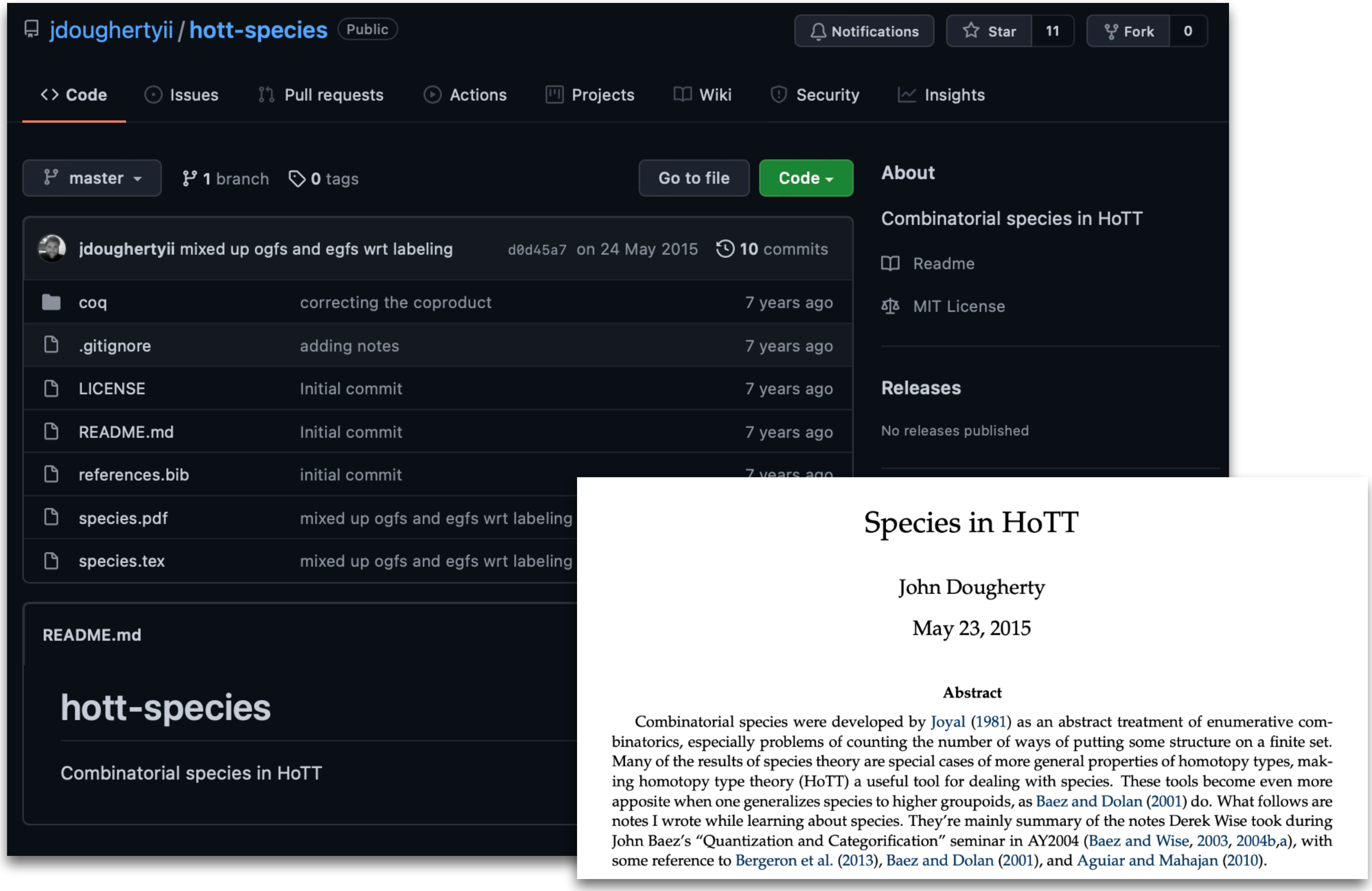
in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania
in
Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy

2014

https://repository.upenn.edu/edissertations/1512/




https://github.com/jdoughertyii/hott-species

# A proposal for a **wiki-topic case study: HoTT Species**



| If F is a... | then $\lvert F\rvert(z)=\sum \frac{a_n}{n!} z^n$ where: | since these numbers are cardinalities of: |
|---|---|---|
| stuff type | $a_n \in \mathbb{R}^+ = [0,\infty)$ | (tame) groupoids = 1-groupoids |
| structure type | $a_n \in \mathbb{N}$ | (finite) sets = 0-groupoids |
| property type | $a_n \in \{0,1\} \cong \{F,T\}$ | truth values = -1-groupoids |
| vacuous property type | $a_n \in \{1\} \cong \{T\}$ | true = the only -2-groupoid |

**John Baez**, Quantum Gravity Seminar — Spring 2004: Quantization and Categorification, *Week 3* - Evaluating and composing stuff types (notes by *D. Wise*)

https://math.ucr.edu/home/baez/qg-spring2004/s04week03.pdf

# A proposal for a **wiki-topic case study: HoTT Species**



| If F is a... | then $\lvert F\rvert(z)=\sum \frac{a_n}{n!} z^n$ where: | Since these numbers are cardinalities of: |
|---|---|---|
| stuff type | $a_n \in \mathbb{R}^+ = [0,\infty)$ | (tame) groupoids = 1-groupoids |
| structure type | $a_n \in \mathbb{N}$ | (finite) sets = 0-groupoids |
| property type | $a_n \in \{0,1\} \cong \{F,T\}$ | truth values = -1-groupoids |
| vacuous property type | $a_n \in \{1\} \cong \{T\}$ | true = the only -2-groupoid |

**John Baez**, Quantum Gravity Seminar — Spring 2004: Quantization and Categorification, *Week 3* - Evaluating and composing stuff types (notes by *D. Wise*)

https://math.ucr.edu/home/baez/qg-spring2004/s04week03.pdf

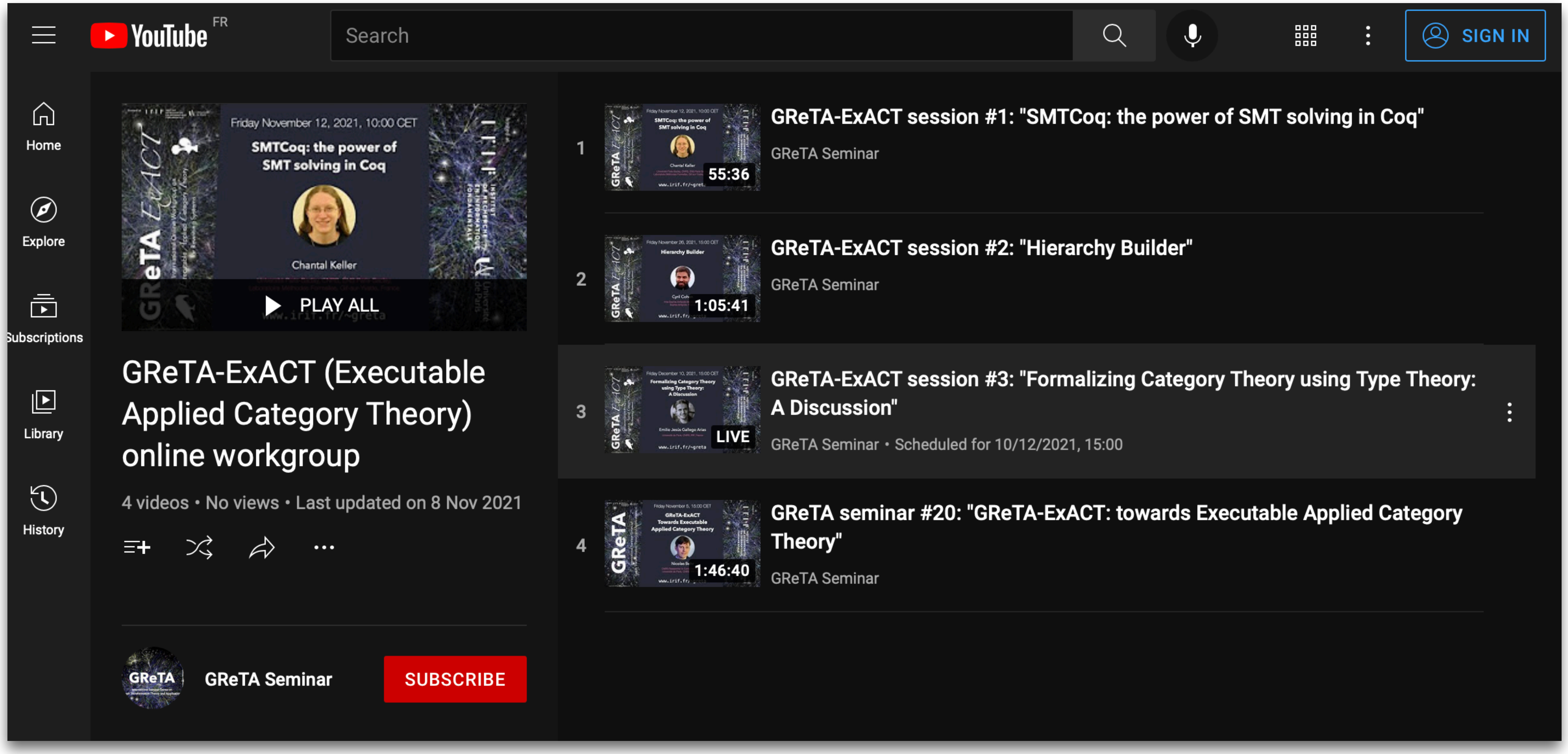
# A proposal for a **wiki-topic case study: HoTT Species**



| If F is a... | then $\lvert F\rvert(z)=\sum \frac{a_n}{n!} z^n$ where: | since these numbers are cardinalities of: |
|---|---|---|
| stuff type | $a_n \in \mathbb{R}^+ = [0,\infty)$ | (tame) groupoids = 1-groupoids |
| structure type | $a_n \in \mathbb{N}$ | (finite) sets = 0-groupoids |
| property type | $a_n \in \{0,1\} \cong \{F,T\}$ | truth values = -1-groupoids |
| vacuous property type | $a_n \in \{1\} \cong \{T\}$ | true = the only -2-groupoid |

**John Baez**, Quantum Gravity Seminar — Spring 2004: Quantization and Categorification, *Week 3* - Evaluating and composing stuff types (notes by *D. Wise*)

https://math.ucr.edu/home/baez/qg-spring2004/s04week03.pdf


coreact.wiki


# Merci beaucoup !