Combinatorics via Rule-Algebraic Methods

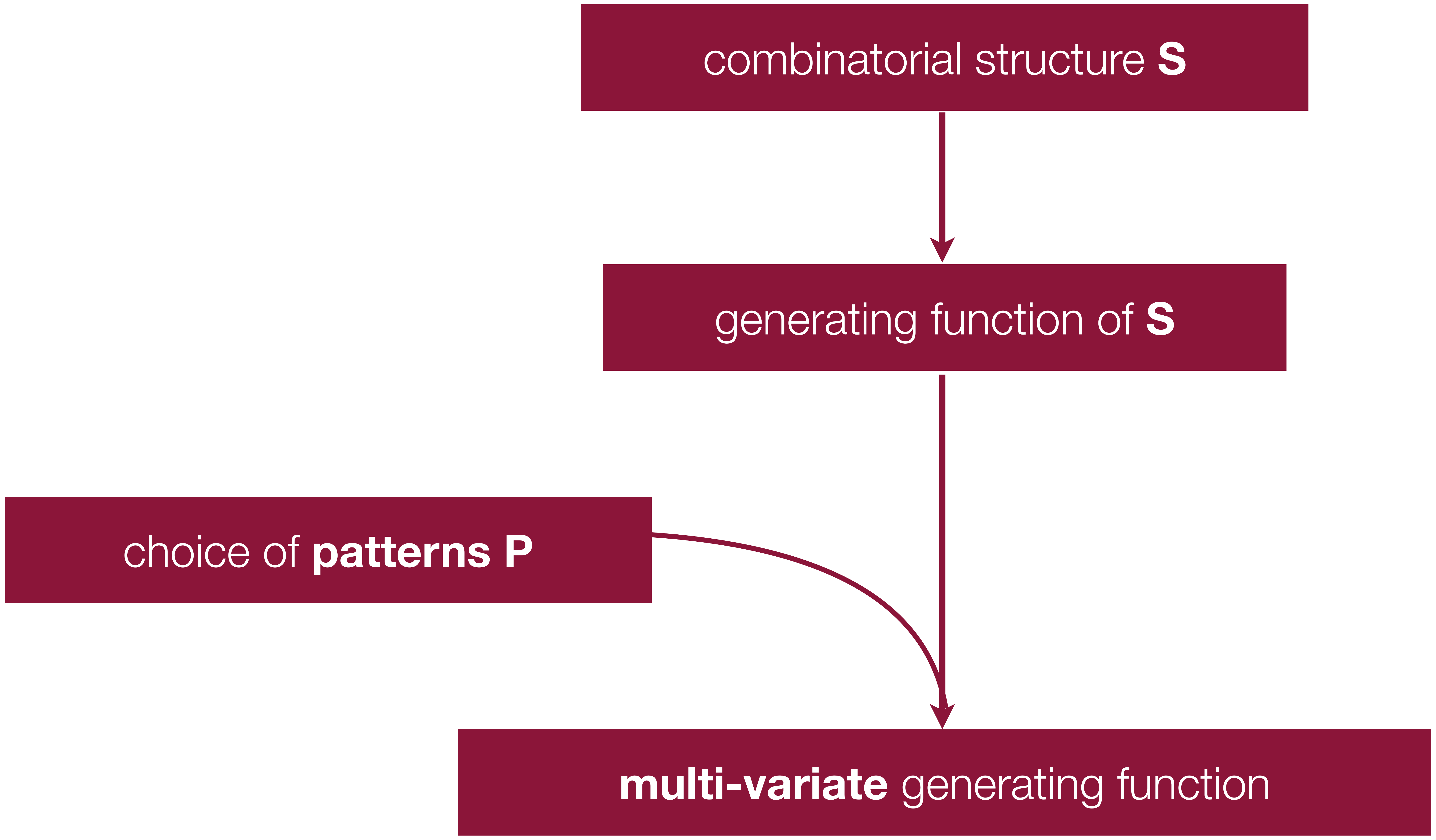
Nicolas Behr (Université de Paris, CNRS, IRIF)

Species and operads in combinatorics and semantics (SOCS 2020)

December 11, 2020

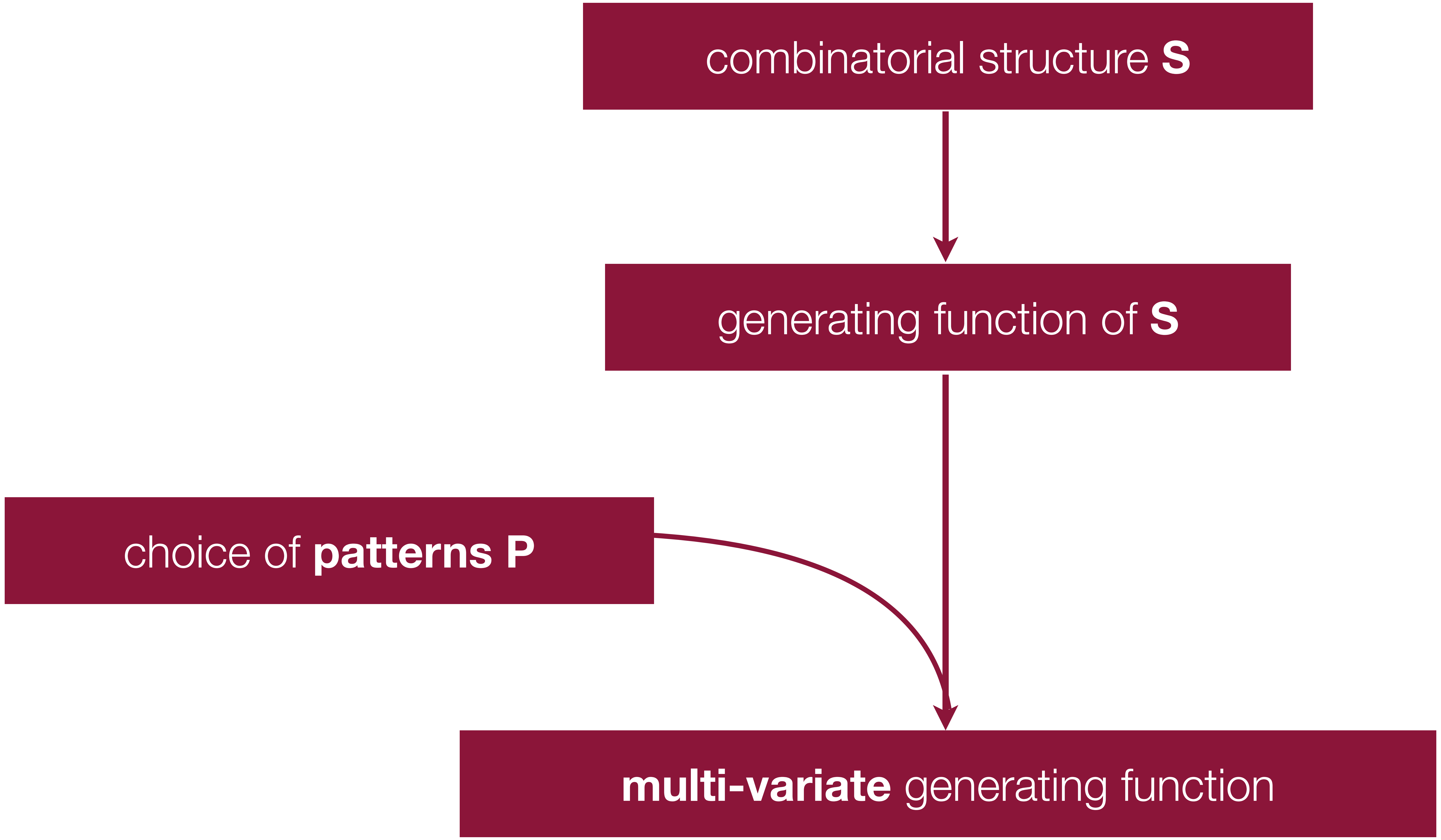
Motivation

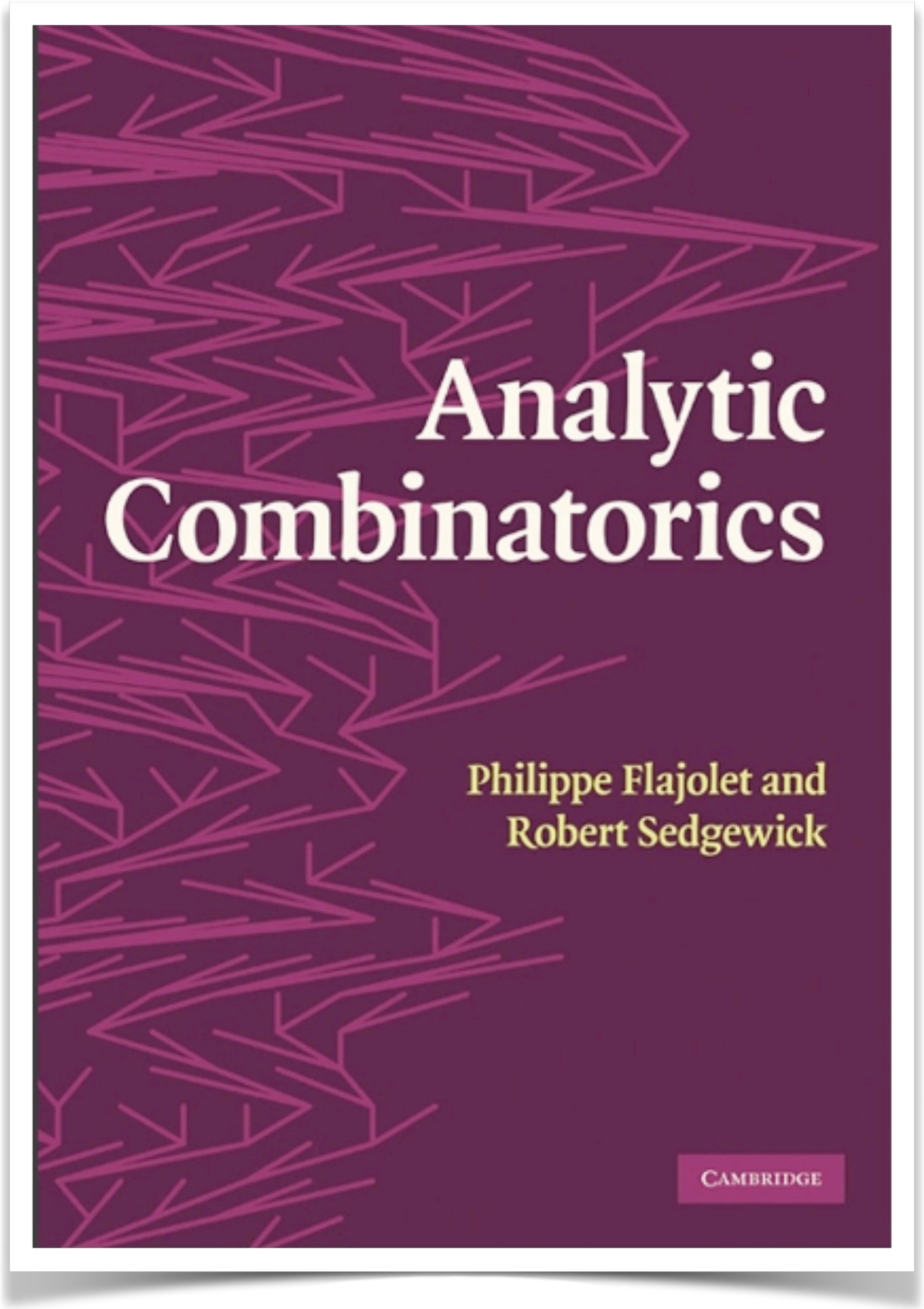
The enumerative combinatorics "workflow" (à la Flajolet):



Motivation

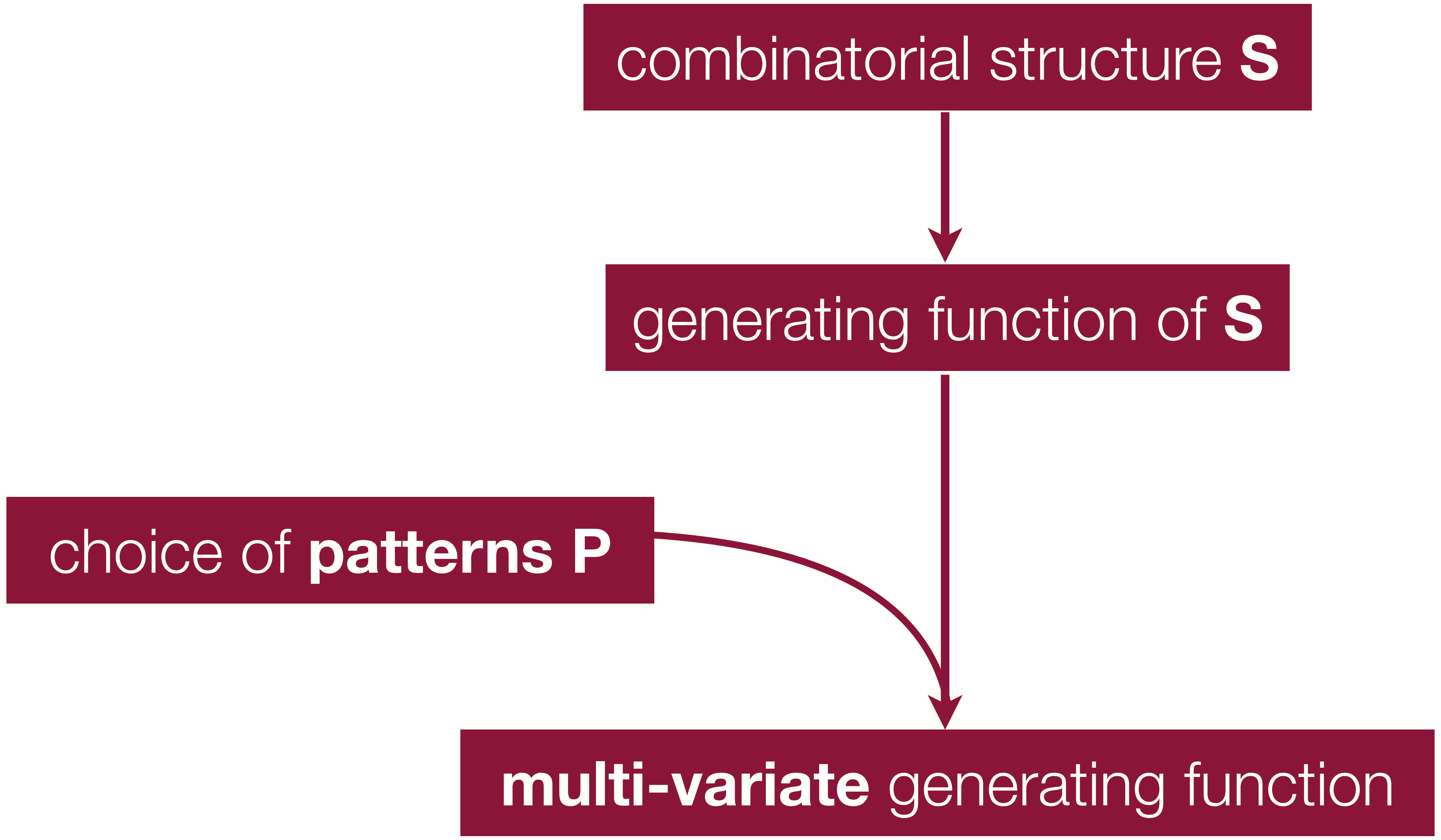
The enumerative combinatorics "workflow" (à la Flajolet):





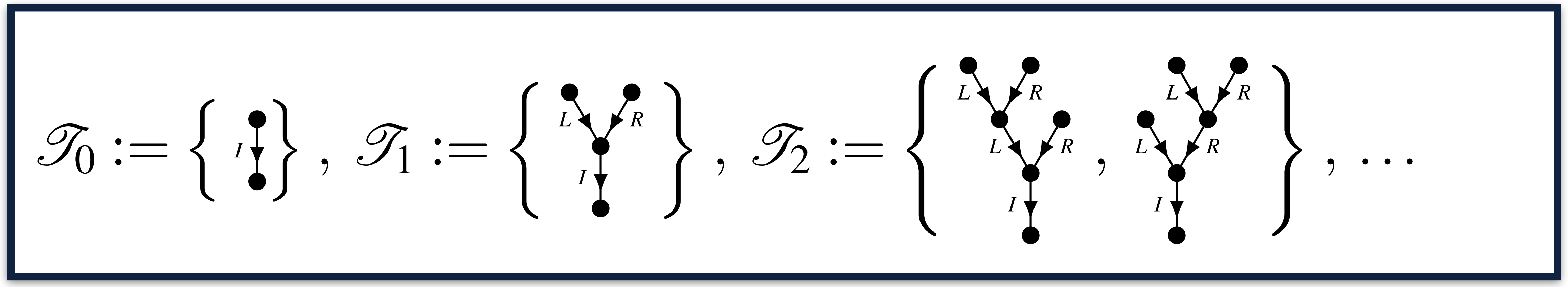
Motivation

Example: planar rooted binary trees (PRBTs)



Motivation

Example: planar rooted binary trees (PRBTs)



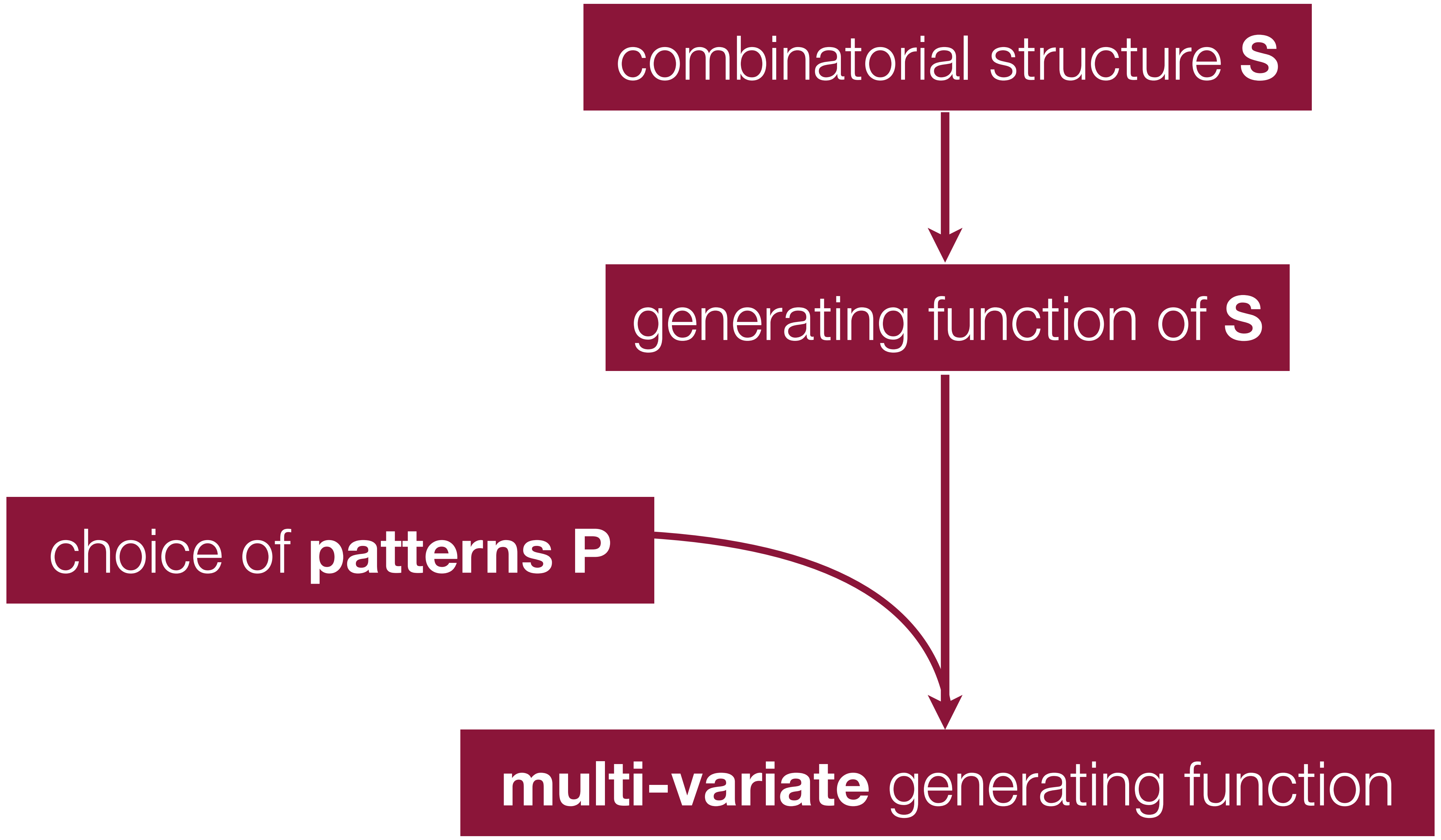
combinatorial structure **S**

generating function of **S**

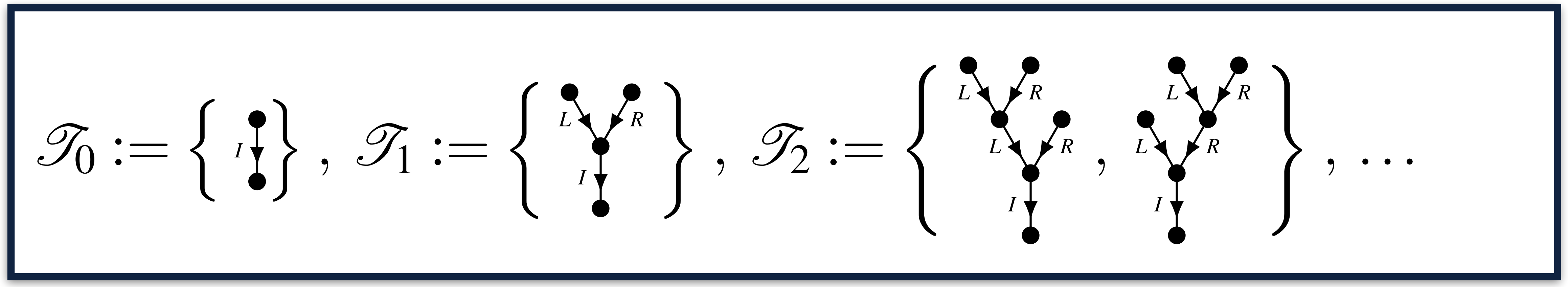
choice of **patterns P**

multi-variate generating function

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

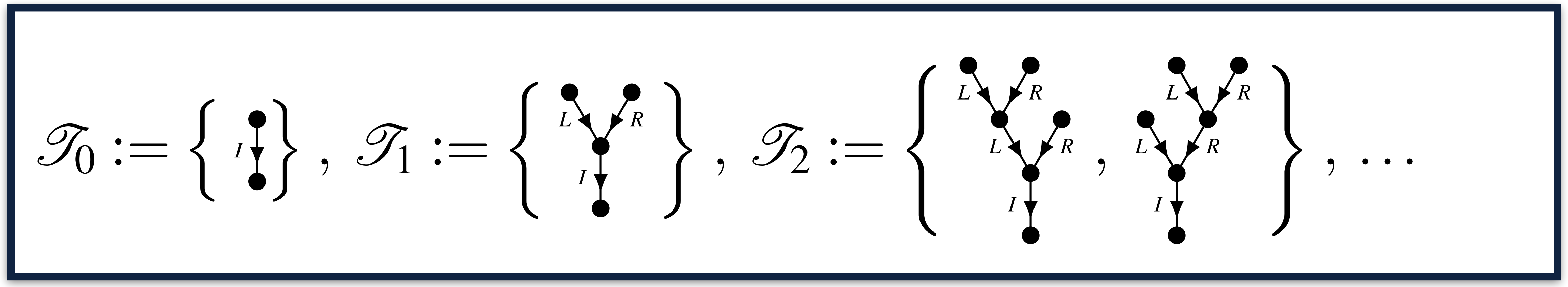
combinatorial structure **S**

generating function of **S**

choice of **patterns P**

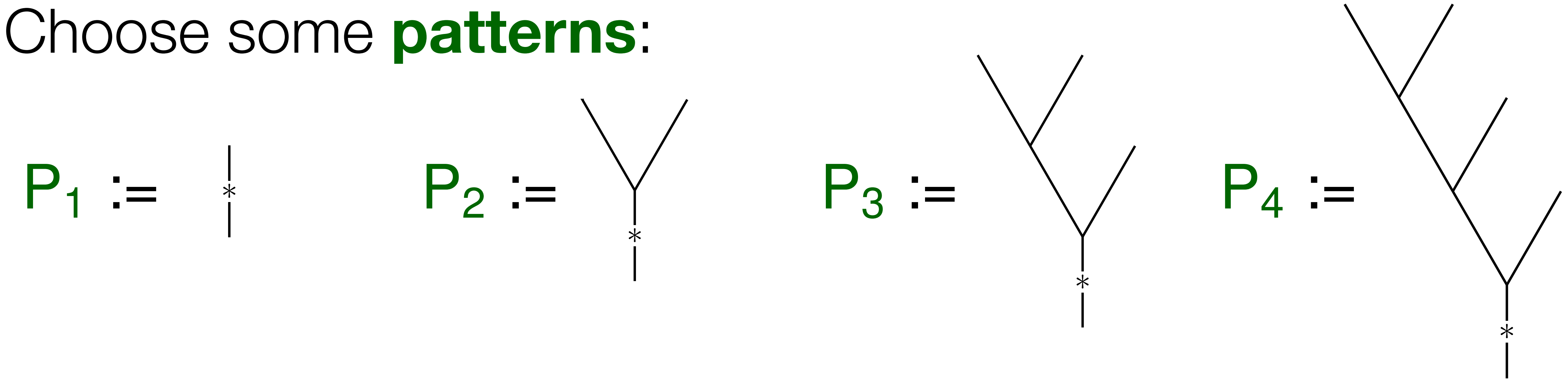
multi-variate generating function

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**:



Motivation

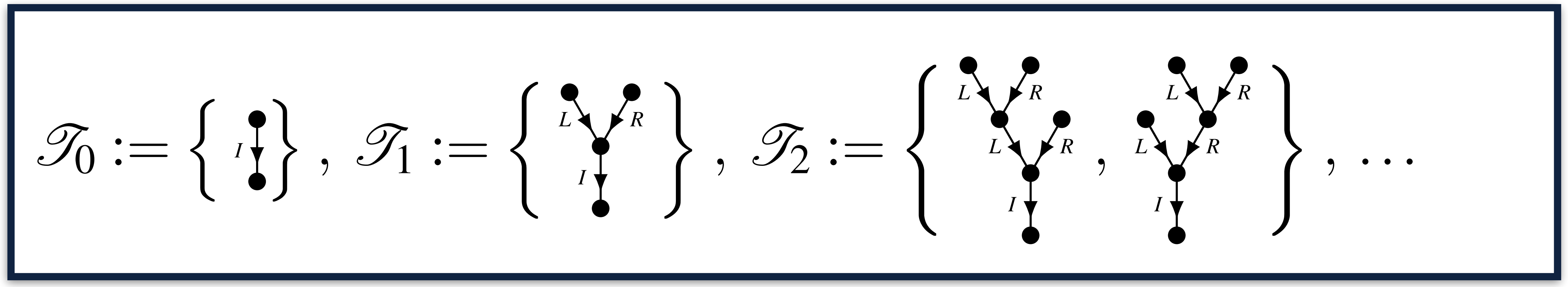
combinatorial structure **S**

generating function of **S**

choice of **patterns P**

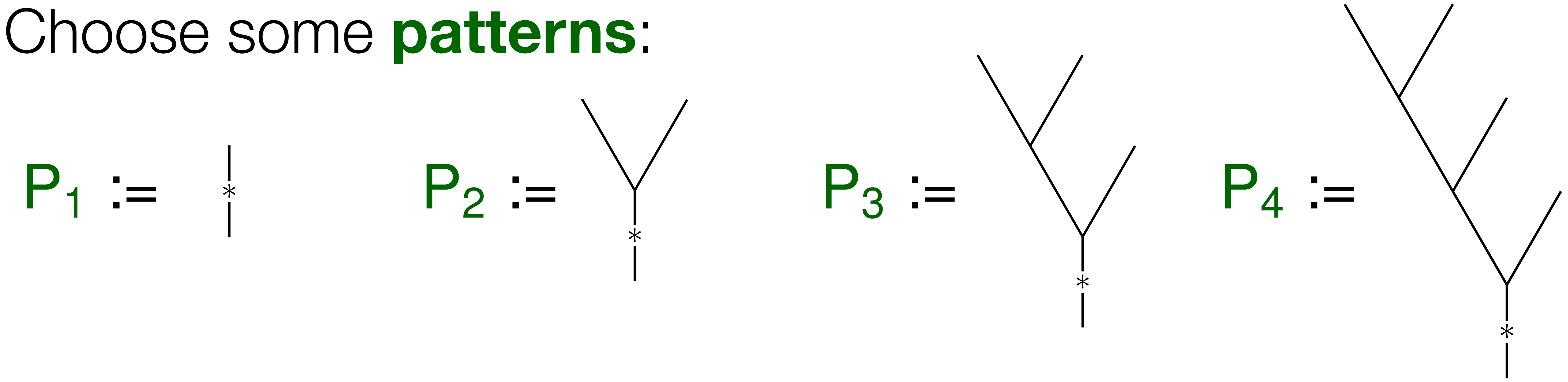
multi-variate generating function

Example: planar rooted binary trees (PRBTs)



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

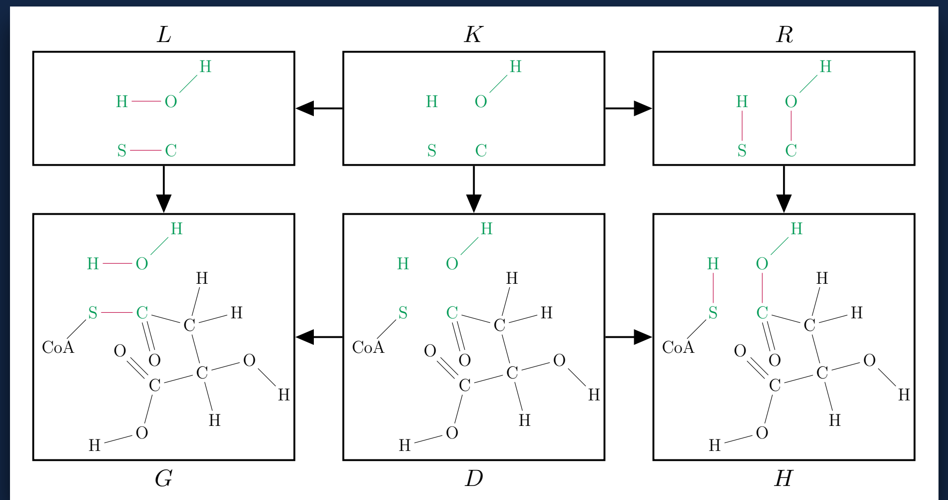
Choose some **patterns**:



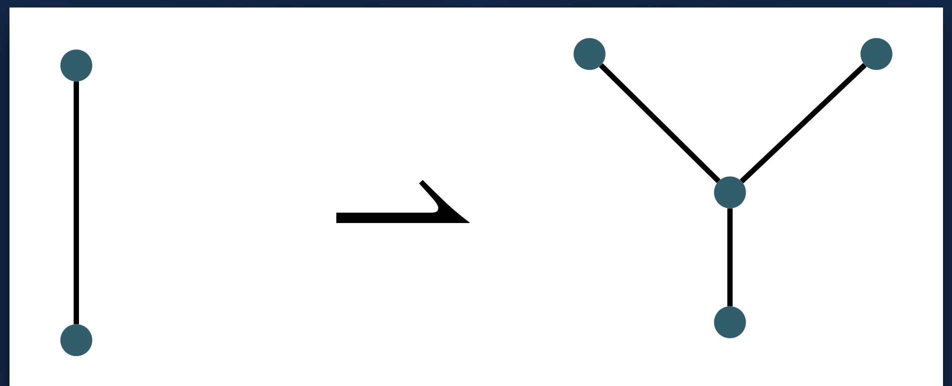
$$\mathcal{G}(\lambda; \omega_1, \ldots, \omega_k) := \sum_{n\geq 0} \frac{\lambda^n}{n!} \sum_{p_1,\ldots,p_k \geq 0} \frac{\omega_1^{p_1} \cdots \omega_k^{p_k}}{p_1! \cdots p_k!} \left(\begin{array}{l} \text{\# of structures of size } n \\ \quad \text{and with } p_i \text{ occurrences of pattern }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

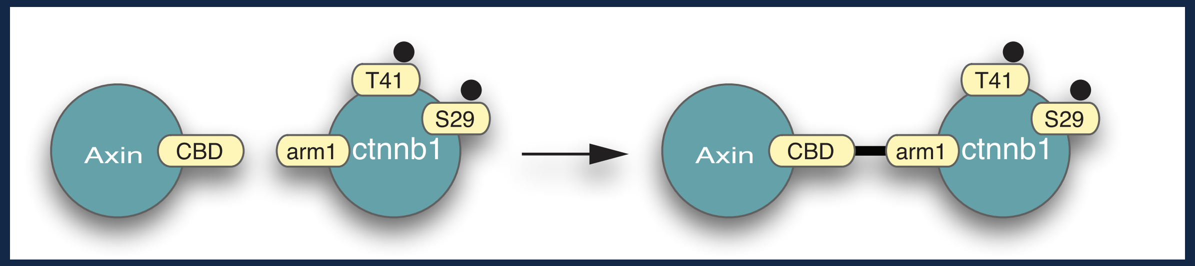
COMPUTER SCIENCE

- **semantics** and **stochastic rewriting theory**
- **concurrency theory**
- **algorithms** for **bio-** and **organo-chemistry**

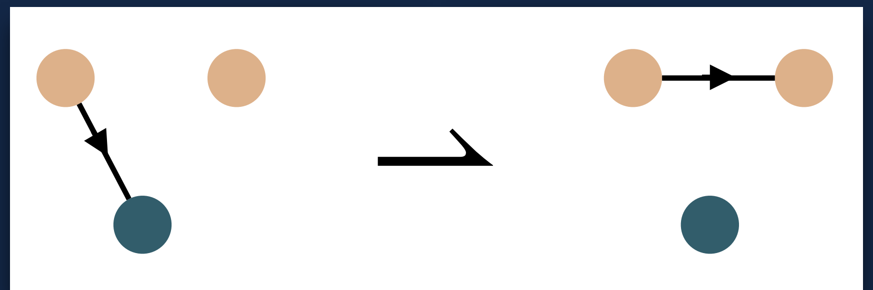


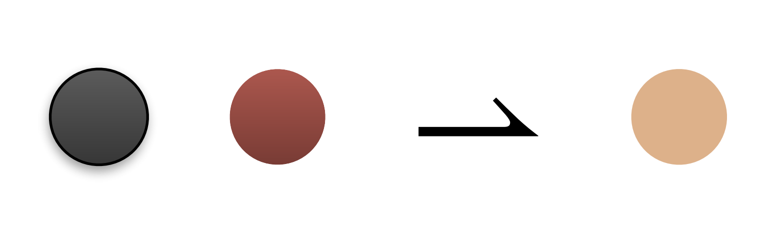
organic chemistry





biochemistry





MATHEMATICAL PHYSICS

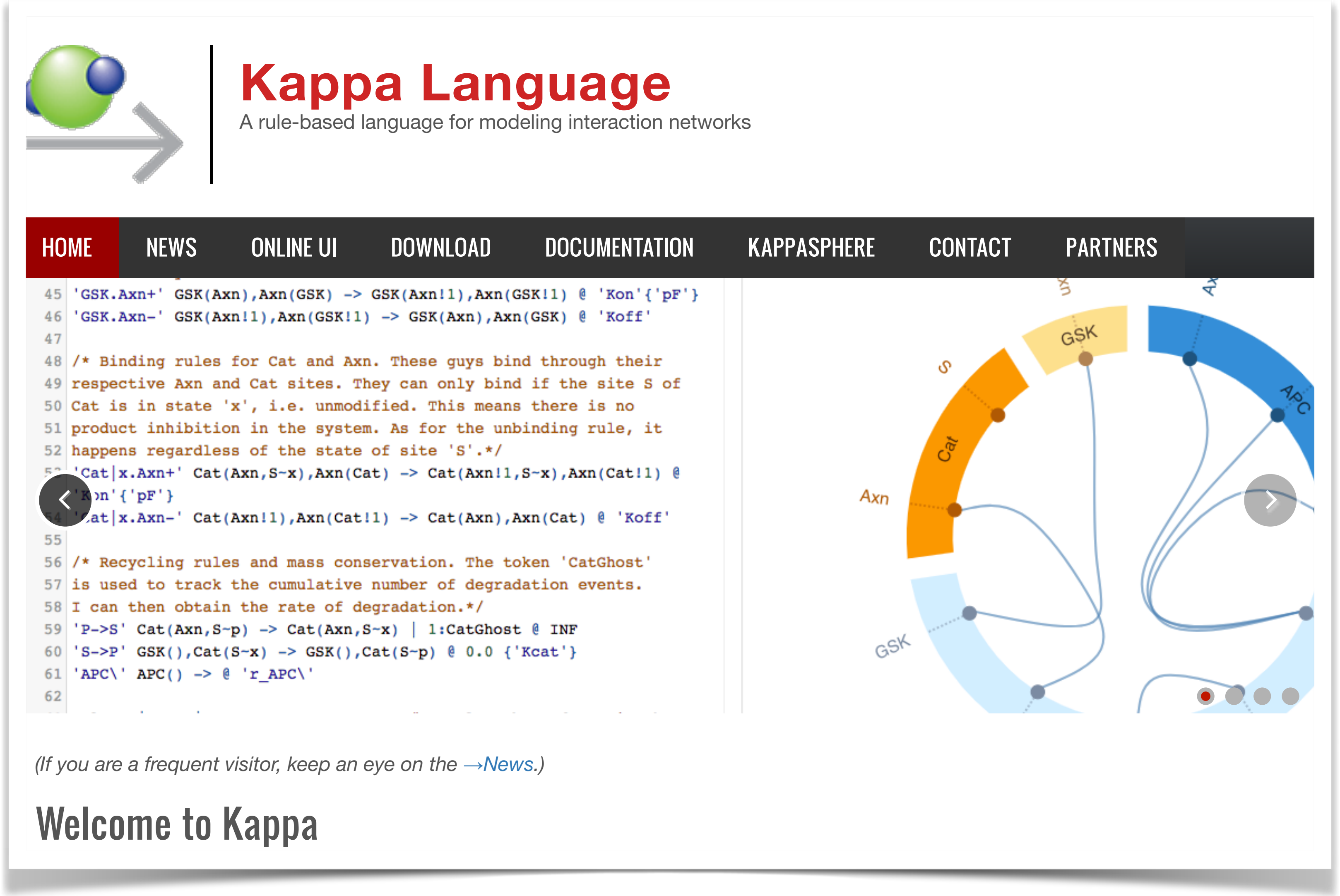
- **continuous-time Markov chains (CTMCs)**
- **statistical mechanics**

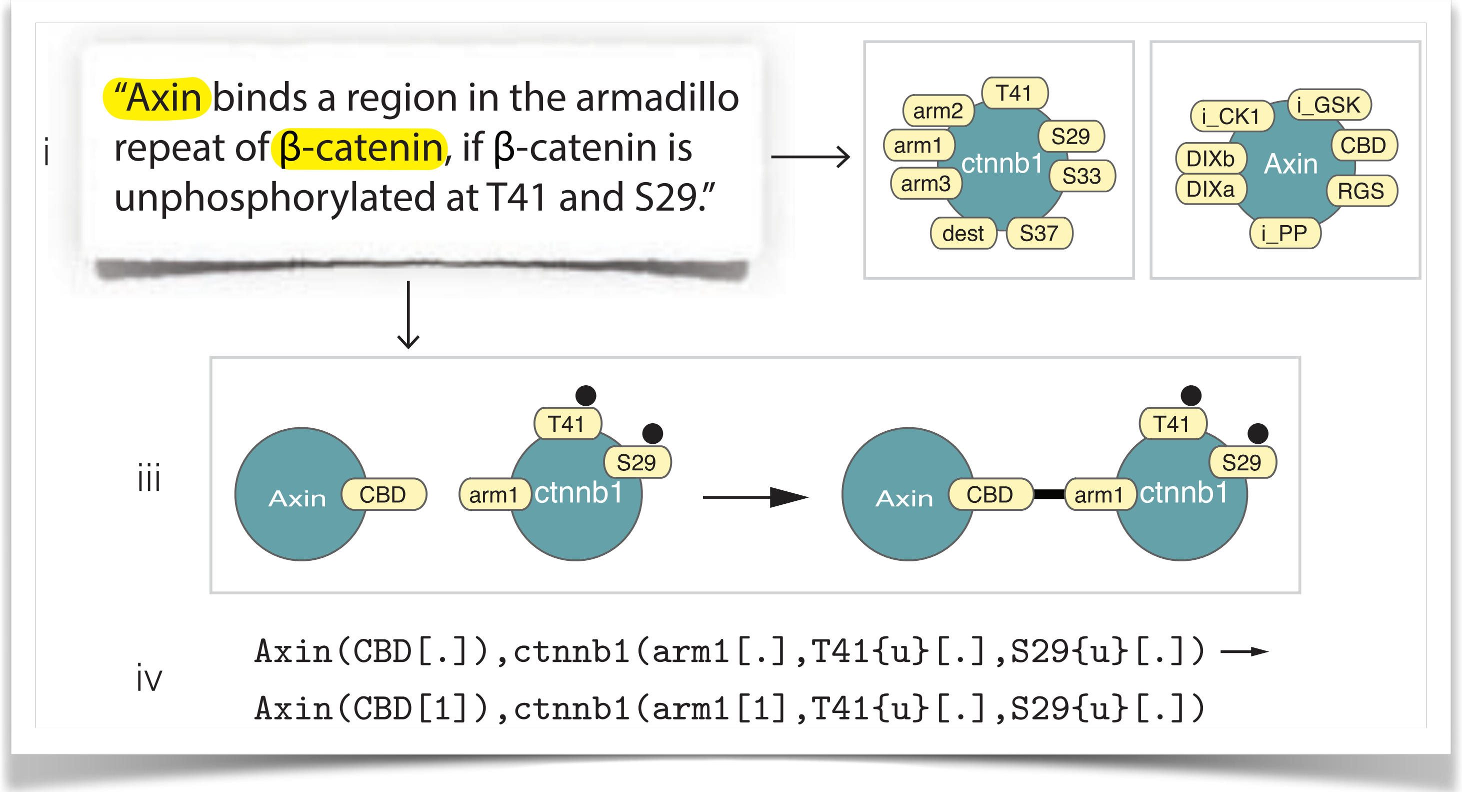
MATHEMATICS

- enumerative/algebraic **combinatorics**
- theory of **M-adhesive categories**

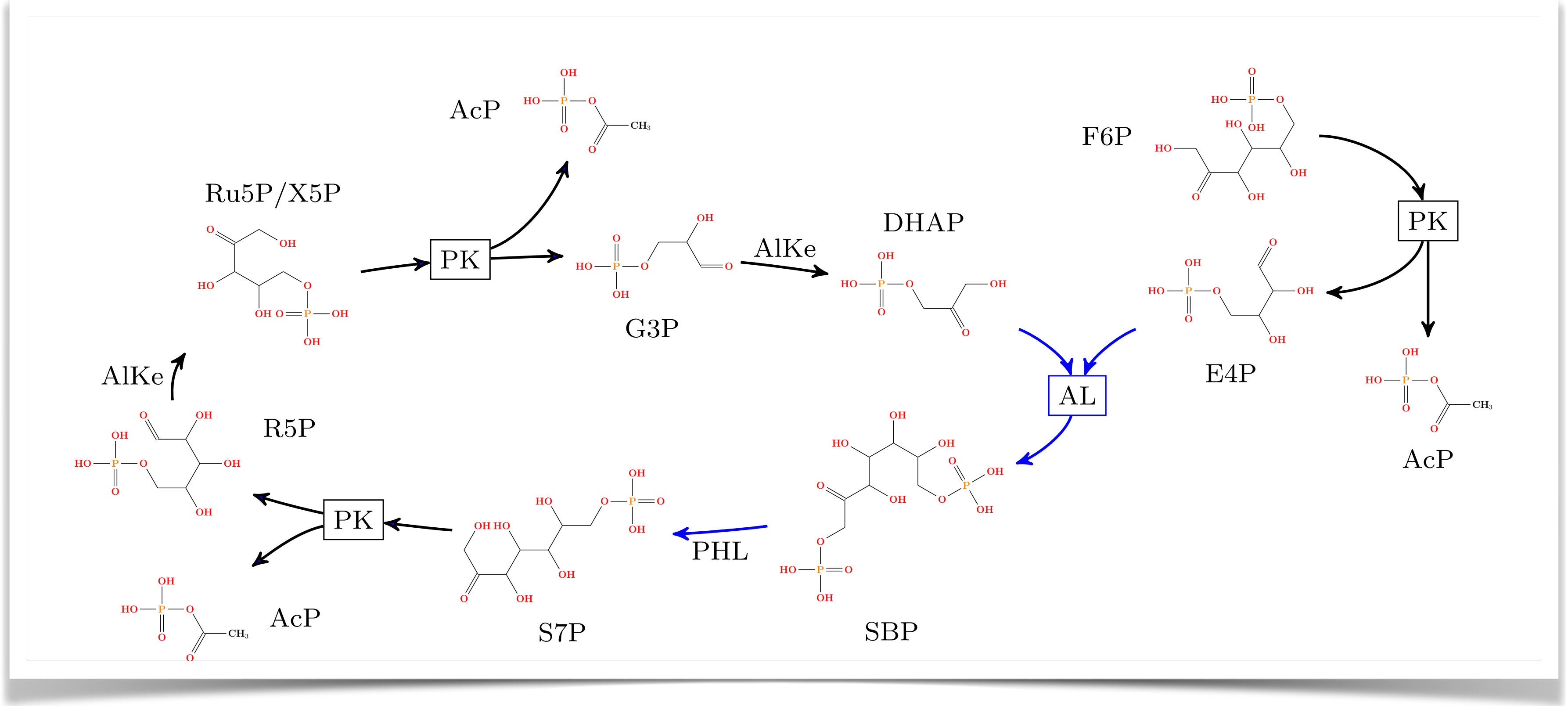
Rewriting in the life sciences: bio- and chemo-informatics

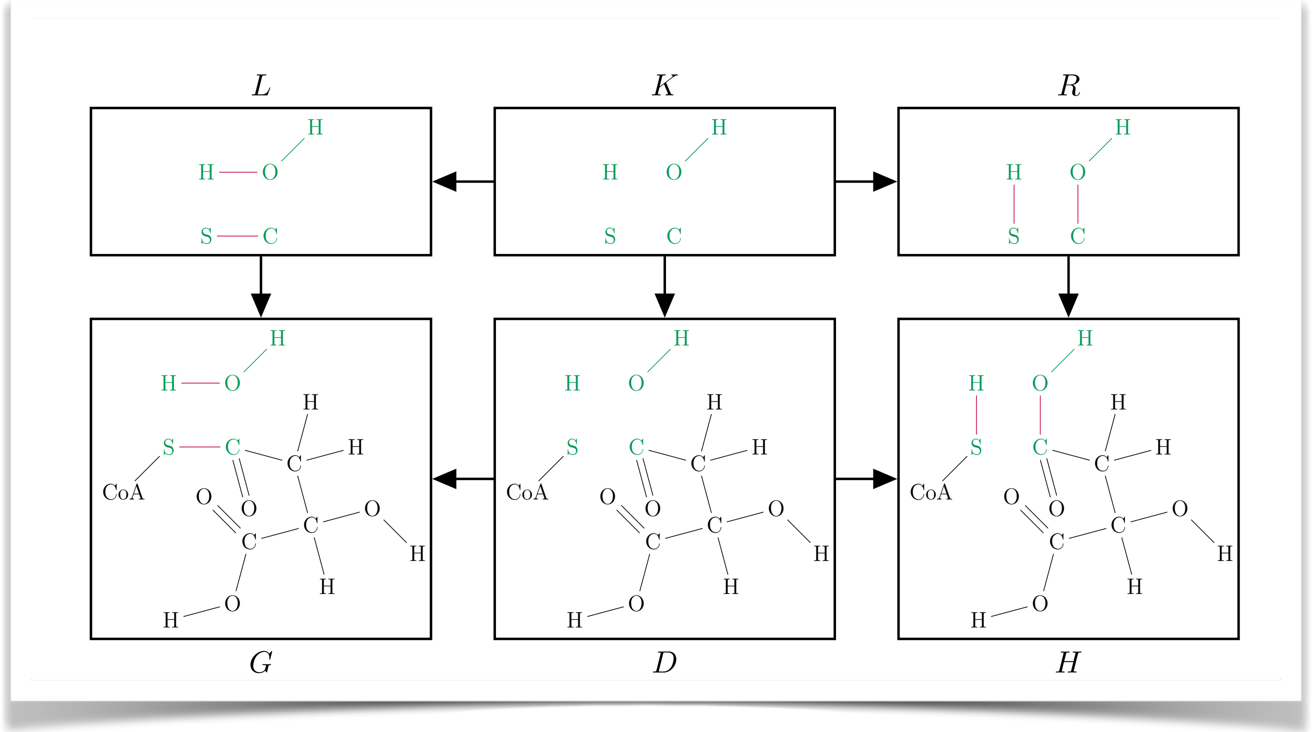
KAPPA





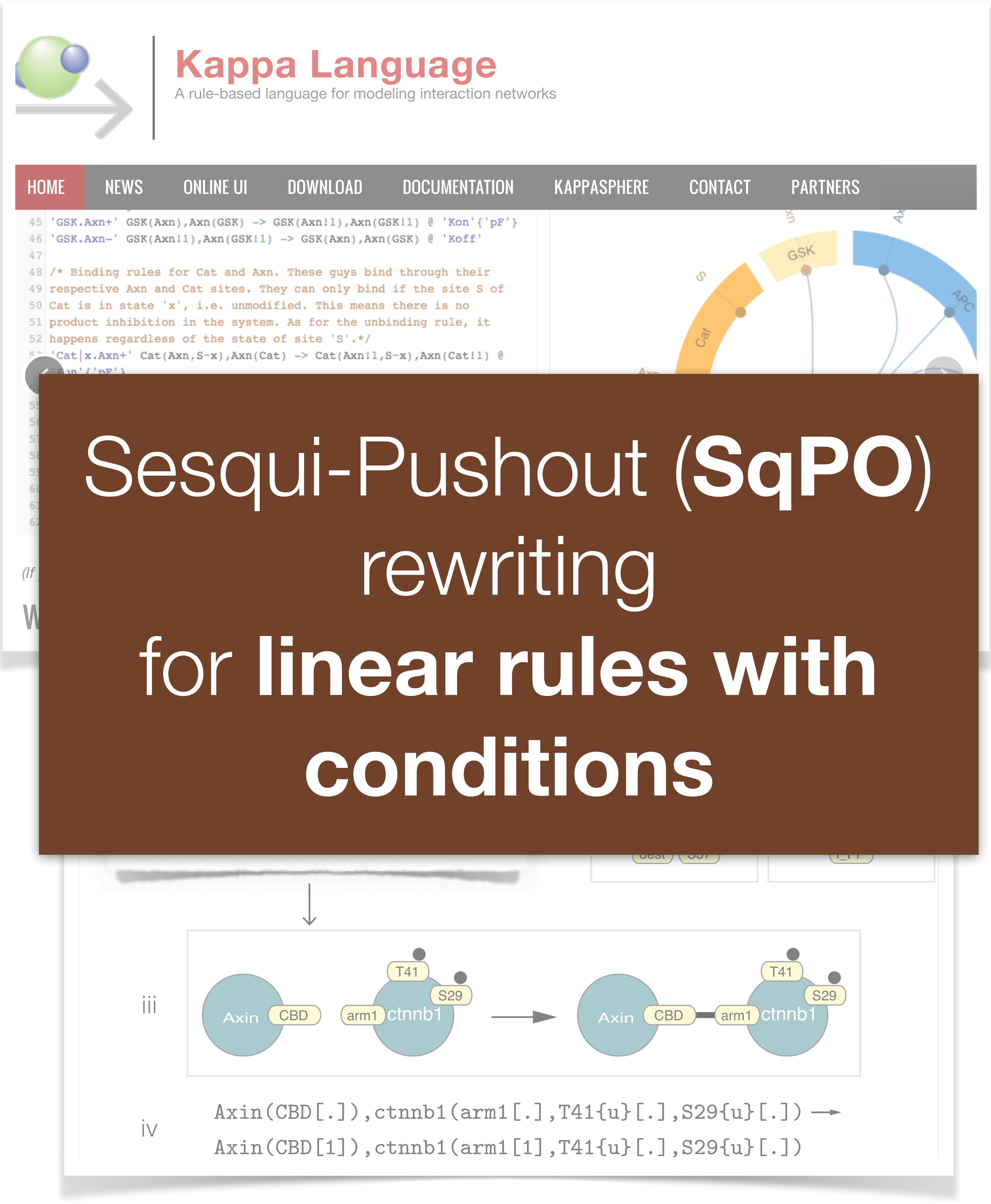
MØD





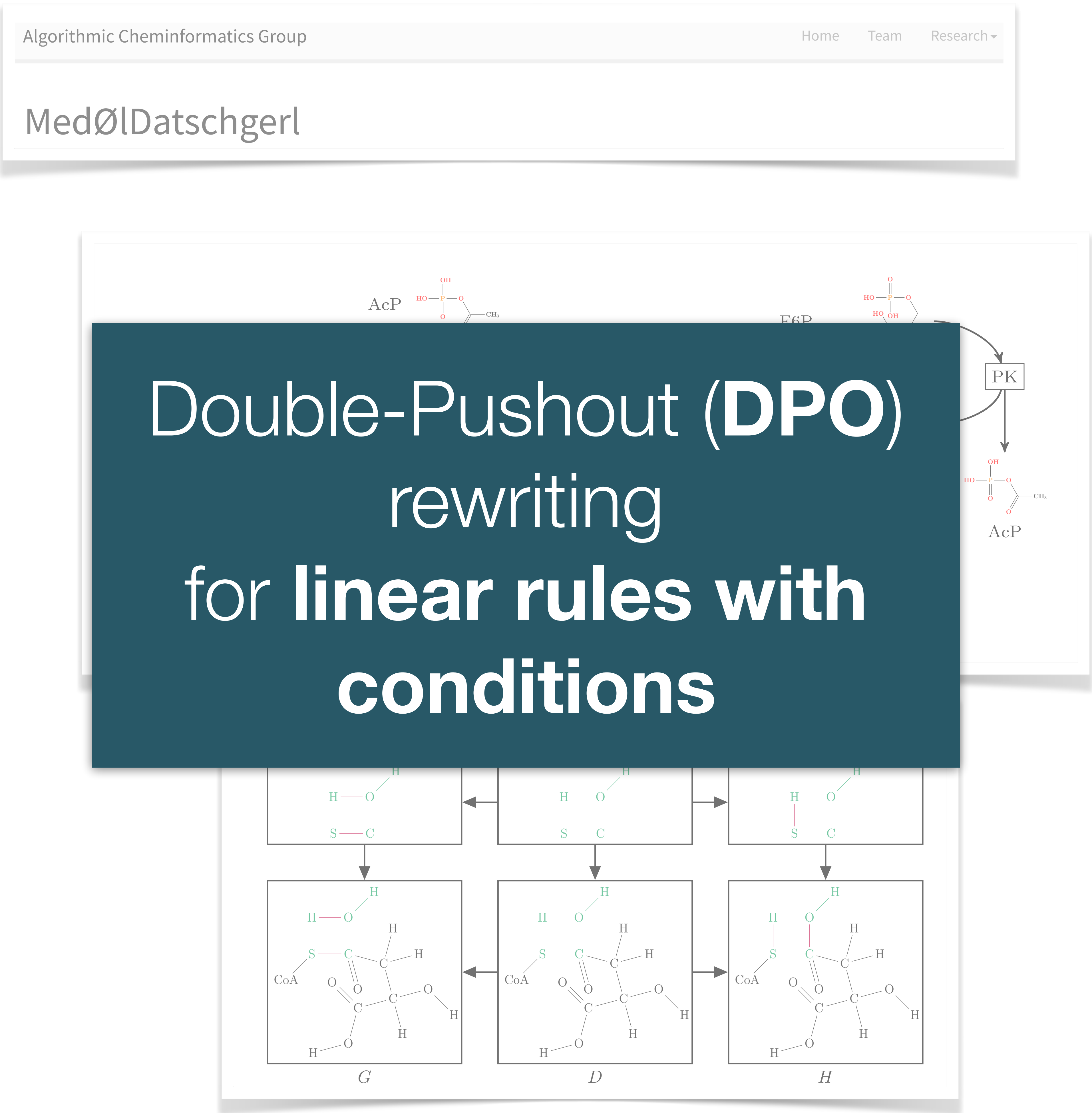
Rewriting in the life sciences: **bio-** and **chemo-informatics**

KAPPA



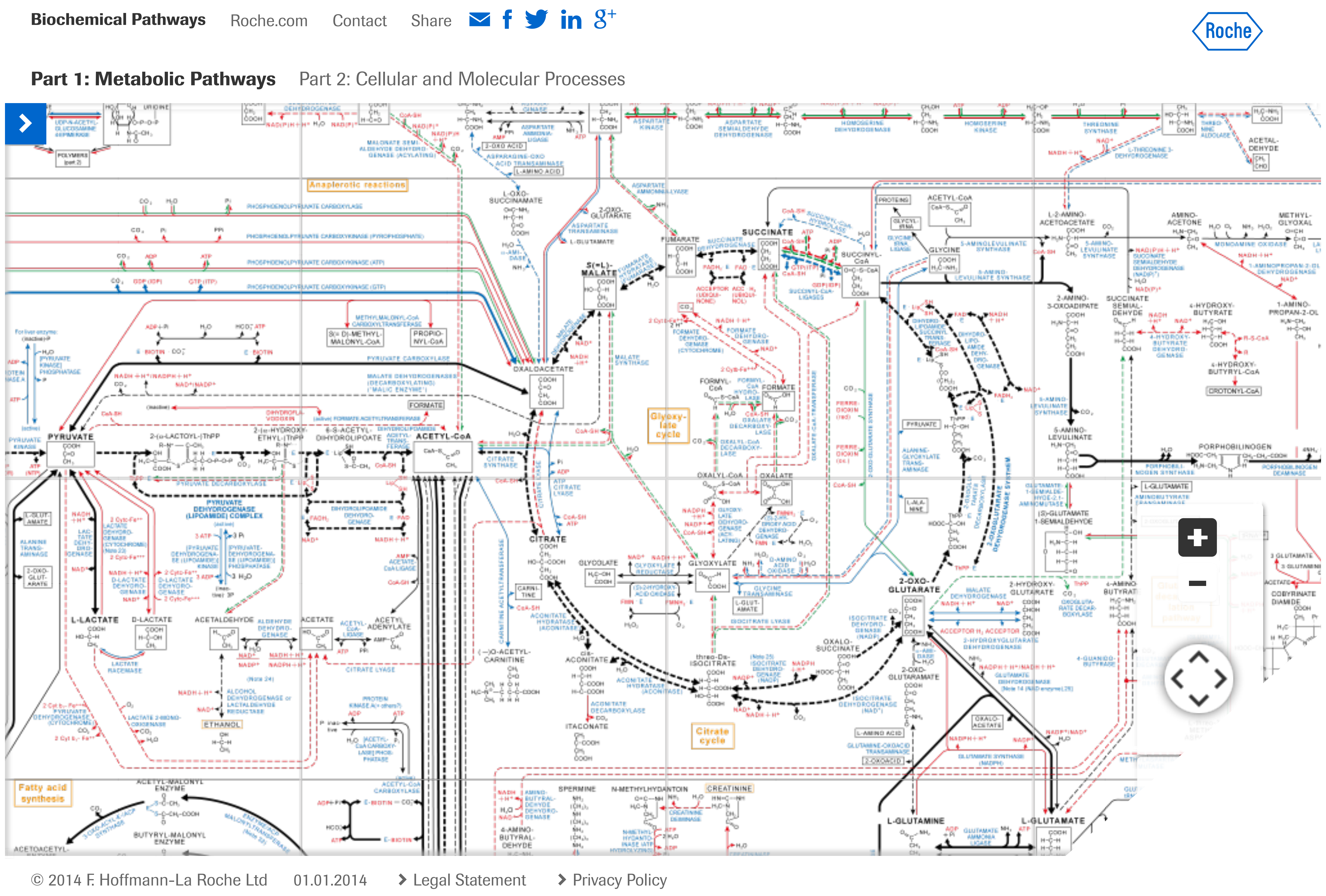
Sesqui-Pushout (**SqPO**) rewriting for **linear rules with conditions**

MØD



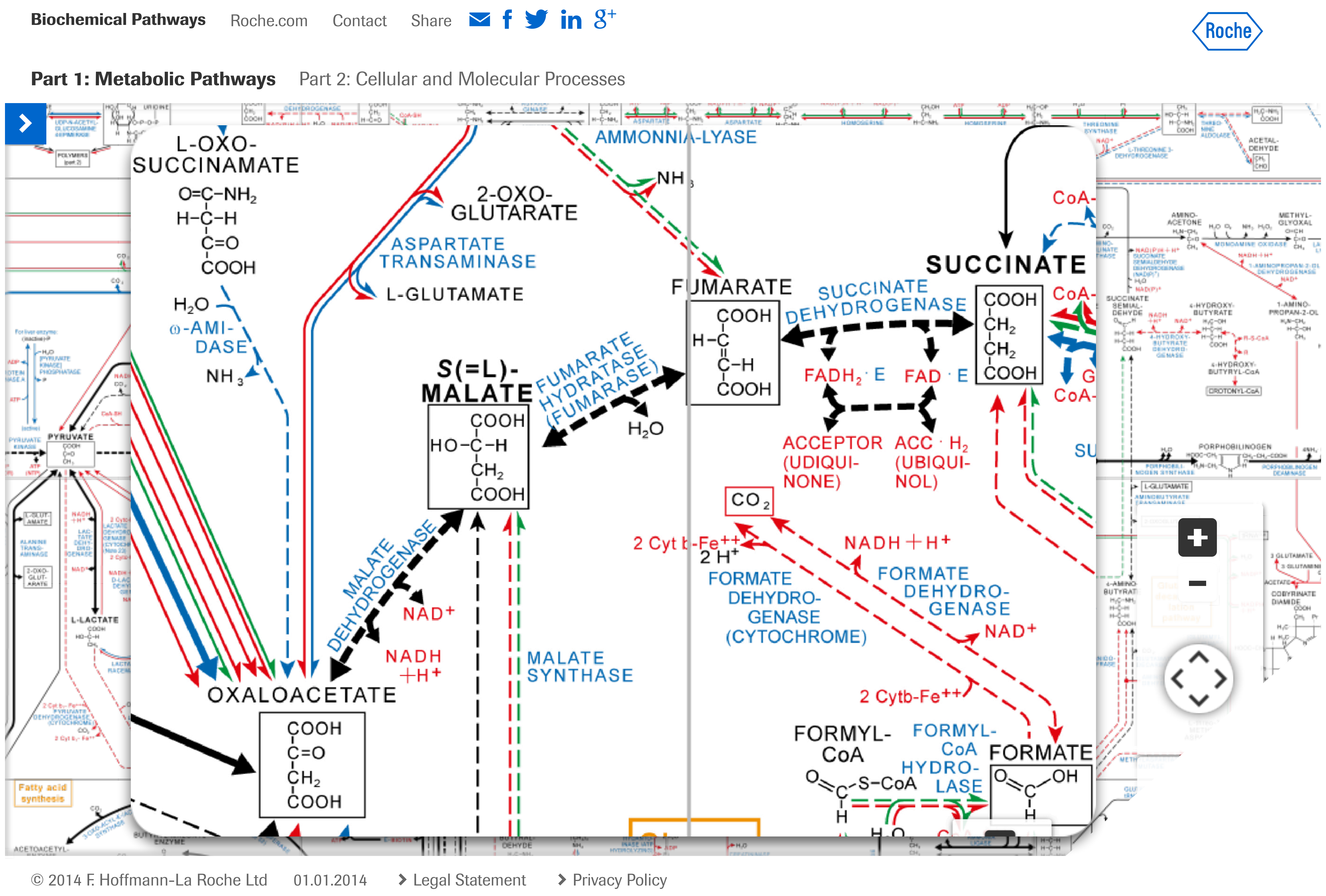
Double-Pushout (**DPO**) rewriting for **linear rules with conditions**

A fundamental challenge: **causal pathway dynamics**



http://biochemical-pathways.com

A fundamental challenge: **causal pathway dynamics**

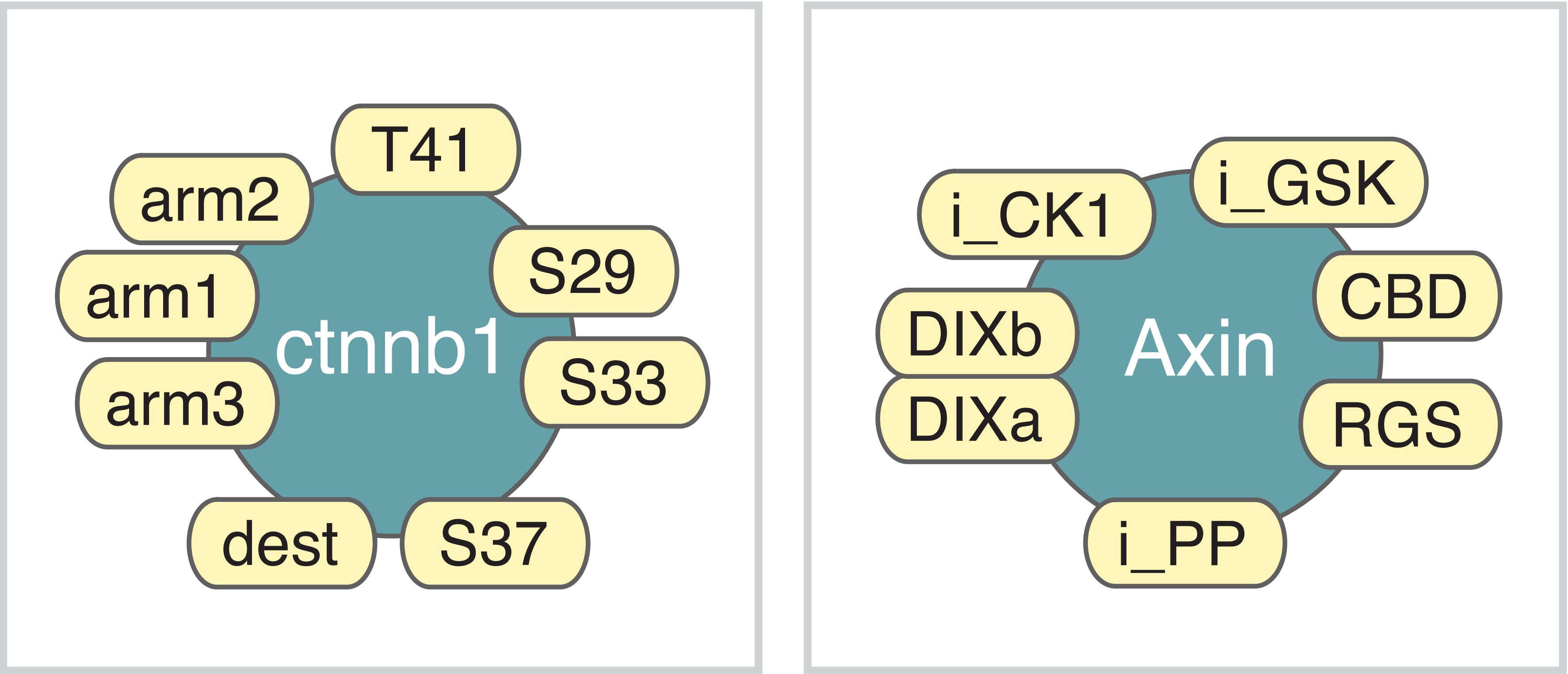


http://biochemical-pathways.com

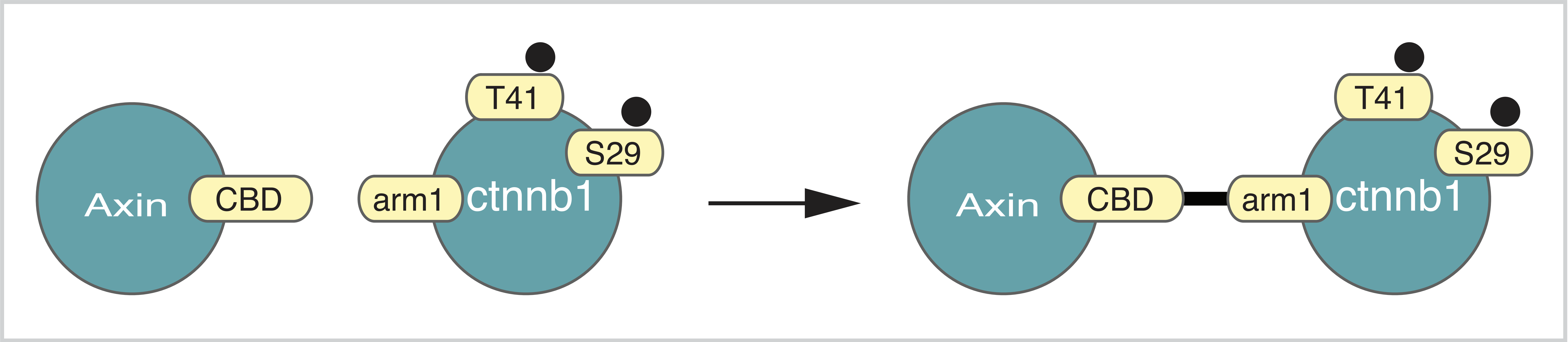
A fundamental challenge: **causal pathway dynamics**

i

"Axin binds a region in the armadillo repeat of β-catenin, if β-catenin is unphosphorylated at T41 and S29."



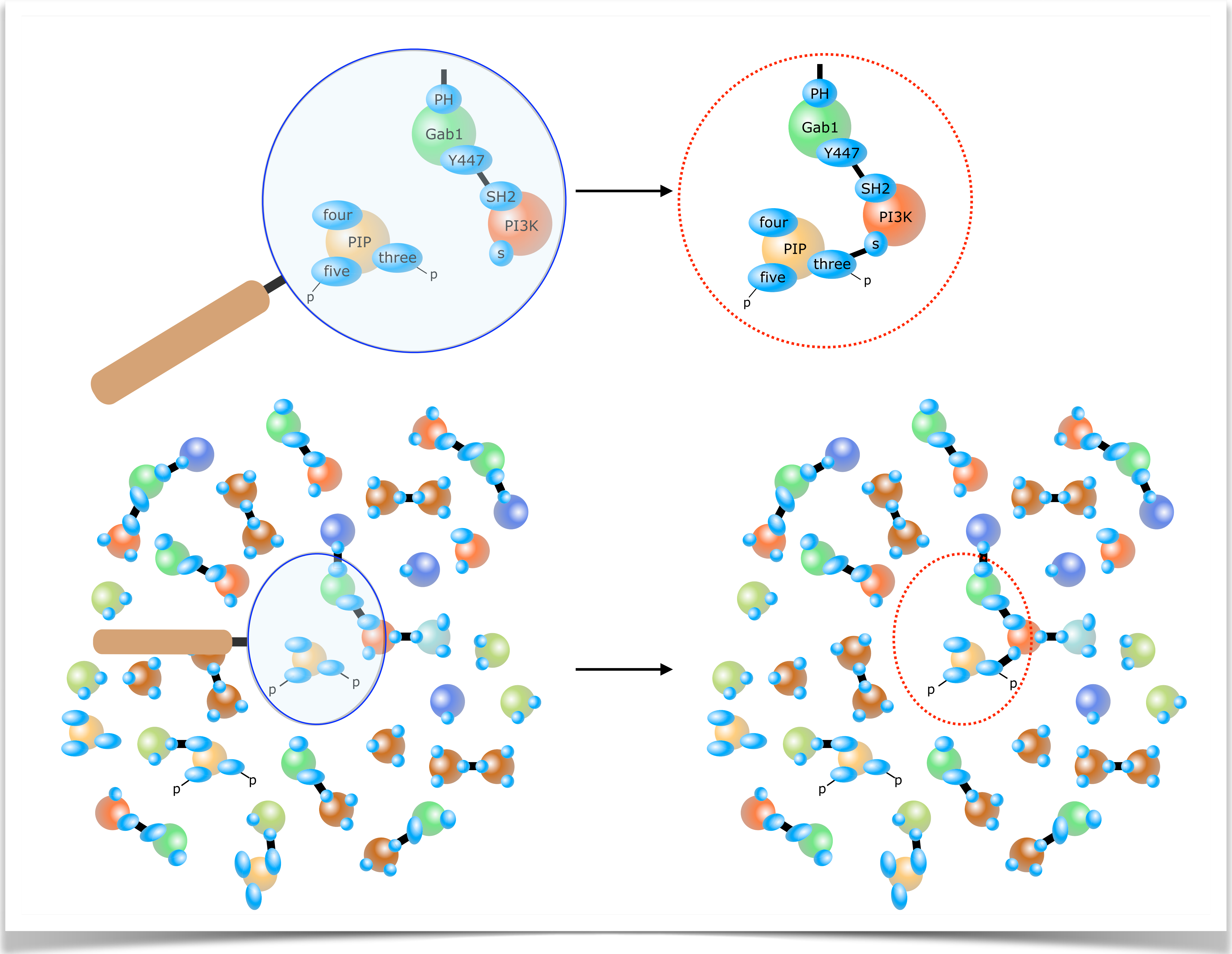
iii



iv

```
Axin(CBD[.]),ctnnb1(arm1[.],T41{u}[.],S29{u}[.]) ⟶
Axin(CBD[1]),ctnnb1(arm1[1],T41{u}[.],S29{u}[.])
```

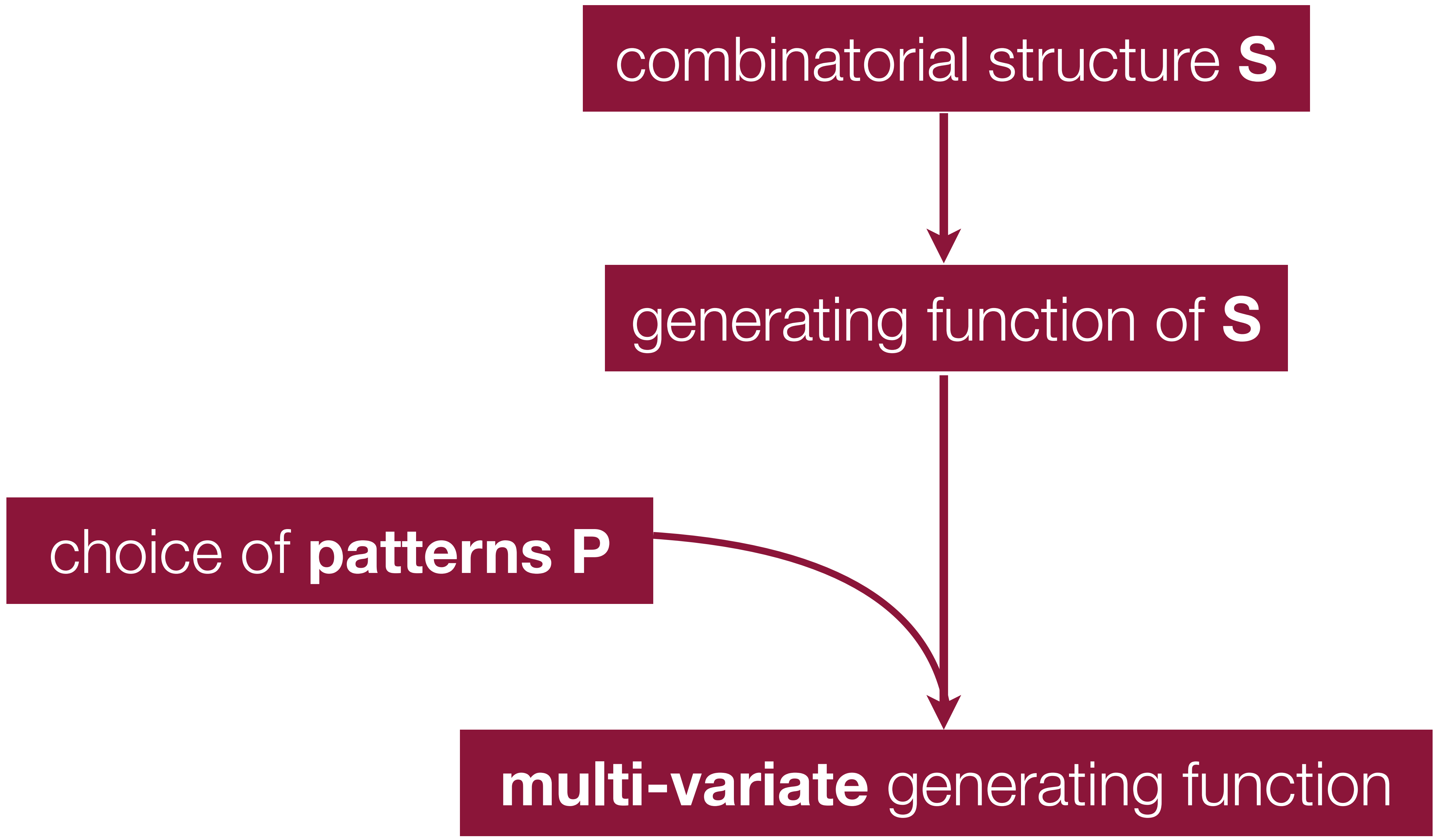
A fundamental challenge: **causal pathway dynamics**



source: *The Kappa platform for rule-based modeling*, Boutillier, P., Maasha, M., Li, X., Medina-Abarca, H.F., Krivine, J., Feret, J., Cristescu, I., Forbes, A.G. and Fontana, W., 2018, Bioinformatics, 34(13), pp.i583-i592.

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!

Plan of the talk

I. **Categorical Rewriting Theory**

II. **Rule Algebra Framework**

III. **Rule-Algebraic Combinatorics**

Main references and further reading

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.

1 Introduction

An important aspect of the standard theory of continuous-time Markov chains (CTMCs) [23] concerns the well-known fact that the CTMC semantics may be equivalently described via a pair of discrete-time Markov chains (DTMCs), where the so-called *embedded DTMC* encodes the probabilities for each of the possible transitions, and with the second DTMC encoding the jump-times for the transitions. This feature permits to design algorithms for *simulating* CTMCs, for instance in the form of Gillespie's stochastic simulation algorithms for chemical reaction systems [20], but in particular also in several variations for the simulation of stochastic rewriting systems, such as via the *KaSim* simulation engine of the Kappa platform [13]. The main contribution of the present paper consists in uncovering a hitherto unknown intimate relationship between three types of *moment generating functions* that are constructible from the data that specifies a stochastic rewriting system, and for a chosen set of *pattern count observables*: those of the CTMC itself, those of the embedded DTMC, and those of the (weighted) combinatorial species generated by the rewriting rules.

2 Prerequisite: the rule algebra framework

The methodology developed in the present paper relies heavily upon the mathematical formalism introduced in [2, 3, 9, 4, 7, 10], yet due to space restrictions, we will only provide some notations and essential definitions here, inviting the interested readers to consult loc. cit. for the full technical details.

2.1 DPO- and SqPO-type compositional rewriting semantics

Throughout this paper, we will consider categorical rewriting theories over categories that satisfy the following sets of properties (with DPO- and SqPO-semantics to be introduced below)[1]:

*The author would like to thank Paul-André Melliès and Noam Zeilberger for fruitful discussions and valuable feedback.
[1] We invite the readers to consult [6] or [7] for compact accounts of the relevant technical definitions of $\mathcal{M}$-adhesive categories, pullbacks, pushouts, pushout complements, final pullback complements and their respective properties.

Submitted to:
TERMGRAPH 2020

Compositionality of Rewriting Rules with Conditions

Nicolas Behr and Jean Krivine
Université de Paris, CNRS, IRIF, F-75006, Paris, France

We extend the notion of compositional associative rewriting as recently studied in the rule algebra framework literature to the setting of rewriting rules with conditions. Our methodology is category-theoretical in nature, where the definition of rule composition operations is encoding the non-deterministic sequential concurrent application of rules in Double-Pushout (DPO) and Sesqui-Pushout (SqPO) rewriting with application conditions based upon $\mathcal{M}$-adhesive categories. We uncover an intricate interplay between the category-theoretical concepts of conditions on rules and morphisms, the compositionality and compatibility of certain shift and transport constructions for conditions, and thirdly the property of associativity of the composition of rules.

1 Introduction and relation to previous work

Graph rewriting has emerged as a powerful formalism to represent complex systems whose dynamics can be captured by a finite set of rules. The *rule-based modeling* approach, originally introduced by V. Danos and C. Laneve in the early 2000's [32–34], has developed into one of the main frameworks for the study of biochemical reaction systems (in the form of the two main frameworks Kappa [20, 29] and BioNetGen[18, 50]). The approach proposes to model protein-protein interaction networks using graph rewriting models, in which proteins are the vertices of a graph whose connected components denote molecular complexes. As formal methods are expending in the molecular biology community, it is expected that large models describing signaling pathways and self-assembling processes occurring in the cell will be commonplace in a near future.

While the algorithmic aspects of graph rewriting are well-studied, programming language approaches to modeling with graphs are to date still a comparatively underdeveloped topic. Contrary to classical term rewriting, the notion of a *match* of a graph rewriting rule and its *effects* on a term (a graph) is subject to various definitions, allowing more of less control over possible rewrites. In addition, the mere nature of the graphs that are being rewritten impacts both the algorithmic design and expressiveness of graph rewriting. Category theory is a practical toolkit for equipping graphs with well-defined operational semantics. *Double-Pushout (DPO) rewriting* [27] is a popular technique, partly because it does not yield side effects when rules are applied (which makes it amenable to static analysis for instance). However, when a graph rewriting rule entails node deletion, DPO semantics will not allow a match of such a rule to trigger if the node that is deleted is connected outside the domain of the match (which would yield side effect). This has limited the practicality of DPO semantics in the context of biological modeling, where more permissive techniques have been employed. *Sesqui-Pushout (SqPO) rewriting* [26] in particular is the technique that is used to rewrite Kappa graphs [29], one of the main graph rewriting formalisms for biological models.

Quite orthogonal to the issue of defining rule matches and effects, having access to a fine-grained control over rule triggering is a key issue when graph rewriting is used as a modeling language. To this aim, graph rewriting rules have been equipped with *application conditions* [38, 47], which can be seen as constraints that need to be checked "on the fly" when a rewrite rule is applied.

This paper presents a *compositional* variant of DPO and SqPO-type rewriting for rules with conditions in a very general category-theoretical setting. From a mathematical perspective, while

Nicolas Behr: nicolas.behr@irif.fr, http://nicolasbehr.com, corresponding author; the work of N.B. was supported by a *Marie Skłodowska-Curie Individual fellowship* (Grant Agreement No. 753750 – RaSiR)
Jean Krivine: jean.krivine@irif.fr, https://www.irif.fr/~jkrivine/homepage/Home.html

Rewriting Theory for the Life Sciences: A Unifying Theory of CTMC Semantics*

Nicolas Behr[1] [0000−0002−8738−5040] and Jean Krivine[2] [0000−0001−7261−7462]

[1] Center for Research and Interdisciplinarity (CRI)
Université de Paris, INSERM U1284
8-10 Rue Charles V, 75004 Paris, France
nicolas.behr@cri-paris.org
[2] Institut de Recherche en Informatique Fondamentale (IRIF)
Université de Paris, CNRS UMR 8243
8 Place Aurélie Nemours, 75205 Paris Cedex 13, France
jean.krivine@irif.fr

Abstract. The Kappa biochemistry and the MØD organo-chemistry frameworks are amongst the most intensely developed applications of rewriting theoretical methods in the life sciences to date. A typical feature of these types of rewriting theories is the necessity to implement certain structural constraints on the objects to be rewritten (a protein is empirically found to have a certain signature of sites, a carbon atom can form at most four bonds, . . .). In this paper, we contribute to the theoretical foundations of these types of rewriting theory a number of conceptual and technical developments that permit to implement a universal theory of continuous-time Markov chains (CTMCs) for stochastic rewriting systems. Our core mathematical concepts are a novel rule algebra construction for the relevant setting of rewriting rules with conditions, both in Double- and in Sesqui-Pushout semantics, augmented by a suitable stochastic mechanics formalism extension that permits to derive dynamical evolution equations for pattern-counting statistics.

Keywords: Double-Pushout rewriting · Sesqui-pushout rewriting · rule algebra · stochastic mechanics · biochemistry · organic chemistry.

1 Motivation

One of the key applications that rewriting theory may be considered for in the life sciences is the theory of continuous-time Markov chains (CTMCs) modeling complex systems. In fact, since Delbrück's seminal work on autocatalytic reaction systems in the 1940s [20], the mathematical theory of chemical reaction systems has effectively been formulated as a rewriting theory in disguise, namely via the rule algebra of discrete graph rewriting [11]. In the present paper, we provide the necessary technical constructions in order to consider

* An extended version of this paper containing additional technical appendices is available online [9].

Main references and further reading

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.

Main references and further reading

Compositionality of Rewriting Rules with Conditions

Nicolas Behr and Jean Krivine

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

We extend the notion of compositional associative rewriting as recently studied in the rule algebra framework literature to the setting of rewriting rules with conditions. Our methodology is category-theoretical in nature, where the definition of rule composition operations is encoding the non-deterministic sequential concurrent application of rules in Double-Pushout (DPO) and Sesqui-Pushout (SqPO) rewriting with application conditions based upon $\mathcal{M}$-adhesive categories. We uncover an intricate interplay between the category-theoretical concepts of conditions on rules and morphisms, the compositionality and compatibility of certain shift and transport constructions for conditions, and thirdly the property of associativity of the composition of rules.

Main references and further reading

Rewriting Theory for the Life Sciences: A Unifying Theory of CTMC Semantics*

Nicolas Behr[1] ✉ [0000−0002−8738−5040] and Jean Krivine[2] [0000−0001−7261−7462]

[1] Center for Research and Interdisciplinarity (CRI)
Université de Paris, INSERM U1284
8-10 Rue Charles V, 75004 Paris, France
nicolas.behr@cri-paris.org

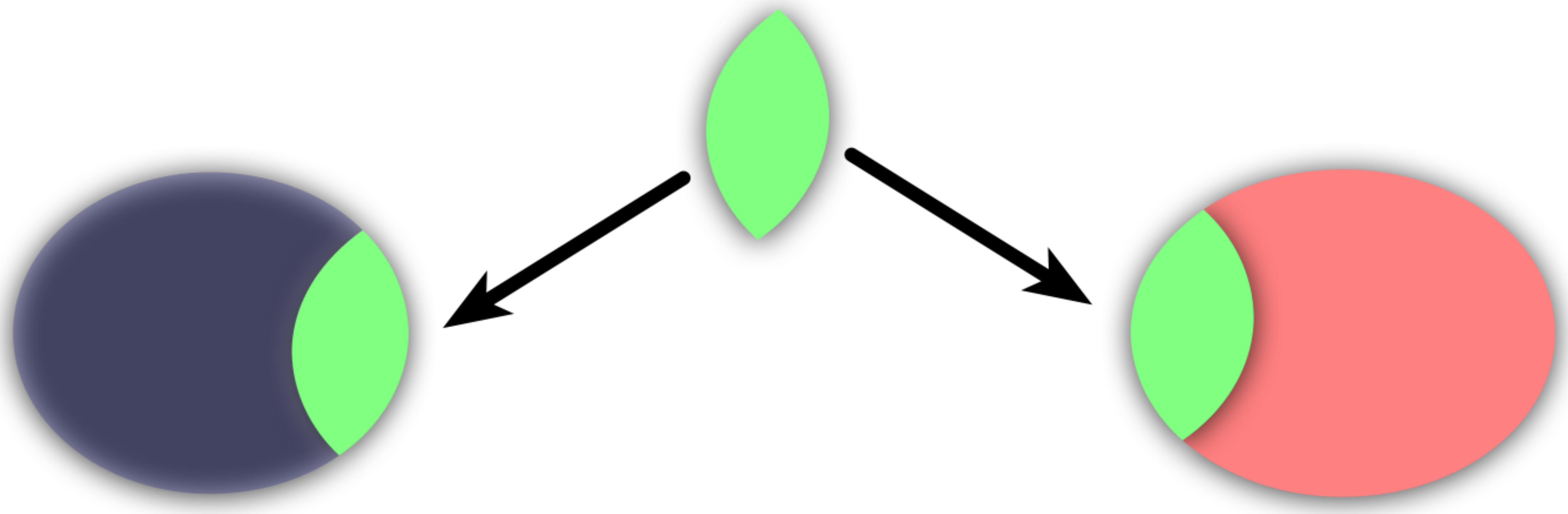
[2] Institut de Recherche en Informatique Fondamentale (IRIF)
Université de Paris, CNRS UMR 8243
8 Place Aurélie Nemours, 75205 Paris Cedex 13, France
jean.krivine@irif.fr

I. Categorical Rewriting Theory

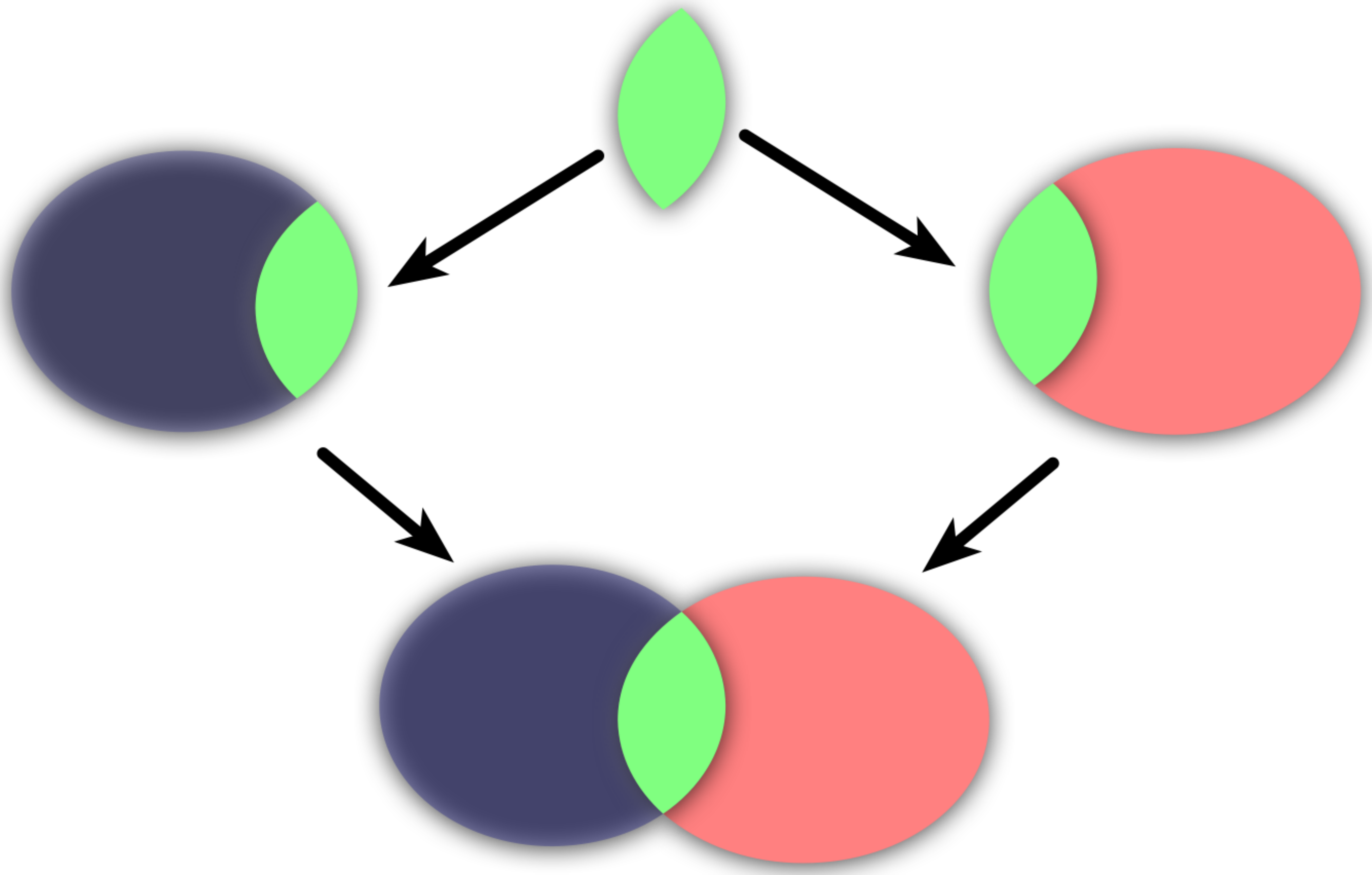
II. Rule Algebra Framework

III. Rule-Algebraic Combinatorics

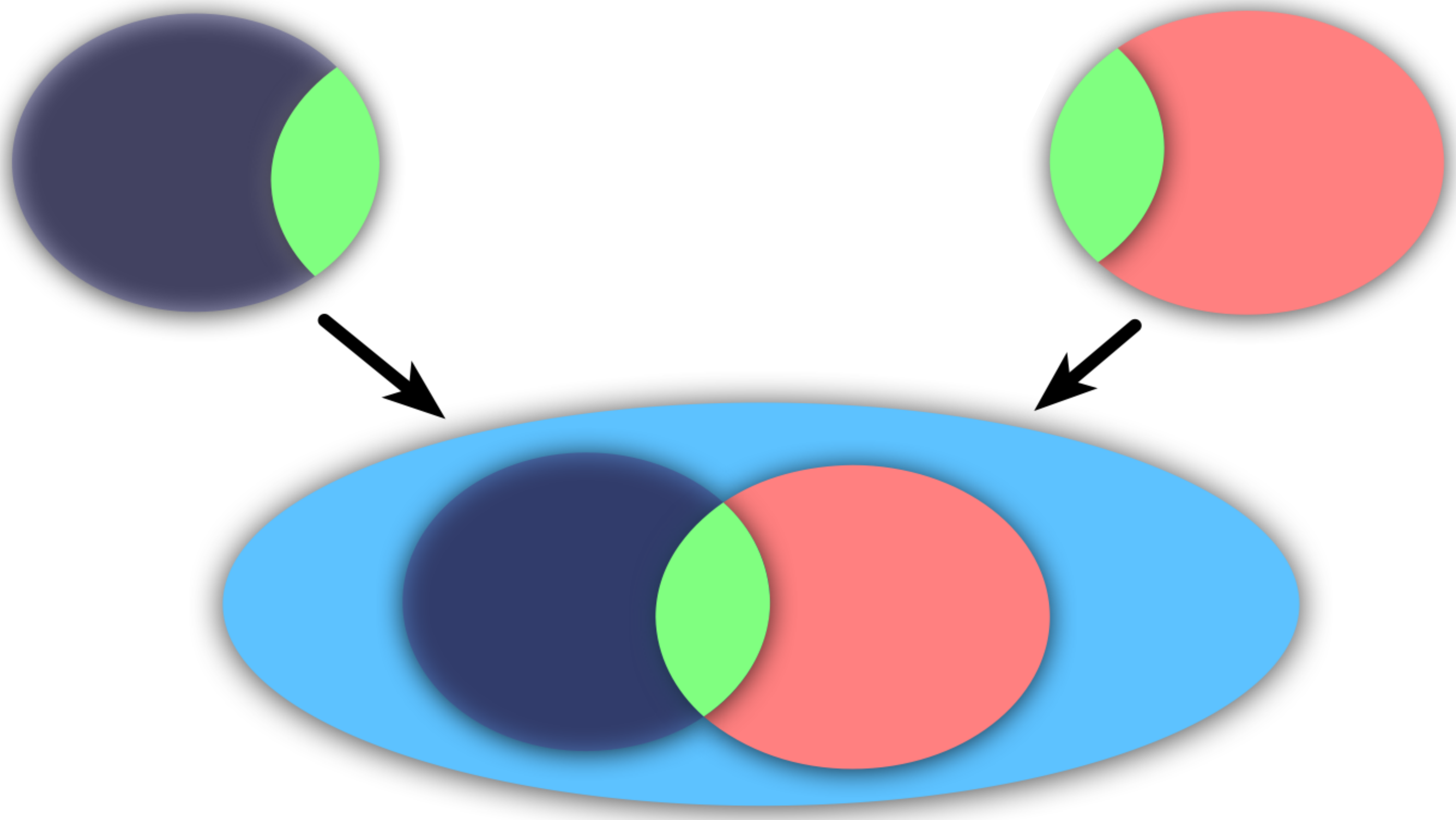
Set **union**



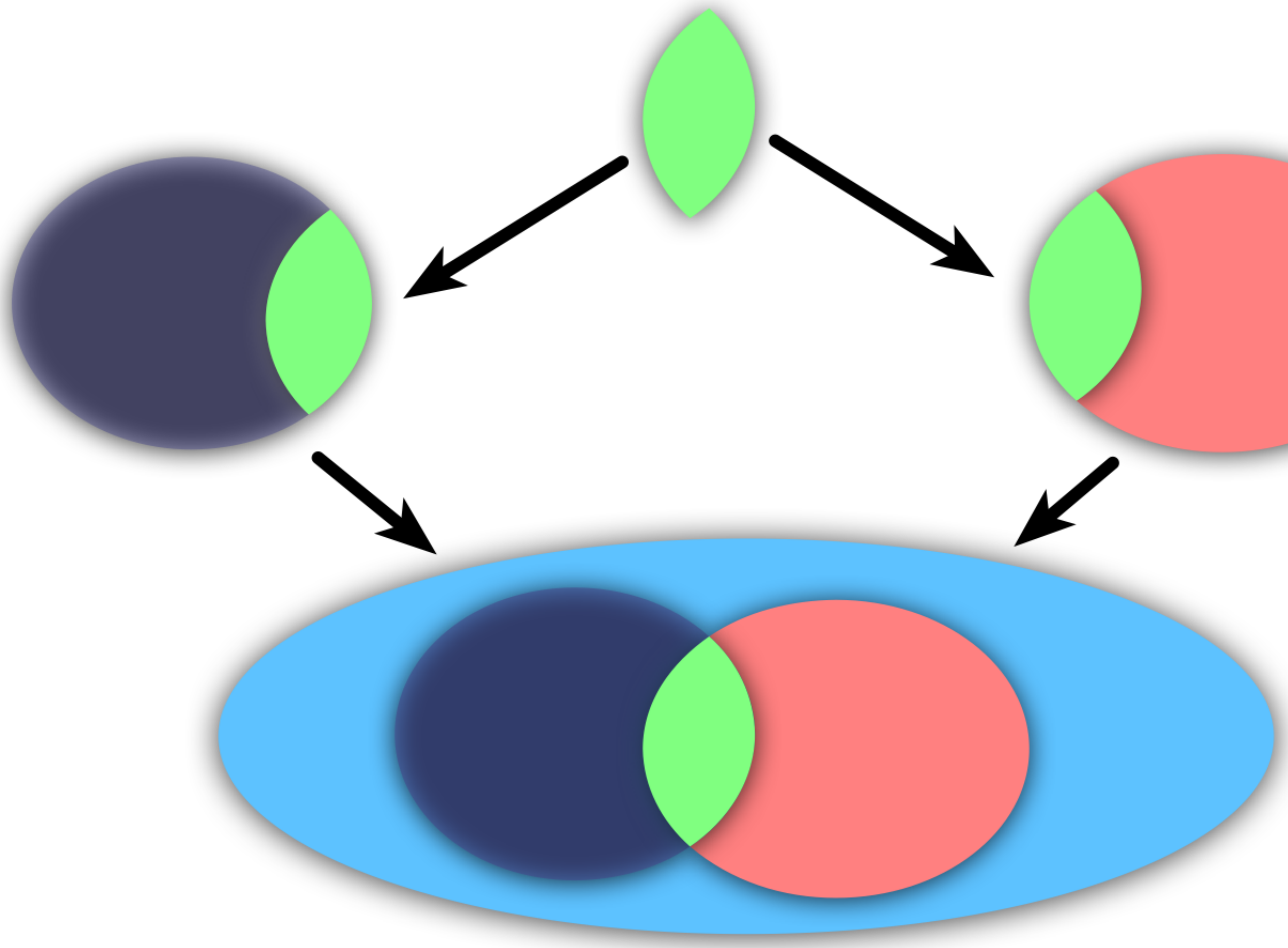
Set **union**



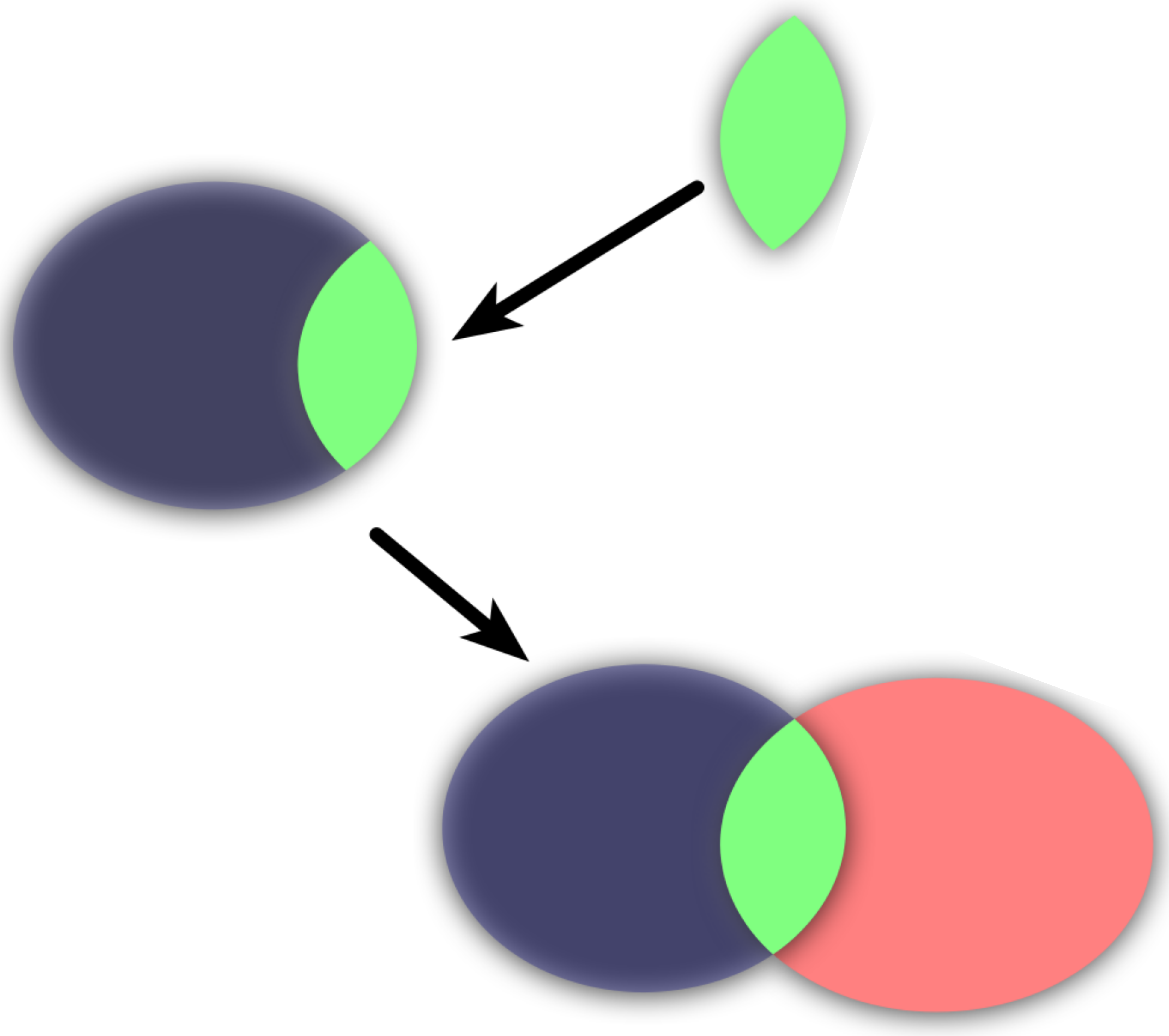
Set **intersection** (of two sets within another set)



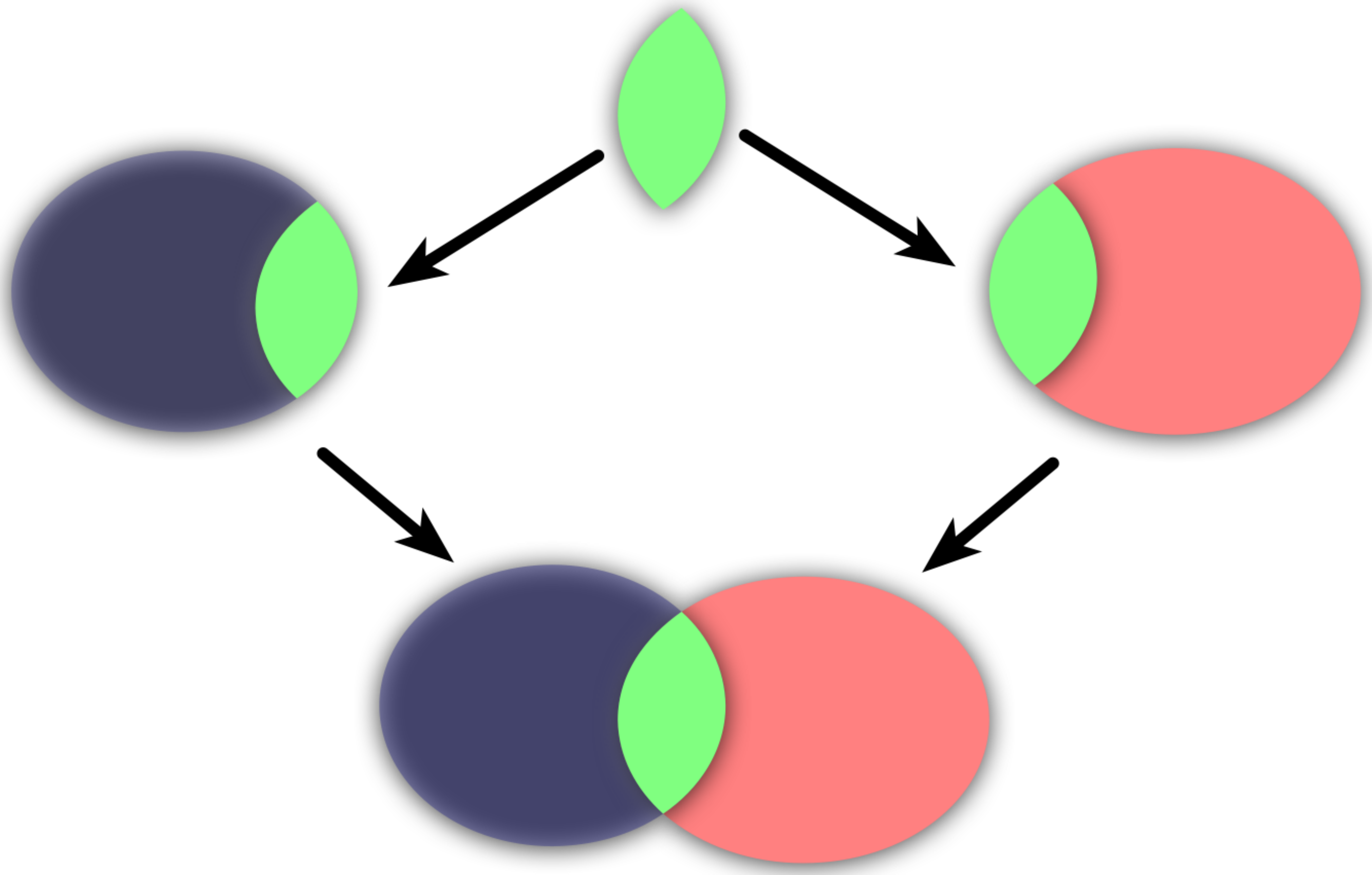
Set **intersection** (of two sets within another set)



Set **(relative) complement**

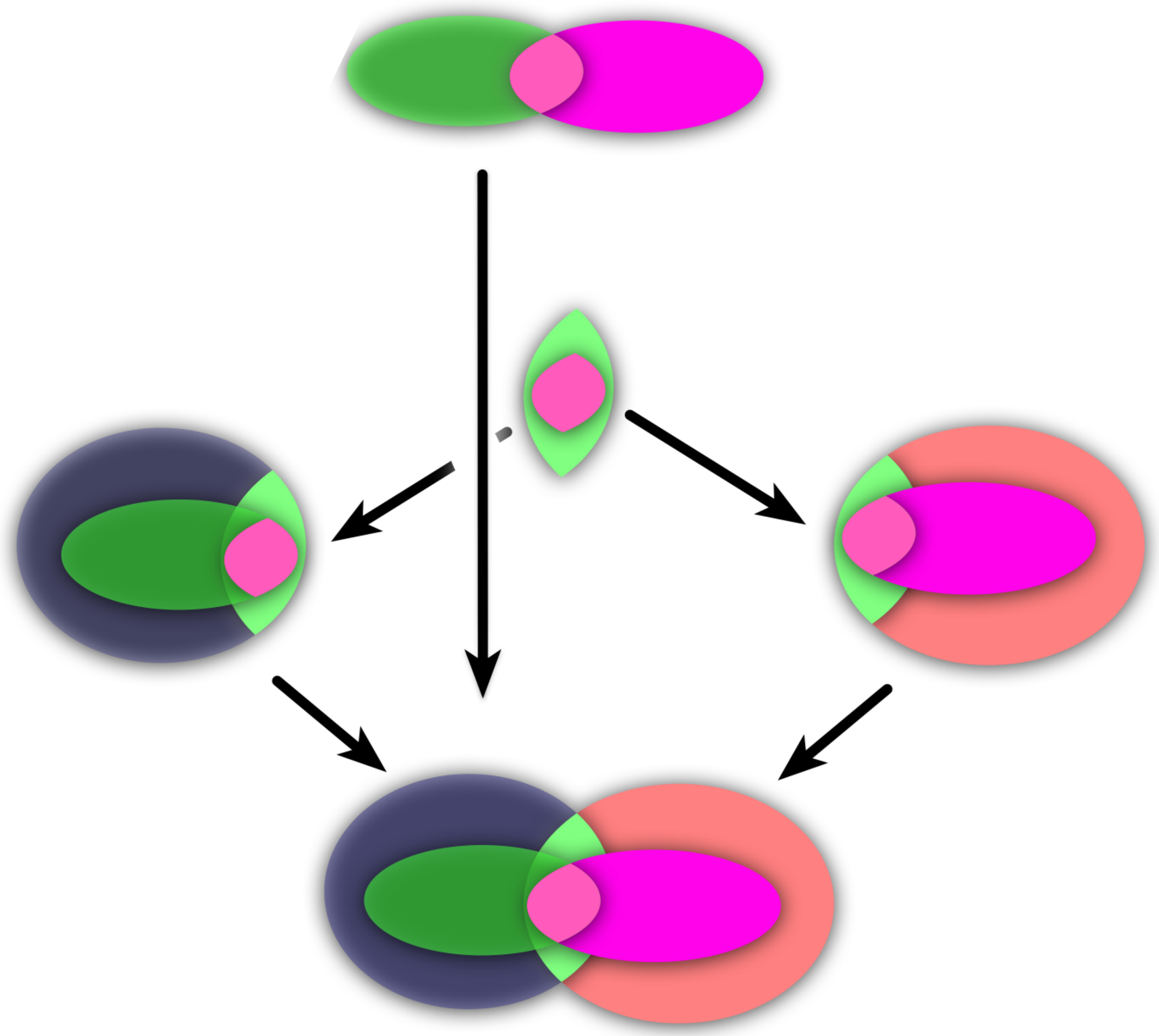


Set **(relative) complement**

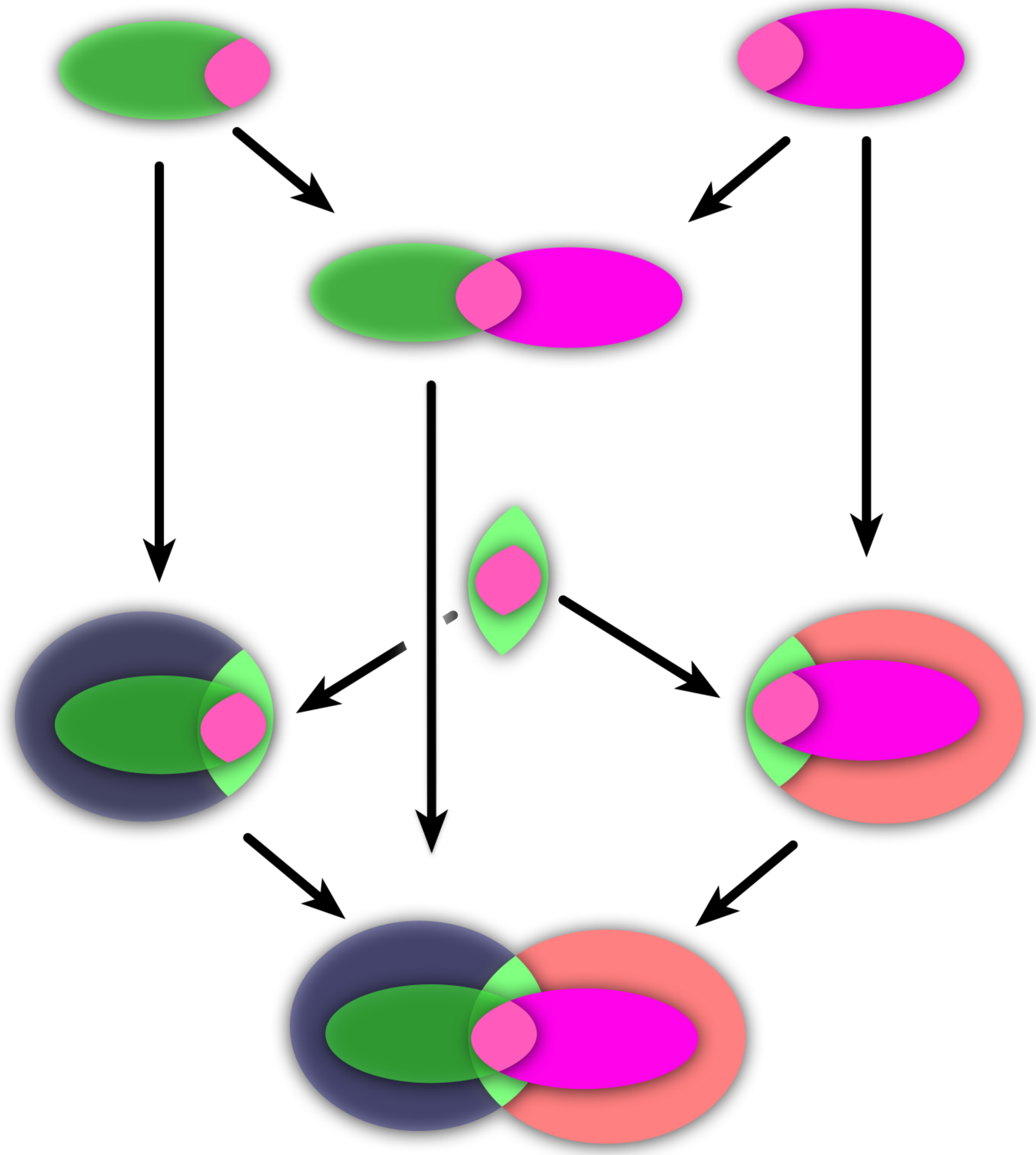


Van Kampen property (*Lack & Sobocinski 2003*)

If the bottom square is a **pushout**



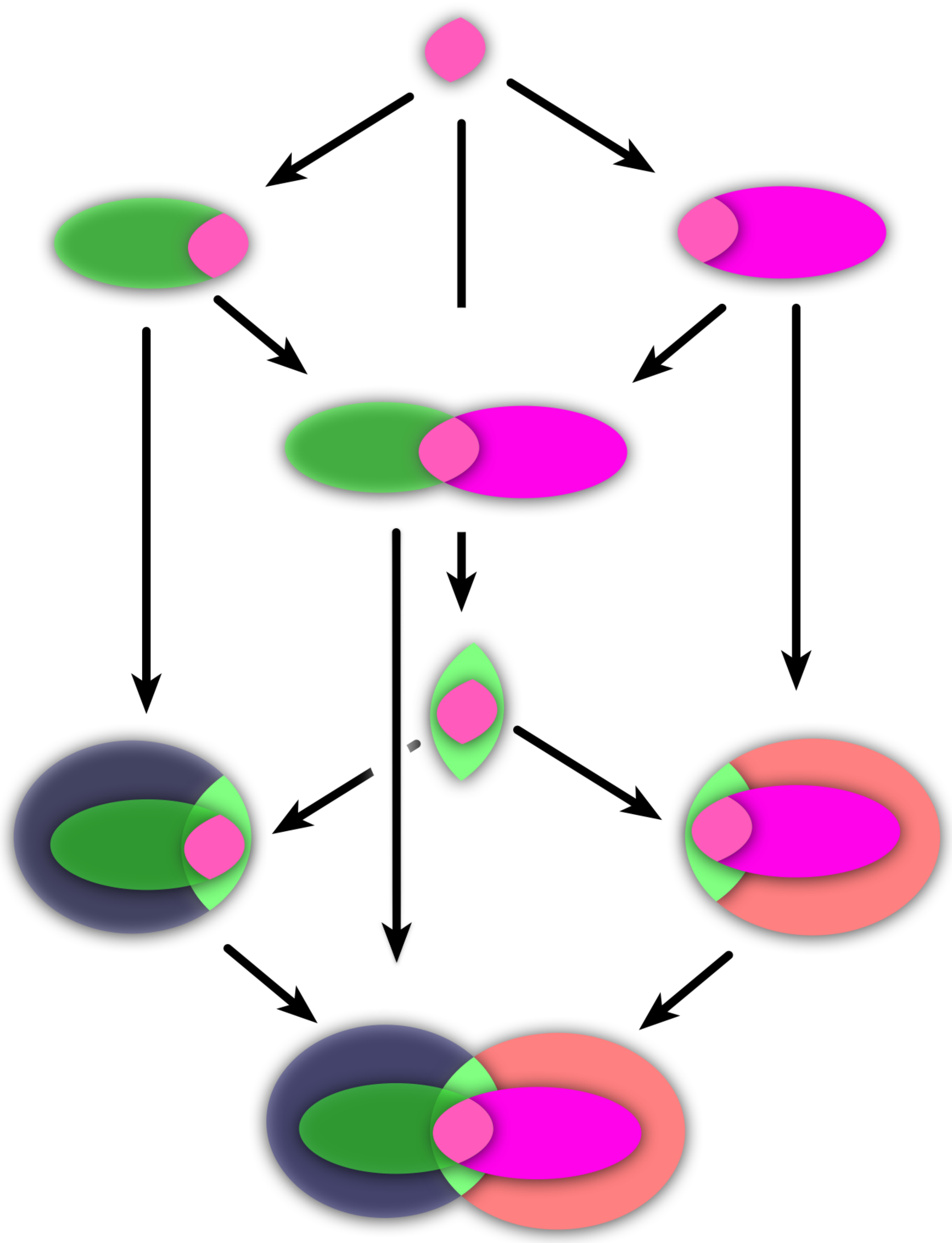
Van Kampen property (*Lack & Sobocinski 2003*)



If the bottom square is a **pushout** and the front squares are **pullbacks**…

… then the bottom square is a **van Kampen square**, i.e. the following property holds:

Van Kampen property (*Lack & Sobocinski 2003*)



If the bottom square is a **pushout** and the front squares are **pullbacks**…

… then the bottom square is a **van Kampen square**, i.e. the following property holds:

The back squares are pullbacks if and only if the top square is a pushout.

The basic prerequisites for **category-theoretical rewriting theories**

The basic prerequisites for **category-theoretical rewriting theories**

A category **C** is **adhesive** if

1. **C** has **all pullbacks**
2. **C** has **pushouts along monomorphisms**
3. Pushouts along monomorphisms are **van Kampen squares**.

S. Lack & P. Sobociński **(2005)**. *Adhesive and quasiadhesive categories.* RAIRO-Theoretical Informatics and Applications, 39(3), 511-545.

The basic prerequisites for **category-theoretical rewriting theories**

A category **C** is **adhesive** if

1. **C** has **all pullbacks**
2. **C** has **pushouts along monomorphisms**
3. Pushouts along monomorphisms are **van Kampen squares**.

S. Lack & P. Sobociński **(2005)**. *Adhesive and quasiadhesive categories.* RAIRO-Theoretical Informatics and Applications, 39(3), 511-545.

A category **C** is **finitary** if every object $X \in \mathrm{ob}(\mathbf{C})$ has only **finitely many subobjects** (up to isomorphism).

K. Gabriel, B. Braatz, H. Ehrig & U. Golas **(2014)**. *Finitary M-adhesive categories.* Mathematical Structures in Computer Science, 24(4).

The basic prerequisites for **category-theoretical rewriting theories**

A category **C** is **adhesive** if

1. **C** has **all pullbacks**
2. **C** has **pushouts along monomorphisms**
3. Pushouts along monomorphisms are **van Kampen squares**.

S. Lack & P. Sobociński **(2005)**. *Adhesive and quasiadhesive categories.* RAIRO-Theoretical Informatics and Applications, 39(3), 511-545.

A category **C** is **finitary** if every object $X \in \mathrm{ob}(\mathbf{C})$ has only **finitely many subobjects** (up to isomorphism).

K. Gabriel, B. Braatz, H. Ehrig & U. Golas **(2014)**. *Finitary M-adhesive categories.* Mathematical Structures in Computer Science, 24(4).

A category **C** possesses a **strict initial object** $\varnothing \in |\mathbf{C}|$ (the "*empty object*") if

1. $\forall X \in \mathrm{ob}(\mathbf{C}) : \exists!(\iota_X : \varnothing \hookrightarrow X) \in \mathrm{mono}(\mathbf{C})$
2. $\forall X \in \mathrm{ob}(\mathbf{C}) : \exists(X \to \varnothing) \;\Rightarrow\; X \cong \varnothing$

Example: **presheaves**

Definition: For $\mathbb{S}$ a (small) category, the category $\hat{\mathbb{S}}$ of **presheaves** over $\mathbb{S}$ has

- **objects** of $\hat{\mathbb{S}}$ are functors $F : \mathbb{S}^{op} \to \mathbb{SET}$
- **morphisms** of $\hat{\mathbb{S}}$ are natural transformations $\phi : F \dot{\to} G$

Example: **presheaves**

Definition: For $\mathbb{S}$ a (small) category, the category $\hat{\mathbb{S}}$ of **presheaves** over $\mathbb{S}$ has

- **objects** of $\hat{\mathbb{S}}$ are functors $F : \mathbb{S}^{op} \to \mathbb{SET}$
- **morphisms** of $\hat{\mathbb{S}}$ are natural transformations $\phi : F \dot{\to} G$

Special case: (directed) **multigraphs** as $\hat{\mathbb{G}}$, where $\mathbb{G} : V \underset{t}{\overset{s}{\rightrightarrows}} E$

Example: **presheaves**

Definition: For $\mathbb{S}$ a (small) category, the category $\hat{\mathbb{S}}$ of **presheaves** over $\mathbb{S}$ has

- **objects** of $\hat{\mathbb{S}}$ are functors $F : \mathbb{S}^{op} \to \mathbb{SET}$
- **morphisms** of $\hat{\mathbb{S}}$ are natural transformations $\phi : F \dot{\to} G$

Special case: (directed) **multigraphs** as $\hat{\mathbb{G}}$, where $\mathbb{G} : V \overset{s}{\underset{t}{\rightrightarrows}} E$

$\Rightarrow$ a **graph** G is given by the data $G(V)$ (set of ***vertices***), $G(E)$ (set of ***edges***) and two morphisms $G(s), G(t) : G(E) \to G(V)$ (***source/target*** *maps*)

Example: **presheaves**

Definition: For $\mathbb{S}$ a (small) category, the category $\hat{\mathbb{S}}$ of **presheaves** over $\mathbb{S}$ has

- **objects** of $\hat{\mathbb{S}}$ are functors $F : \mathbb{S}^{op} \to \mathbb{SET}$
- **morphisms** of $\hat{\mathbb{S}}$ are natural transformations $\phi : F \dot{\to} G$

Special case: (directed) **multigraphs** as $\hat{\mathbb{G}}$, where $\mathbb{G} : V \underset{t}{\overset{s}{\rightrightarrows}} E$

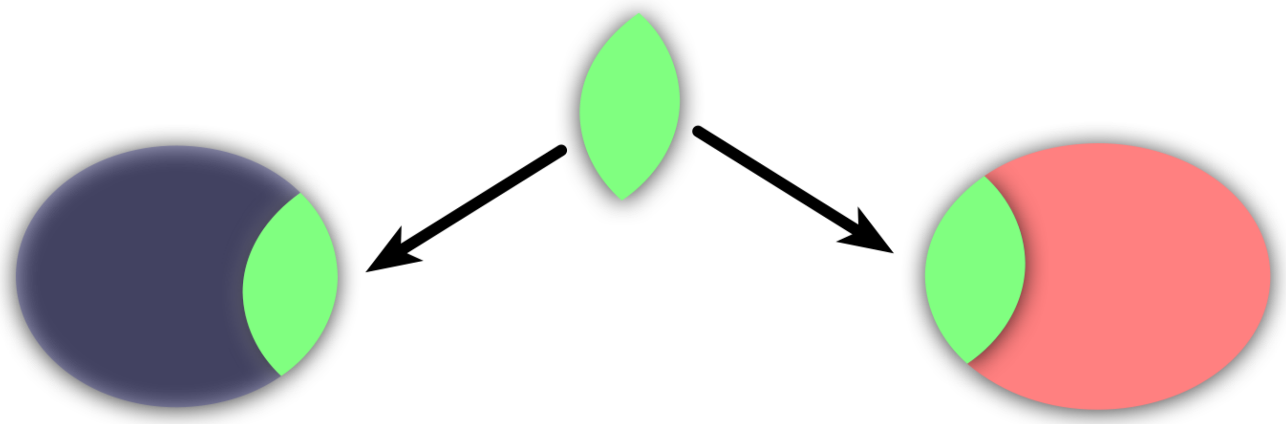
$\Rightarrow$ a **graph** G is given by the data $G(V)$ (set of ***vertices***), $G(E)$ (set of ***edges***) and two morphisms $G(s), G(t) : G(E) \to G(V)$ (***source/target*** *maps*)

$\Rightarrow$ a **graph homomorphism** $\phi = (\phi_V, \phi_E) : G_1 \to G_2$ is a natural transformation, i.e.

$$\begin{tikzcd} G_1(E) \arrow[r, "G_1(s)"] \arrow[d, "\phi_E"'] & G_1(V) \arrow[d, "\phi_V"] \\ G_2(E) \arrow[r, "G_2(s)"'] & G_2(V) \end{tikzcd} \qquad \begin{tikzcd} G_1(E) \arrow[r, "G_1(t)"] \arrow[d, "\phi_E"'] & G_1(V) \arrow[d, "\phi_V"] \\ G_2(E) \arrow[r, "G_2(t)"'] & G_2(V) \end{tikzcd}$$

commute

Pushouts implement **"gluing" along partial overlaps**



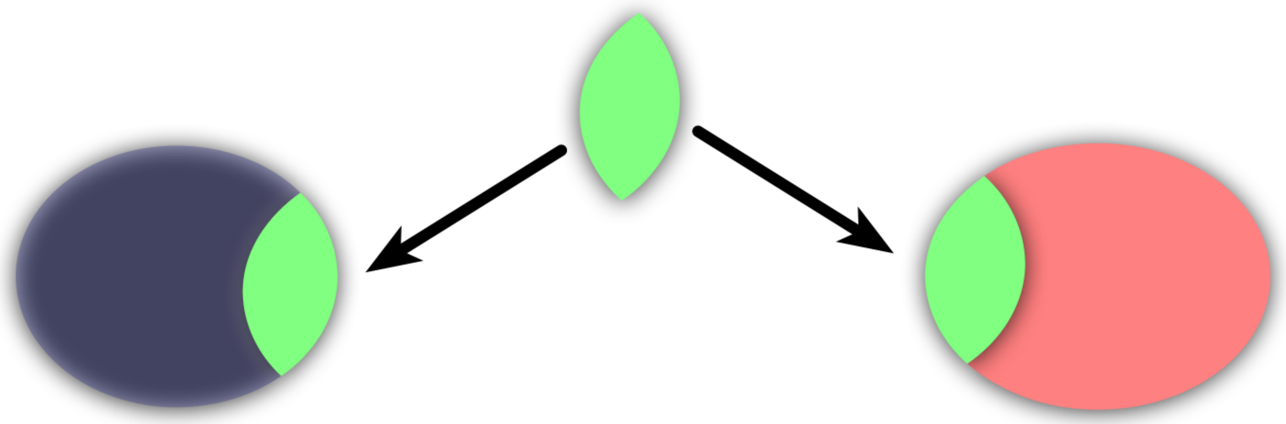
$$\begin{tikzcd} & A \arrow[dl, hook] \arrow[dr, hook] & \\ B & & C \end{tikzcd}$$

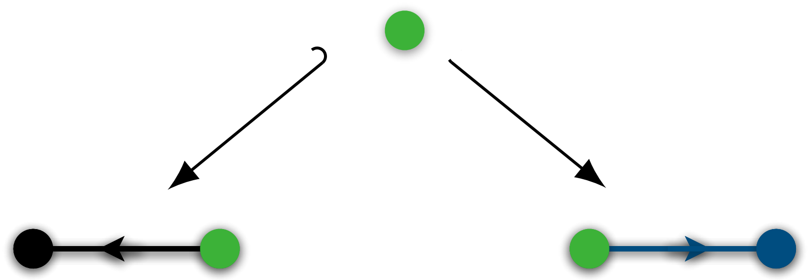
Interpretation:

A — **intersection** of B and C in D

D — **union** of B and C along A

Pushouts implement **"gluing" along partial overlaps**



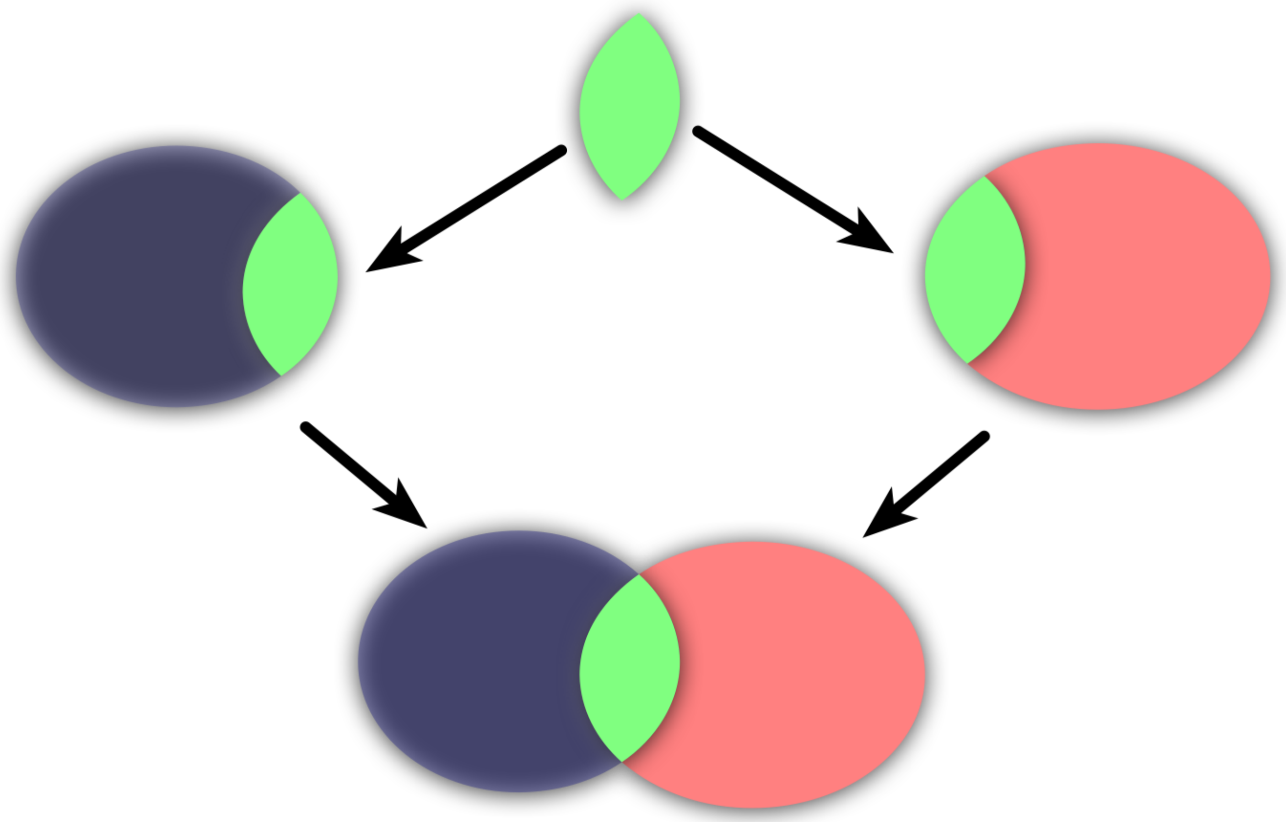


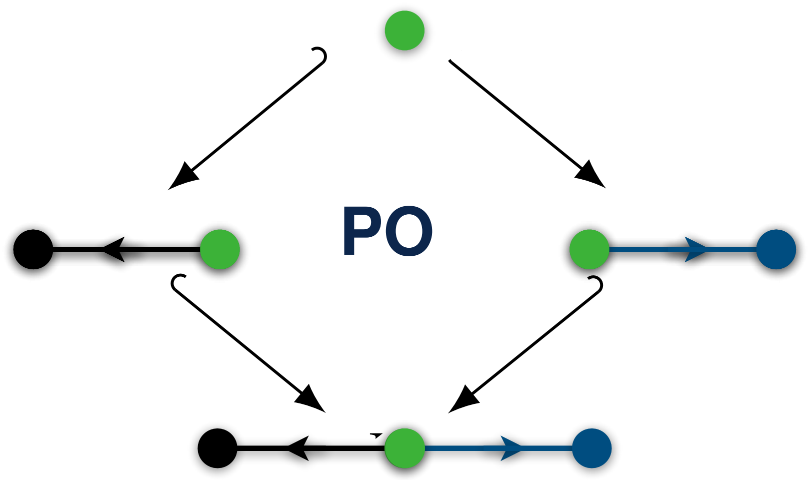
Interpretation:

A — **intersection** of B and C in D

D — **union** of B and C along A

Pushouts implement **“gluing” along partial overlaps**



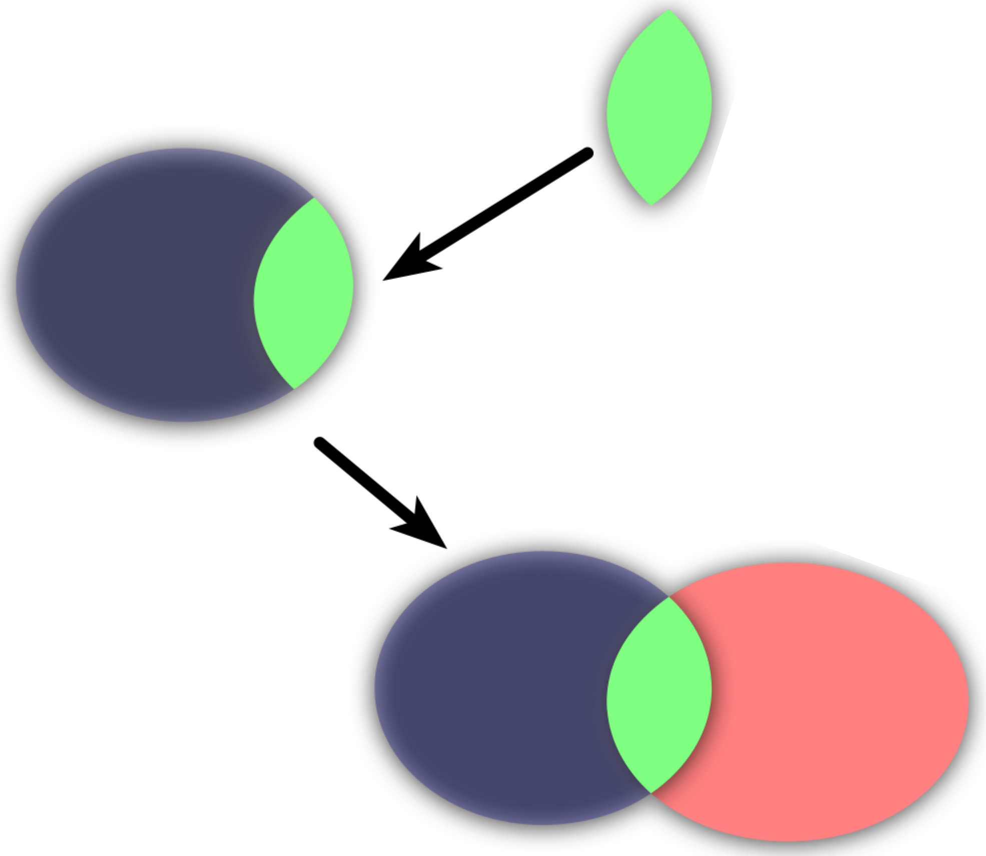


Interpretation:

A — **intersection** of B and C in D

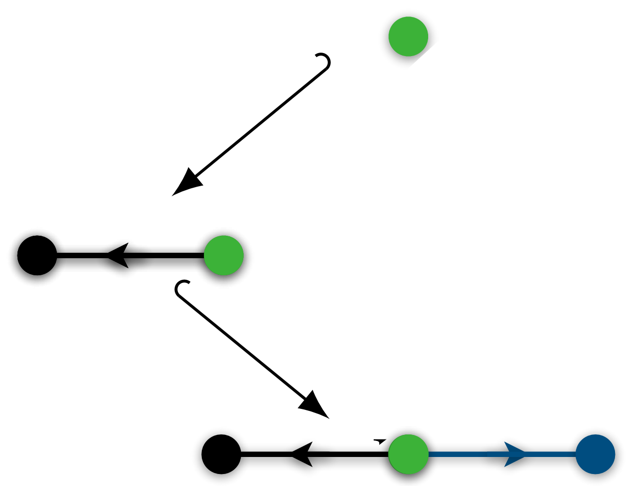
D — **union** of B and C along A

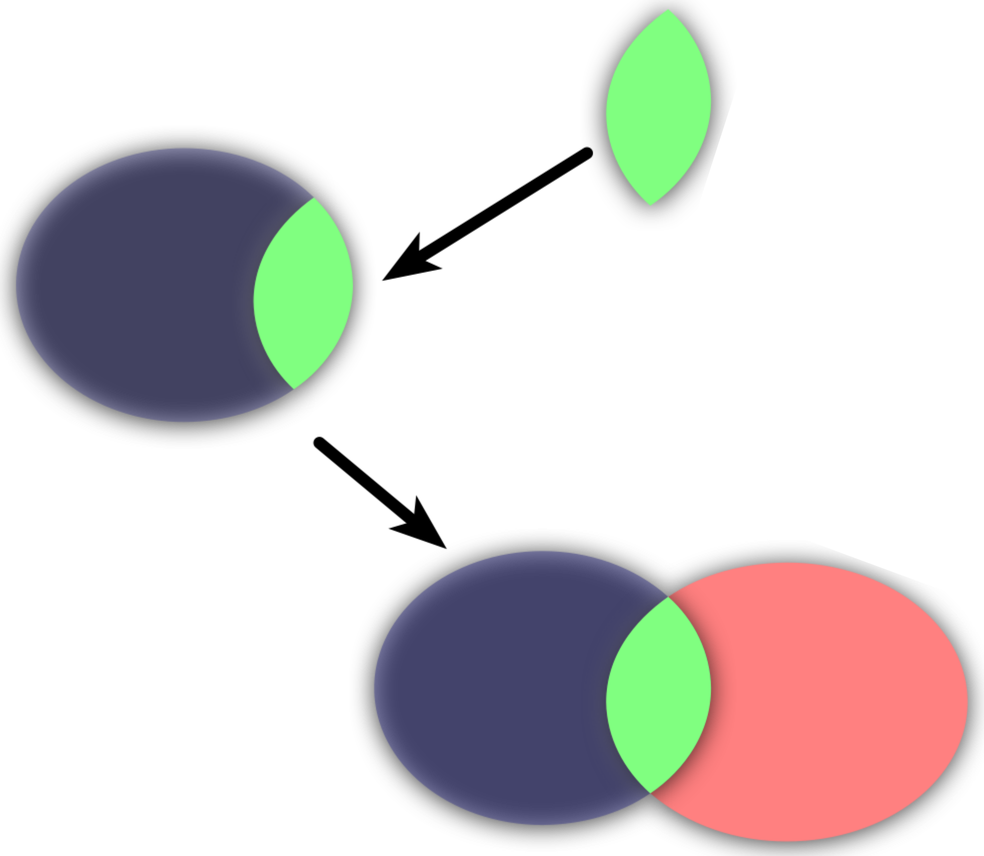
Pushout complements implement **partial deletions**



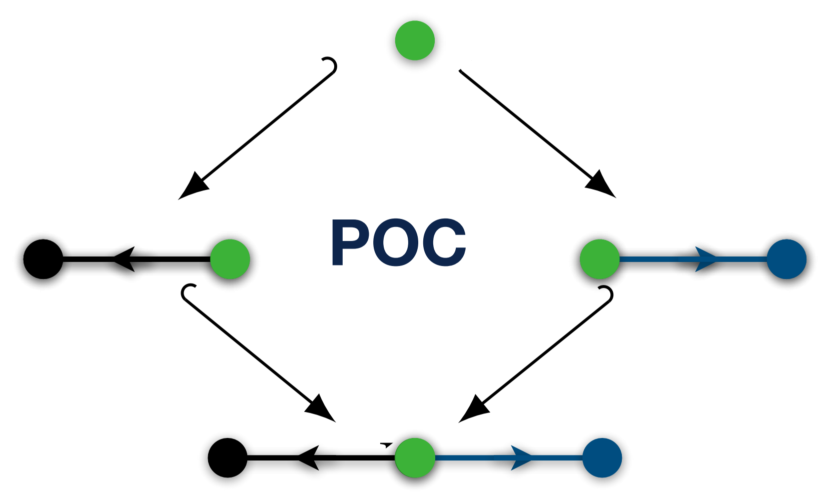
$$\begin{tikzcd} & A \arrow[dl, hook] \\ B \arrow[dr, hook] & \\ & D \end{tikzcd}$$

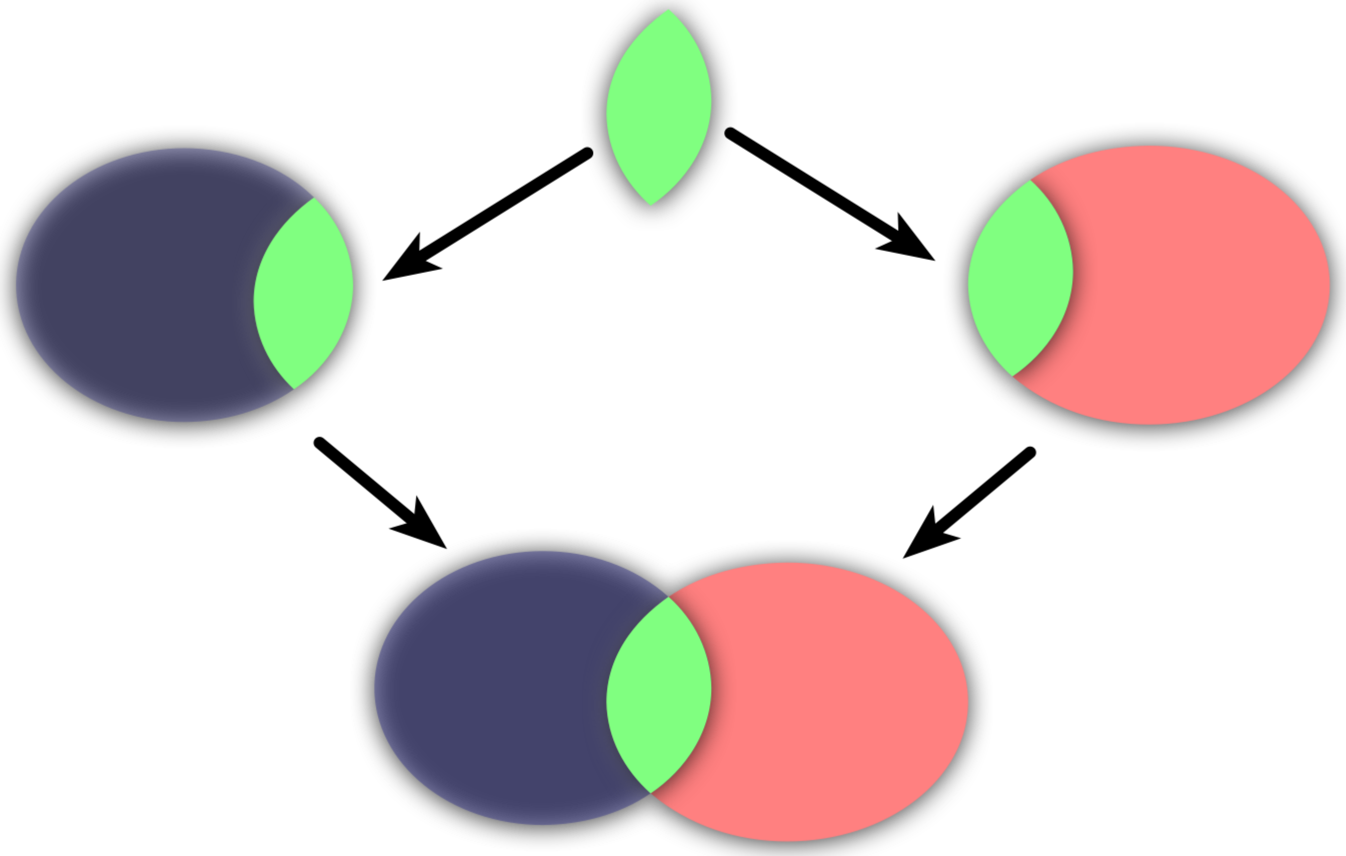
Pushout complements implement **partial deletions**





Pushout complements implement **partial deletions**





Sources of constructions of adhesive categories

Sources of constructions of adhesive categories

- Every **topos** (elementary or Grothendiek) is an adhesive category.

S. Lack & P. Sobociński (2006). *Toposes are adhesive*. In Proceedings of the Third international conference on Graph Transformations (pp. 184-198). Springer-Verlag.

Sources of constructions of adhesive categories

- Every **topos** (elementary or Grothendiek) is an adhesive category.

S. Lack & P. Sobociński (2006). *Toposes are adhesive*. In Proceedings of the Third international conference on Graph Transformations (pp. 184-198). Springer-Verlag.

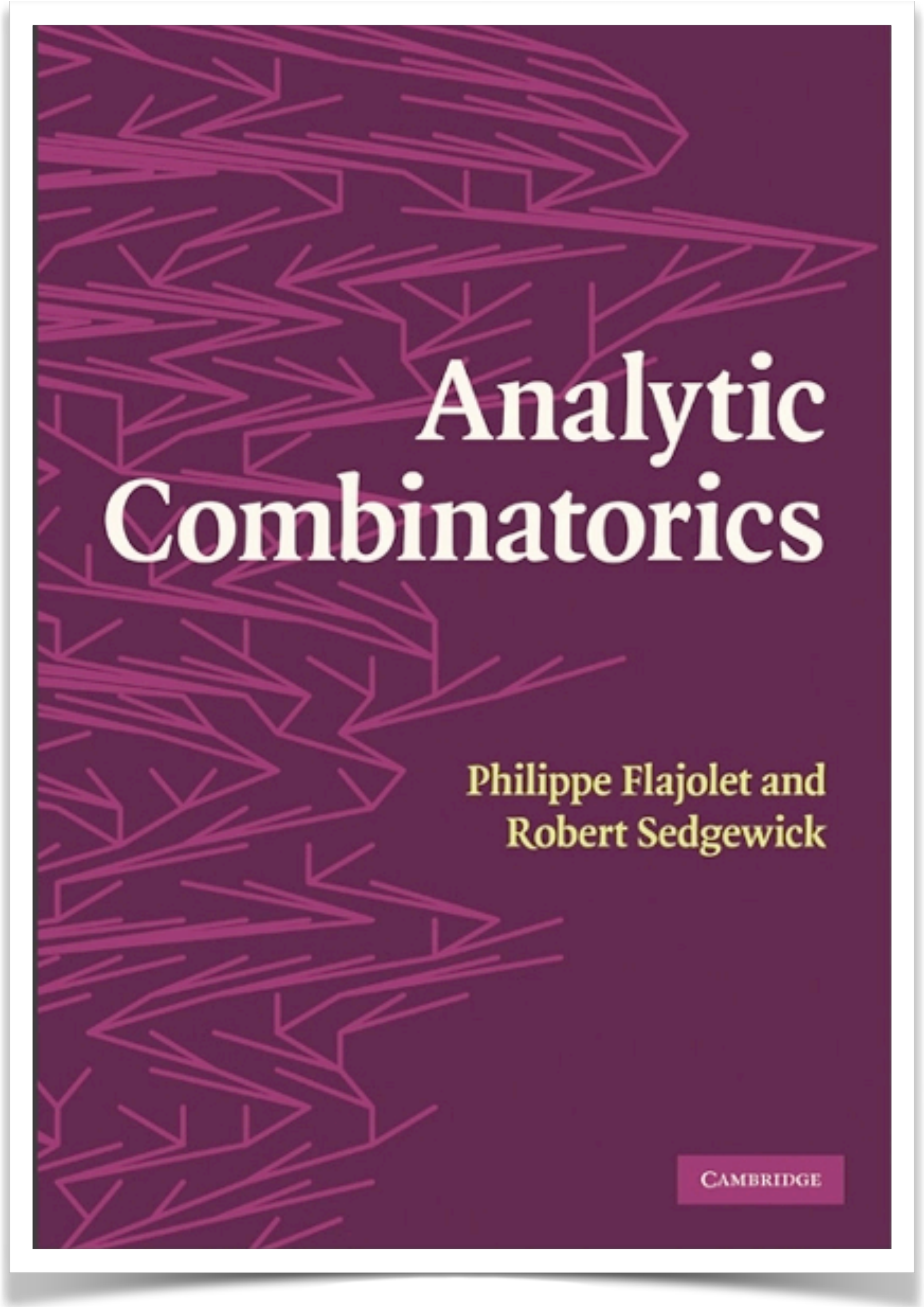
- **Categorical constructions** that yield new adhesive categories:
  - cartesian product
  - slice and coslice
  - comma categories (and other functor category constructions)
  - ...

Sources of constructions of adhesive categories

- Every **topos** (elementary or Grothendiek) is an adhesive category.

S. Lack & P. Sobociński (2006). *Toposes are adhesive*. In Proceedings of the Third international conference on Graph Transformations (pp. 184-198). Springer-Verlag.

- **Categorical constructions** that yield new adhesive categories:
  - cartesian product
  - slice and coslice
  - comma categories (and other functor category constructions)
  - . . .



On Stochastic Rewriting and Combinatorics via Rule-Algebraic Methods*

Nicolas Behr

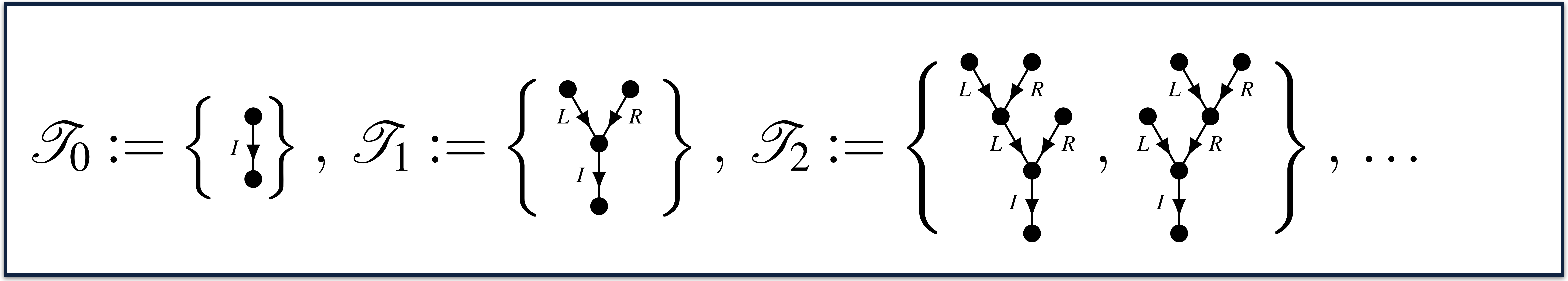
Université de Paris, CNRS, IRIF
F-75006, Paris, France
nicolas.behr@irif.fr

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.

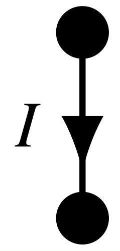
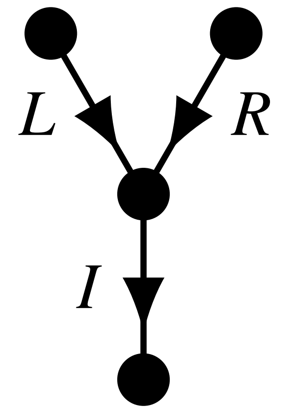
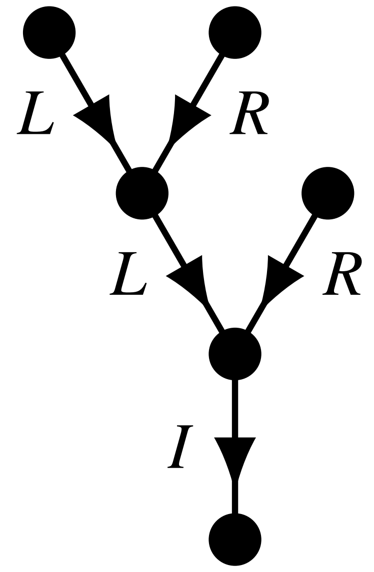
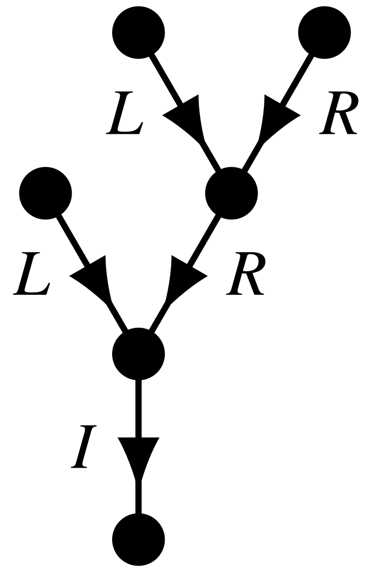
TERMGRAPH 2020

Running example: **planar rooted binary trees (PRBTs)**

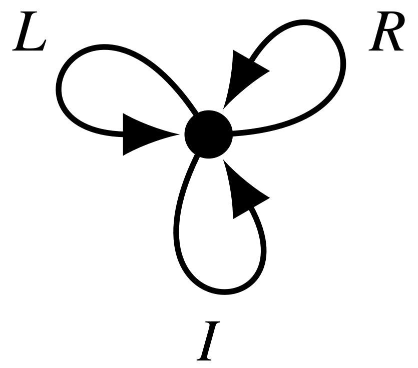
Running example: **planar rooted binary trees (PRBTs)**



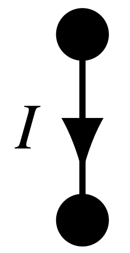
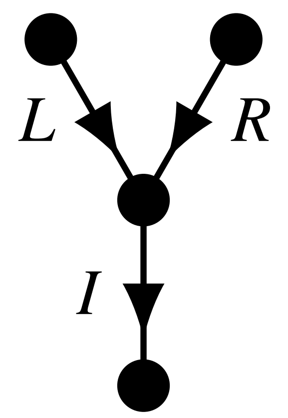
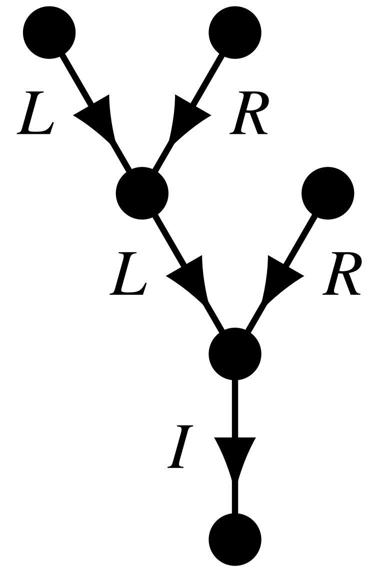
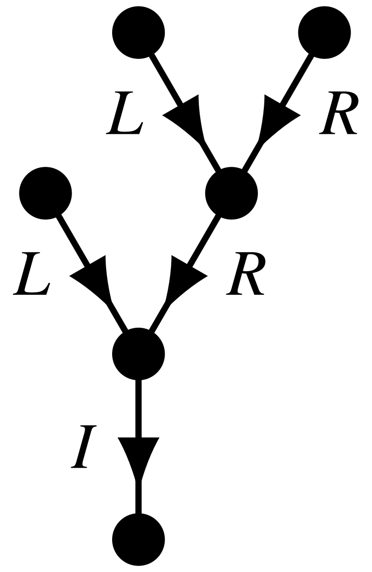
Running example: **planar rooted binary trees (PRBTs)**

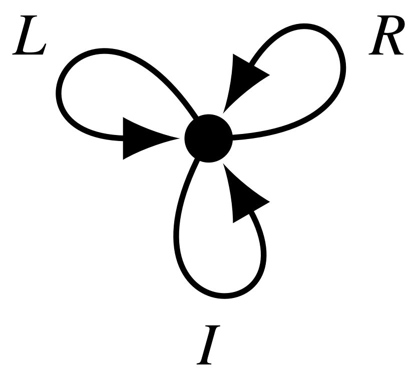
$\mathscr{T}_0 := \Big\{$  $\Big\}$, $\mathscr{T}_1 := \Big\{$  $\Big\}$, $\mathscr{T}_2 := \Big\{$  ,  $\Big\}$, $\ldots$

$$\mathbf{prePRBF} := \mathbf{FinGraph}/T_{PRBF}\,, \quad T_{PRBF} :=$$



Running example: **planar rooted binary trees (PRBTs)**

$\mathscr{T}_0 := \Big\{$  $\Big\}$, $\mathscr{T}_1 := \Big\{$  $\Big\}$, $\mathscr{T}_2 := \Big\{$  ,  $\Big\}$, $\ldots$

$\mathbf{prePRBF} := \mathbf{FinGraph}/T_{PRBF}\,, \quad T_{PRBF} :=$ 

But: how to encode the **structural properties** of PRBTs?

Constraints formalism for adhesive categories

Definition. For an adhesive, extensive and finitary category $\mathbf{C}$, **constraints** are recursively defined as follows: let $p : P \hookrightarrow P'$ be a monomorphism.

Constraints formalism for adhesive categories

Definition. For an adhesive, extensive and finitary category $\mathbf{C}$, **constraints** are recursively defined as follows: let $\mathsf{p} : P \hookrightarrow P'$ be a monomorphism.

- $\mathsf{p} \vDash \mathtt{true}$ — in words: "p **satisfies the condition** $\mathtt{true}$"

Constraints formalism for adhesive categories

Definition. For an adhesive, extensive and finitary category $\mathbf{C}$, **constraints** are recursively defined as follows: let $\mathsf{p} : \mathsf{P} \hookrightarrow \mathsf{P}'$ be a monomorphism.

- $\mathsf{p} \vDash \mathsf{true}$ — in words: "p **satisfies the condition** true"
- for every mono $\mathsf{a} : \mathsf{P} \hookrightarrow \mathsf{Q}$ and every condition $\mathsf{c_Q}$ (over Q), $\mathsf{p} \vDash \exists(\mathsf{a}, \mathsf{c_Q})$ iff there exists a mono $\mathsf{q} : \mathsf{Q} \hookrightarrow \mathsf{P}'$ such that $\mathsf{p} = \mathsf{q} \circ \mathsf{a}$ and $\mathsf{q} \vDash \mathsf{c_Q}$

$$
\begin{tikzcd}
\exists(a, \mathsf{c}_Q) \arrow[r, phantom, "\blacktriangleright"] & P \arrow[rr, "a"] \arrow[dr, "p"'] & & Q \arrow[dl, dashed, "q", "\vDash"'] & \mathsf{c}_Q \arrow[l, phantom, "\blacktriangleleft"] \\
& & P' & &
\end{tikzcd}
$$

Constraints formalism for adhesive categories

Definition. For an adhesive, extensive and finitary category $\mathbf{C}$, **constraints** are recursively defined as follows: let $\mathsf{p} : \mathsf{P} \hookrightarrow \mathsf{P}'$ be a monomorphism.

- $\mathsf{p} \vDash \mathsf{true}$ — in words: "p **satisfies the condition** true"
- for every mono $\mathsf{a} : \mathsf{P} \hookrightarrow \mathsf{Q}$ and every condition $\mathsf{c_Q}$ (over Q), $\mathsf{p} \vDash \exists(\mathsf{a}, \mathsf{c_Q})$ iff there exists a mono $\mathsf{q} : \mathsf{Q} \hookrightarrow \mathsf{P}'$ such that $\mathsf{p} = \mathsf{q} \circ \mathsf{a}$ and $\mathsf{q} \vDash \mathsf{c_Q}$
- for $\mathsf{c}, \mathsf{c}', \mathsf{c}''$ constraints,
  - $\mathsf{p} \vDash \neg \mathsf{c} \;:\Leftrightarrow\; \neg(\mathsf{p} \vDash \mathsf{c})$
  - $\mathsf{p} \vDash (\mathsf{c}' \wedge \mathsf{c}'') \;:\Leftrightarrow\; (\mathsf{p} \vDash \mathsf{c}') \wedge (\mathsf{p} \vDash \mathsf{c}'')$

$$\begin{tikzcd}
\exists(a, \mathsf{c}_Q) \arrow[r, phantom, "\blacktriangleright"] & P \arrow[rr, "a"] \arrow[dr, "p"'] & & Q \arrow[dl, dashed, "q"] & \mathsf{c}_Q \arrow[l, phantom, "\blacktriangleleft"] \\
& & P' & &
\end{tikzcd}$$

Constraints formalism for adhesive categories

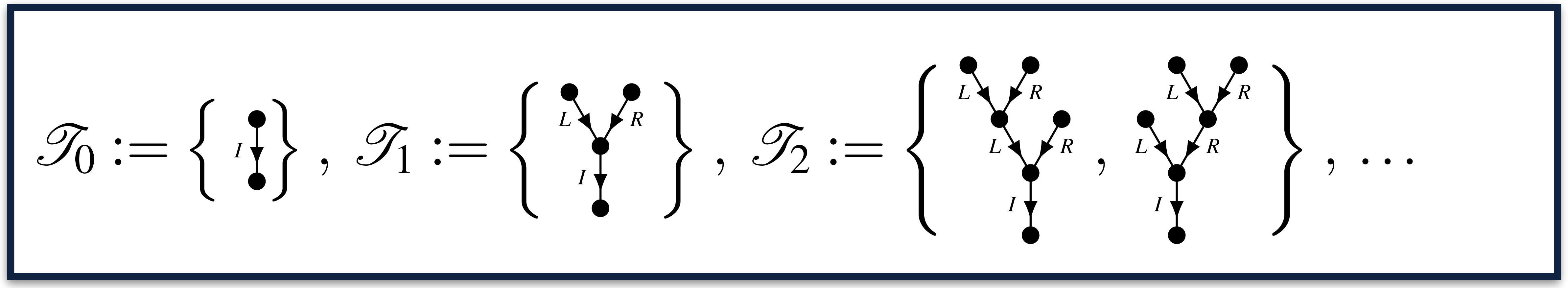
Definition. For an adhesive, extensive and finitary category $\mathbf{C}$, **constraints** are recursively defined as follows: let $\mathsf{p} : \mathsf{P} \hookrightarrow \mathsf{P}'$ be a monomorphism.

- $\mathsf{p} \vDash \mathtt{true}$ — in words: "p **satisfies the condition** $\mathtt{true}$"
- for every mono $\mathsf{a} : \mathsf{P} \hookrightarrow \mathsf{Q}$ and every condition c_Q (over Q), $\mathsf{p} \vDash \exists(\mathsf{a}, \mathsf{c}_\mathsf{Q})$ iff there exists a mono $\mathsf{q} : \mathsf{Q} \hookrightarrow \mathsf{P}'$ such that $\mathsf{p} = \mathsf{q} \circ \mathsf{a}$ and $\mathsf{q} \vDash \mathsf{c}_\mathsf{Q}$
- for $\mathsf{c}, \mathsf{c}', \mathsf{c}''$ constraints,
  - $\mathsf{p} \vDash \neg \mathsf{c} \;:\Leftrightarrow\; \neg(\mathsf{p} \vDash \mathsf{c})$
  - $\mathsf{p} \vDash (\mathsf{c}' \wedge \mathsf{c}'') \;:\Leftrightarrow\; (\mathsf{p} \vDash \mathsf{c}') \wedge (\mathsf{p} \vDash \mathsf{c}'')$

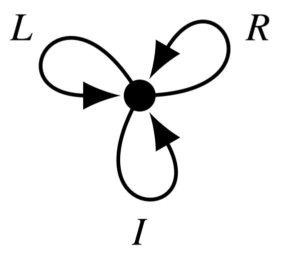
$$\begin{tikzcd}
\exists(a, \mathsf{c}_Q) \arrow[r, phantom, "\blacktriangleright"] & P \arrow[rr, "a"] \arrow[dr, "p"'] & & Q \arrow[dl, dashed, "q"] \arrow[dl, phantom, "\vDash", shift left=3] & \mathsf{c}_Q \arrow[l, phantom, "\blacktriangleleft"] \\
& & P' & &
\end{tikzcd}$$

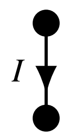
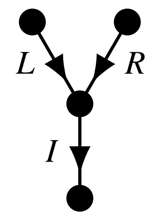
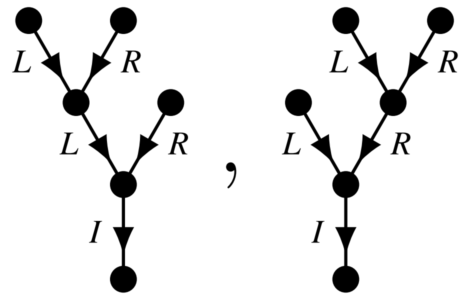
Definition. An **object** $\mathsf{X} \in \mathsf{obj}(\mathbf{C})$ is defined to satisfy a **constraint** $\mathsf{c}_\varnothing$ (i.e. a condition formulated over the **initial object** $\varnothing \in \mathsf{obj}(\mathbf{C})$) iff $(\iota_\mathsf{X} : \varnothing \hookrightarrow \mathsf{X}) \vDash \mathsf{c}_\varnothing$.

Running example: **planar rooted binary trees (PBRTs)**

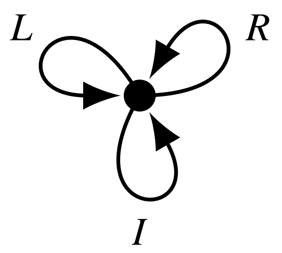


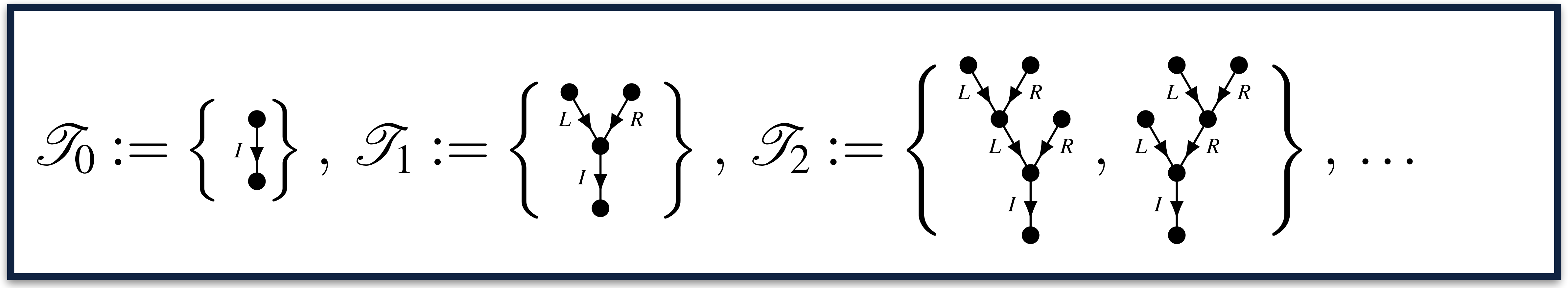
Running example: **planar rooted binary trees (PBRTs)**

$$\mathbf{prePRBF} := \mathbf{FinGraph}/T_{PRBF}\,, \quad T_{PRBF} :=$$ 

$$\mathscr{T}_0 := \Big\{$$  $$\Big\}\,, \quad \mathscr{T}_1 := \Big\{$$  $$\Big\}\,, \quad \mathscr{T}_2 := \Big\{$$  $$\Big\}\,, \ldots$$

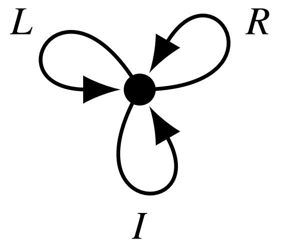
Running example: **planar rooted binary trees (PBRTs)**

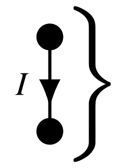
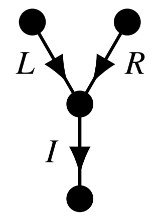
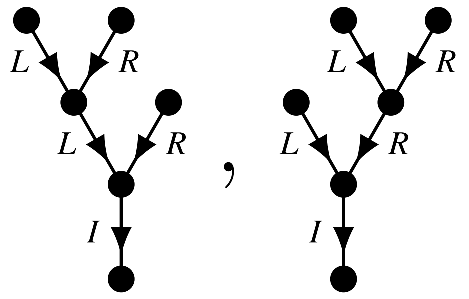
$$\mathbf{prePRBF} := \mathbf{FinGraph}/T_{PRBF}\,, \quad T_{PRBF} :=$$ 



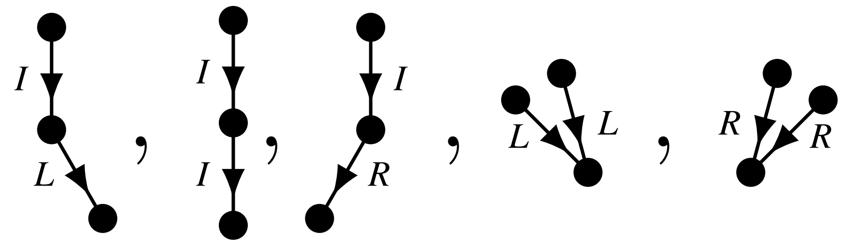
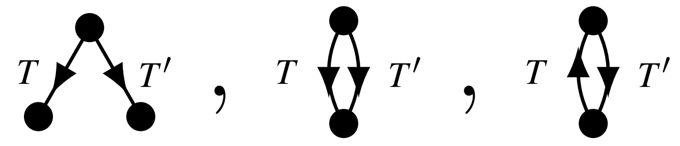
$$\mathsf{c}_{PRBF} := \mathsf{c}^{(-)}_{PRBF} \wedge \mathsf{c}^{(+)}_{PRBF}$$

Running example: **planar rooted binary trees (PBRTs)**

$$\mathbf{prePRBF} := \mathbf{FinGraph}/T_{PRBF}\,, \quad T_{PRBF} :=$$ 

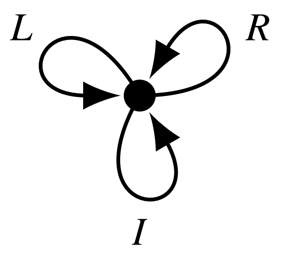
$\mathscr{T}_0 := \Big\{$  $\Big\}$, $\mathscr{T}_1 := \Big\{$  $\Big\}$, $\mathscr{T}_2 := \Big\{$  $\Big\}$, $\ldots$

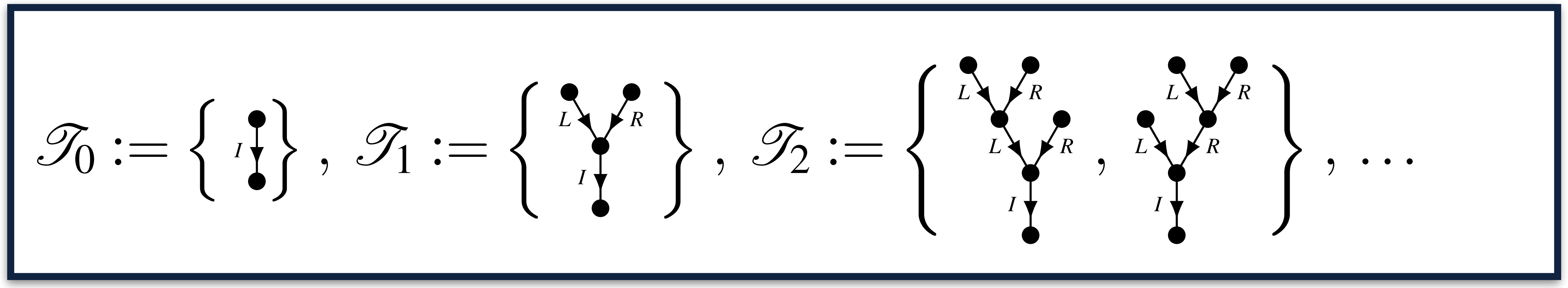
$$\mathsf{c}_{PRBF} := \mathsf{c}^{(-)}_{PRBF} \wedge \mathsf{c}^{(+)}_{PRBF}$$

$$\mathsf{c}^{(-)}_{PRBF} := \bigwedge_{N \in \mathscr{N}_{PRBF}} \nexists(\varnothing \hookrightarrow N)\,, \quad \mathscr{N}_{PRBF} := \Big\{$$  $$\Big\} \cup \bigcup_{T,T' \in \{I,L,R\}} \Big\{$$  $$\Big\}$$

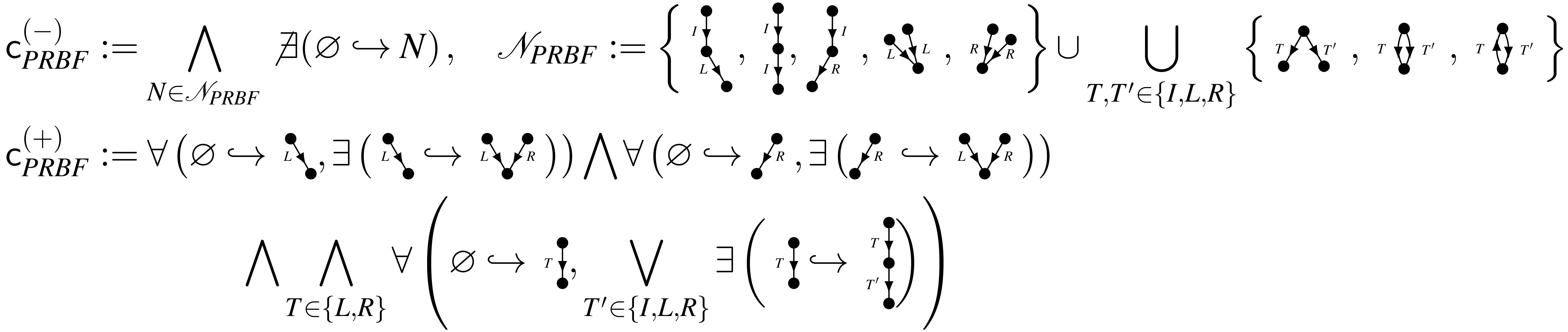
Running example: **planar rooted binary trees (PBRTs)**

$$\mathbf{prePRBF} := \mathbf{FinGraph}/T_{PRBF}\,,\quad T_{PRBF} :=$$

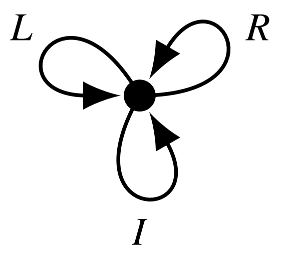


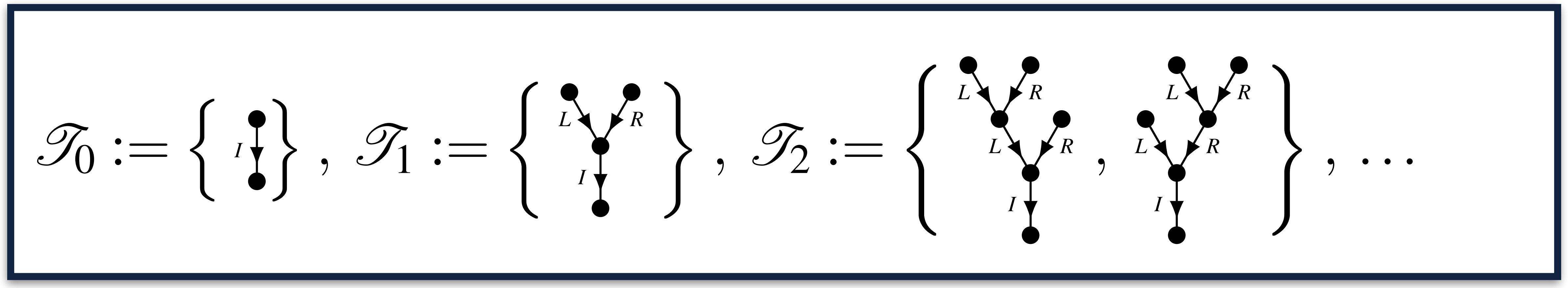


$$\mathsf{c}_{PRBF} := \mathsf{c}^{(-)}_{PRBF} \wedge \mathsf{c}^{(+)}_{PRBF}$$

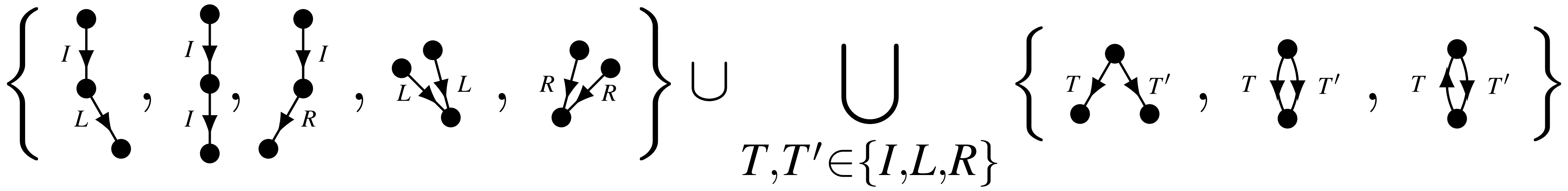


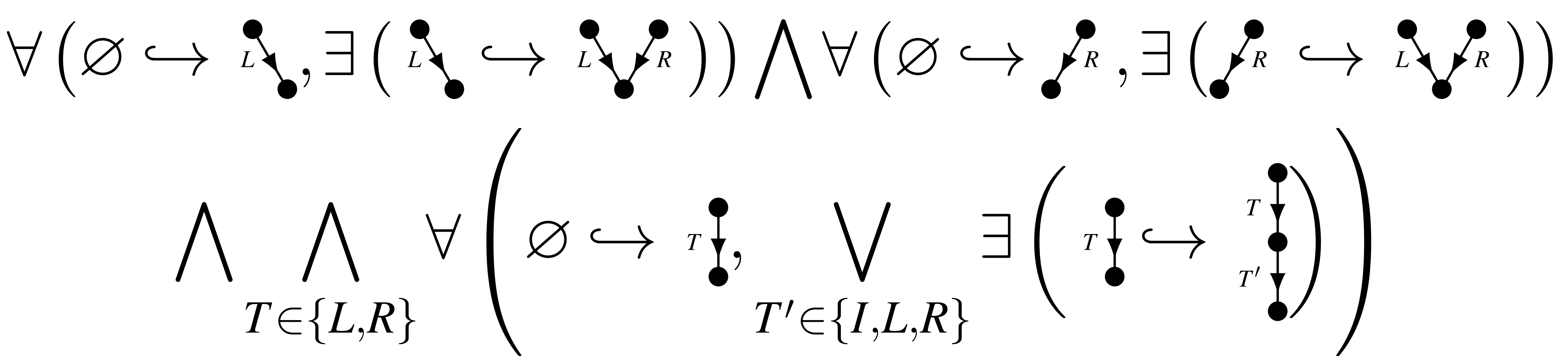
Running example: **planar rooted binary trees (PBRTs)**

$$\mathbf{prePRBF} := \mathbf{FinGraph}/T_{PRBF}\,, \quad T_{PRBF} :=$$ 



$$\mathsf{c}_{PRBF} := \mathsf{c}^{(-)}_{PRBF} \wedge \mathsf{c}^{(+)}_{PRBF}$$

$$\mathsf{c}^{(-)}_{PRBF} := \bigwedge_{N \in \mathscr{N}_{PRBF}} \nexists(\varnothing \hookrightarrow N)\,, \quad \mathscr{N}_{PRBF} :=$$ 

$$\mathsf{c}^{(+)}_{PRBF} :=$$ 

We then define the set $\mathbf{P}_{PRBF}$ of **PRBF patterns** and the set $\mathbf{S}_{PRBF}$ of **states** (with the latter coinciding of course with the set of PRBFs):

$$\mathbf{P}_{PRBF} := \{X \in \mathrm{obj}(\mathbf{prePRBF})_{\cong} \mid X \vDash \mathsf{c}^{(-)}_{PRBF}\}\,, \quad \mathbf{S}_{PRBF} := \{X \in \mathbf{P}_{PRBF} \mid X \vDash \mathsf{c}^{(+)}_{PRBF}\}$$

The central “workflow” of **categorical rewriting theory**

Fix an **adhesive finitary extensive category C**

The central “workflow” of **categorical rewriting theory**

Fix an **adhesive finitary extensive category C**

- Isomorphism classes of **objects** of **C** will model the **configurations**.

The central "workflow" of **categorical rewriting theory**

Fix an **adhesive finitary extensive category C**

- Isomorphism classes of **objects** of **C** will model the **configurations**.
- Isomorphism classes of **spans of monomorphisms** will model the **transitions**, also referred to as (linear) **rewriting rules**:

$$r = (O \xleftarrow{r} I) \equiv (O \overset{o}{\hookleftarrow} K \overset{i}{\hookrightarrow} I)$$

Note: here, "isomorphism" refers to entry-wise isomorphisms of the spans that encode rules, i.e. not the standard notion of isomorphism of spans

The foundation: **"compositional" rewriting theory** for **linear rules with conditions** (**DPO & SqPO**)

Compositionality of Rewriting Rules with Conditions

Nicolas Behr and Jean Krivine

IRIF, Université Paris-Diderot (Paris 07), F-75013 Paris, France

We extend the notion of compositional associative rewriting as recently studied in the rule algebra framework literature to the setting of rewriting rules with conditions. Our methodology is category-theoretical in nature, where the definition of rule composition operations is encoding the non-deterministic sequential concurrent application of rules in Double-Pushout (DPO) and Sesqui-Pushout (SqPO) rewriting with application conditions based upon $\mathcal{M}$-adhesive categories. We uncover an intricate interplay between the category-theoretical concepts of conditions on rules and morphisms, the compositionality and compatibility of certain shift and transport constructions for conditions, and thirdly the property of associativity of the composition of rules.

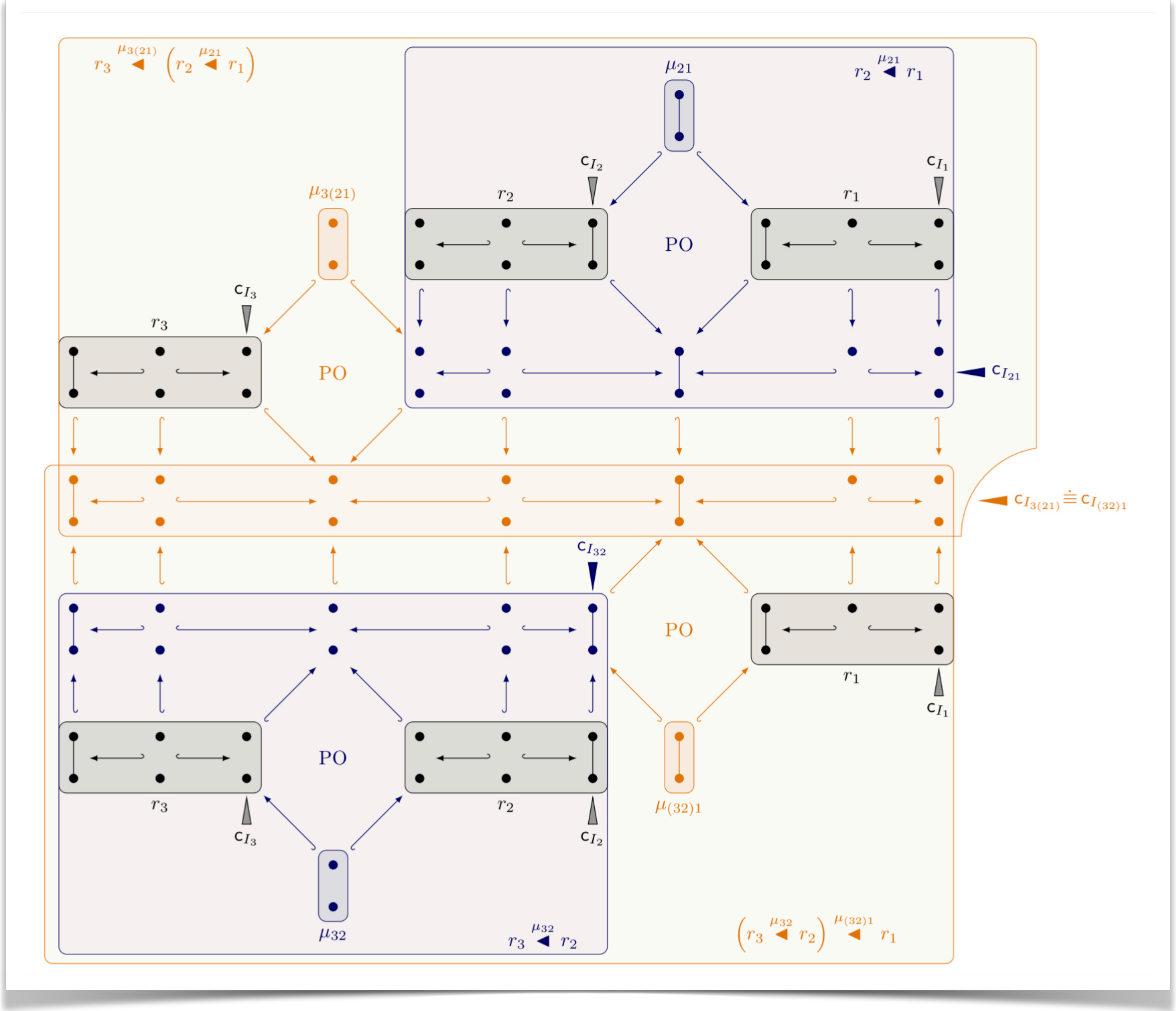
The foundation: **"compositional" rewriting theory for linear rules with conditions** (**DPO & SqPO**)

Compositionality of Rewriting Rules with Conditions

Nicolas Behr and Jean Krivine

IRIF, Université Paris-Diderot (Paris 07), F-75013 Paris, France

We extend the notion of compositional associative rewriting as recently studied in the rule algebra framework literature to the setting of rewriting rules with conditions. Our methodology is category-theoretical in nature, where the definition of rule composition operations is encoding the non-deterministic sequential concurrent application of rules in Double-Pushout (DPO) and Sesqui-Pushout (SqPO) rewriting with application conditions based upon $\mathcal{M}$-adhesive categories. We uncover an intricate interplay between the category-theoretical concepts of conditions on rules and morphisms, the compositionality and compatibility of certain shift and transport constructions for conditions, and thirdly the property of associativity of the composition of rules.



The suitable notion of **rules with conditions**

Definition 1. Let $\overline{\mathsf{Lin}}(\mathbf{C})$ denote the class of **(linear) rules with conditions**, defined as

$$\overline{\mathsf{Lin}}(\mathbf{C}) := \{(O \xleftarrow{o} K \xrightarrow{i} I; \mathsf{c}_I) \mid o, i \in \mathcal{M},\ \mathsf{c}_I \in \mathsf{cond}(\mathbf{C})\}\,. \tag{1}$$

We define two rules with conditions $\mathsf{R_j} = (\mathsf{r_j}, \mathsf{c_{I_j}})$ ($\mathsf{j} = 1, 2$) **equivalent**, denoted $\mathsf{R_2} \sim \mathsf{R_1}$, iff $\mathsf{c_{I_1}} \equiv \mathsf{c_{I_2}}$ and if there exist isomorphisms $\omega, \kappa, \iota \in \mathsf{iso}(\mathbf{C})$ such that the diagram on the right commutes. We denote by $\overline{\mathsf{Lin}}(\mathbf{C})_{\sim}$ the set of equivalence classes under $\sim$ of rules with conditions.

$$\begin{tikzcd} O_1 \arrow[d, "\omega", "\cong"'] & K_1 \arrow[l, hook'] \arrow[r, hook] \arrow[d, "\kappa", "\cong"'] & I_1 \arrow[d, "\iota", "\cong"'] \\ O_2 & K_2 \arrow[l, hook'] \arrow[r, hook] & I_2 \end{tikzcd} \tag{2}$$

The suitable notion of **rule applications**

Definition 2. Let $\mathsf{r} = (\mathsf{O} \hookleftarrow \mathsf{K} \hookrightarrow \mathsf{I}) \in \mathtt{Lin}(\mathbf{C})$ and $\mathsf{c_I} \in \mathtt{cond}(\mathbf{C})$ be concrete representatives of some equivalence class $\mathsf{R} \in \overline{\mathtt{Lin}}(\mathbf{C})_{\sim}$, and let $\mathsf{X}, \mathsf{Y} \in \mathtt{obj}(\mathbf{C})$ be objects. Then a **type $\mathbb{T}$ direct derivation** is defined as a commutative diagram such as below right, where all morphism are in $\mathcal{M}$ (and with the left representation a shorthand notation)

$$
\begin{tikzcd}
O \arrow[d, hook', "m^*"'] \arrow[dr, phantom, "\mathbb{T}"] & I \arrow[l, hook', "r"'] \arrow[d, hook, "m"] \\
Y & X \arrow[l, hook']
\end{tikzcd}
\quad := \quad
\begin{tikzcd}
O \arrow[d, hook', "m^*"'] \arrow[dr, phantom, "(B)"] & K \arrow[l, hook'] \arrow[r, hook] \arrow[d, hook, "k"] \arrow[dr, phantom, "(A)"] & I \arrow[d, hook, "m"] \\
Y & \overline{K} \arrow[l, hook'] \arrow[r, hook] & X
\end{tikzcd}
\;. \tag{3}
$$

with the following pieces of information required relative to the type:

1. $\mathbb{T} =$ **DPO**: given $(\mathsf{m} : \mathsf{I} \hookrightarrow \mathsf{X}) \in \mathcal{M}$, m is a **DPO-admissible match of** R **into** X, denoted $\mathsf{m} \in \mathsf{M}_{\mathsf{R}}^{DPO}(\mathsf{X})$, if $\mathsf{m} \vDash \mathsf{c_I}$ and (A) is constructable as a **pushout complement**, in which case (B) is constructed as a **pushout**.

The suitable notion of **rule applications**

Definition 2. Let $\mathsf{r} = (\mathsf{O} \hookleftarrow \mathsf{K} \hookrightarrow \mathsf{I}) \in \mathsf{Lin}(\mathbf{C})$ and $\mathsf{c_I} \in \mathsf{cond}(\mathbf{C})$ be concrete representatives of some equivalence class $\mathsf{R} \in \overline{\mathsf{Lin}}(\mathbf{C})_{\sim}$, and let $\mathsf{X}, \mathsf{Y} \in \mathsf{obj}(\mathbf{C})$ be objects. Then a **type $\mathbb{T}$ direct derivation** is defined as a commutative diagram such as below right, where all morphism are in $\mathcal{M}$ (and with the left representation a shorthand notation)

$$\begin{tikzcd} O \arrow[d, hookleftarrow, "m^*"'] & I \arrow[l, hook', "r"'] \arrow[d, hook, "m"] \arrow[dl, phantom, "\mathbb{T}"] \\ Y & X \arrow[l, hook'] \end{tikzcd} \quad := \quad \begin{tikzcd} O \arrow[d, hook, "m^*"'] & K \arrow[l, hook'] \arrow[r, hook] \arrow[d, hook, "k"] \arrow[dl, phantom, "(B)"] & I \arrow[d, hook, "m"] \arrow[dl, phantom, "(A)"] \\ Y & \overline{K} \arrow[l, hook'] \arrow[r, hook] & X \end{tikzcd} . \qquad (3)$$

with the following pieces of information required relative to the type:

2. $\mathbb{T} = $ **SqPO**: given $(\mathsf{m} : \mathsf{I} \hookrightarrow \mathsf{X}) \in \mathcal{M}$, m is a **SqPO-admissible match of** R **into** X, denoted $\mathsf{m} \in \mathsf{M}_{\mathsf{R}}^{SqPO}(\mathsf{X})$, if $\mathsf{m} \vDash \mathsf{c_I}$, in which case (A) is constructed as a **final pullback complement** and (B) as a **pushout**.

The suitable notion of **rule applications**

Definition 2. Let $\mathsf{r} = (\mathsf{O} \hookleftarrow \mathsf{K} \hookrightarrow \mathsf{I}) \in \mathsf{Lin}(\mathbf{C})$ and $\mathsf{c_I} \in \mathsf{cond}(\mathbf{C})$ be concrete representatives of some equivalence class $\mathsf{R} \in \overline{\mathsf{Lin}}(\mathbf{C})_{\sim}$, and let $\mathsf{X}, \mathsf{Y} \in \mathsf{obj}(\mathbf{C})$ be objects. Then a **type** $\mathbb{T}$ **direct derivation** is defined as a commutative diagram such as below right, where all morphism are in $\mathcal{M}$ (and with the left representation a shorthand notation)

$$
\begin{tikzcd}
O \arrow[d, hook, "m^*"'] & I \arrow[l, hook', "r"'] \arrow[d, hook, "m"] \arrow[dl, phantom, "\mathbb{T}"] \\
Y & X \arrow[l, hook']
\end{tikzcd}
\quad := \quad
\begin{tikzcd}
O \arrow[d, hook, "m^*"'] & K \arrow[l, hook'] \arrow[r, hook] \arrow[d, hook, "k"] \arrow[dl, phantom, "(B)"] & I \arrow[d, hook, "m"] \arrow[dl, phantom, "(A)"] \\
Y & \overline{K} \arrow[l, hook'] \arrow[r, hook] & X
\end{tikzcd}
\; . \tag{3}
$$

with the following pieces of information required relative to the type:

2. $\mathbb{T} = \mathbf{SqPO}$: given $(\mathsf{m} : \mathsf{I} \hookrightarrow \mathsf{X}) \in \mathcal{M}$, m is a **SqPO-admissible match of** R **into** X, denoted $\mathsf{m} \in \mathsf{M}_{\mathsf{R}}^{SqPO}(\mathsf{X})$, if $\mathsf{m} \vDash \mathsf{c_I}$, in which case (A) is constructed as a **final pullback complement** and (B) as a **pushout**.

For types $\mathbb{T} \in \{\mathsf{DPO}, \mathsf{SqPO}\}$, we will sometimes employ the notation $\mathsf{R_m}(\mathsf{X})$ for the object Y.

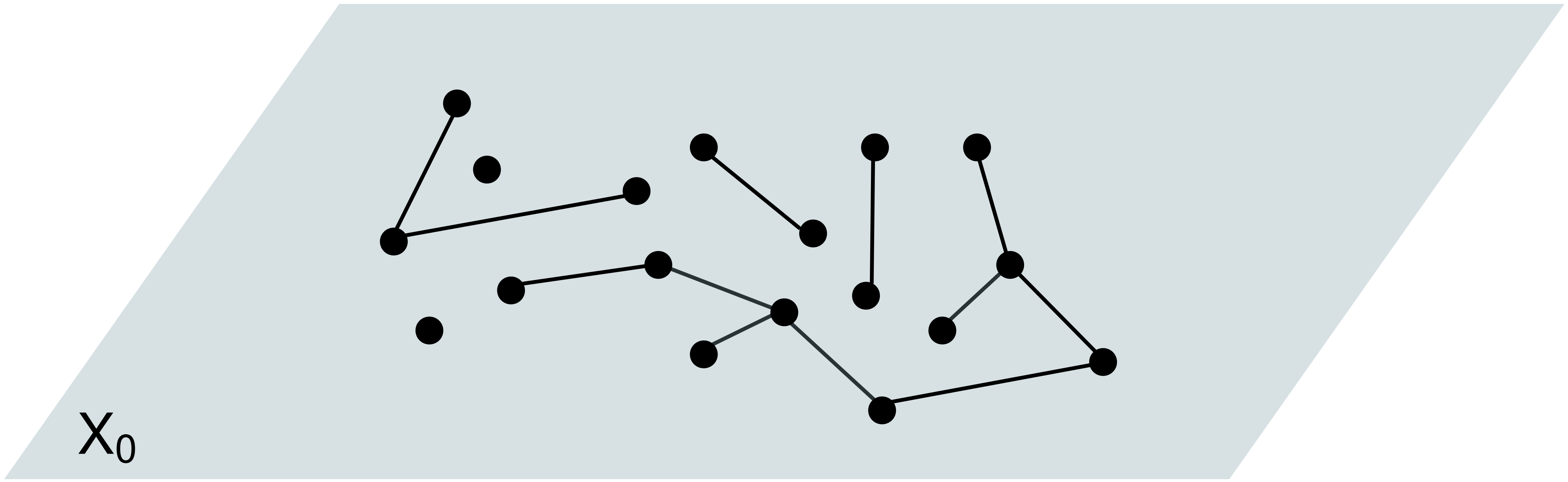
The suitable notion of **rule applications**

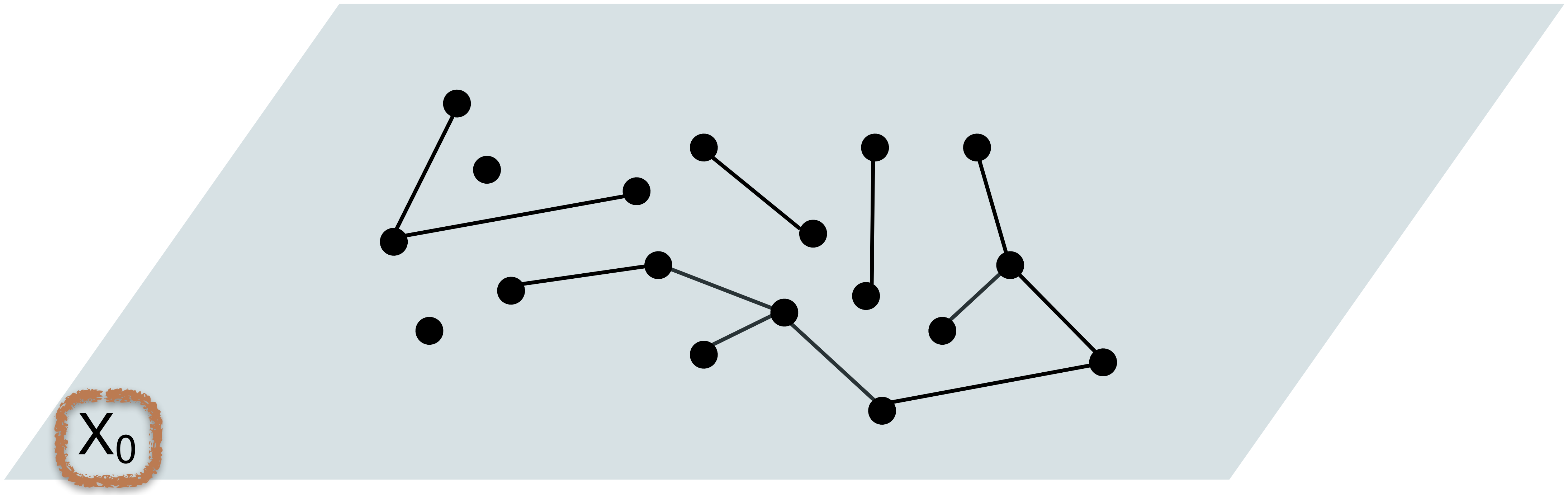
Definition 2. Let $\mathsf{r} = (O \hookleftarrow K \hookrightarrow I) \in \mathsf{Lin}(\mathbf{C})$ and $\mathsf{c}_I \in \mathsf{cond}(\mathbf{C})$ be concrete representatives of some equivalence class $R \in \overline{\mathsf{Lin}}(\mathbf{C})_{\sim}$, and let $X, Y \in \mathsf{obj}(\mathbf{C})$ be objects. Then a **type $\mathbb{T}$ direct derivation** is defined as a commutative diagram such as below right, where all morphism are in $\mathcal{M}$ (and with the left representation a shorthand notation)

$$\begin{tikzcd} O \arrow[d, hook, "m^*"'] & I \arrow[l, hook', "r"'] \arrow[d, hook, "m"] \arrow[dl, phantom, "\mathbb{T}"] \\ Y & X \arrow[l, hook'] \end{tikzcd} \quad := \quad \begin{tikzcd} O \arrow[d, hook, "m^*"'] & K \arrow[l, hook'] \arrow[r, hook] \arrow[d, hook, "k"] \arrow[dl, phantom, "(B)"] & I \arrow[d, hook, "m"] \arrow[dl, phantom, "(A)"] \\ Y & \overline{K} \arrow[l, hook'] \arrow[r, hook] & X \end{tikzcd} . \tag{3}$$

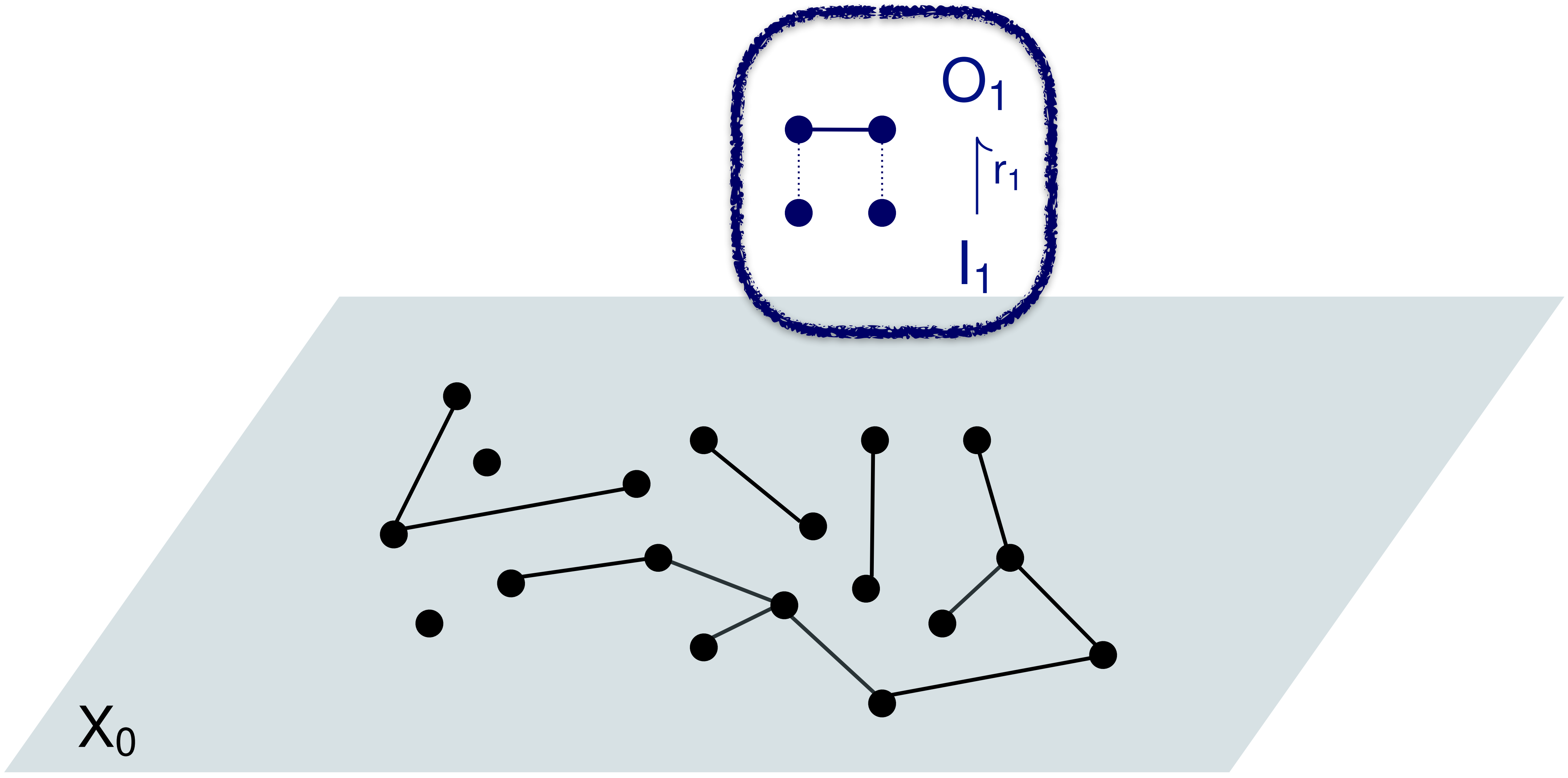
with the following pieces of information required relative to the type:

3. $\mathbb{T} = \mathbf{DPO}^{\dagger}$: given just the "plain rule" r and $(m^* : O \hookrightarrow Y) \in \mathcal{M}$, m^* is a **$\mathbf{DPO}^{\dagger}$-admissible match of r into X**, denoted $m \in \mathsf{M}_{\mathsf{r}}^{DPO^{\dagger}}(Y)$, if (B) is constructable as a **pushout complement**, in which case (B) is constructed as a **pushout**.


X_0

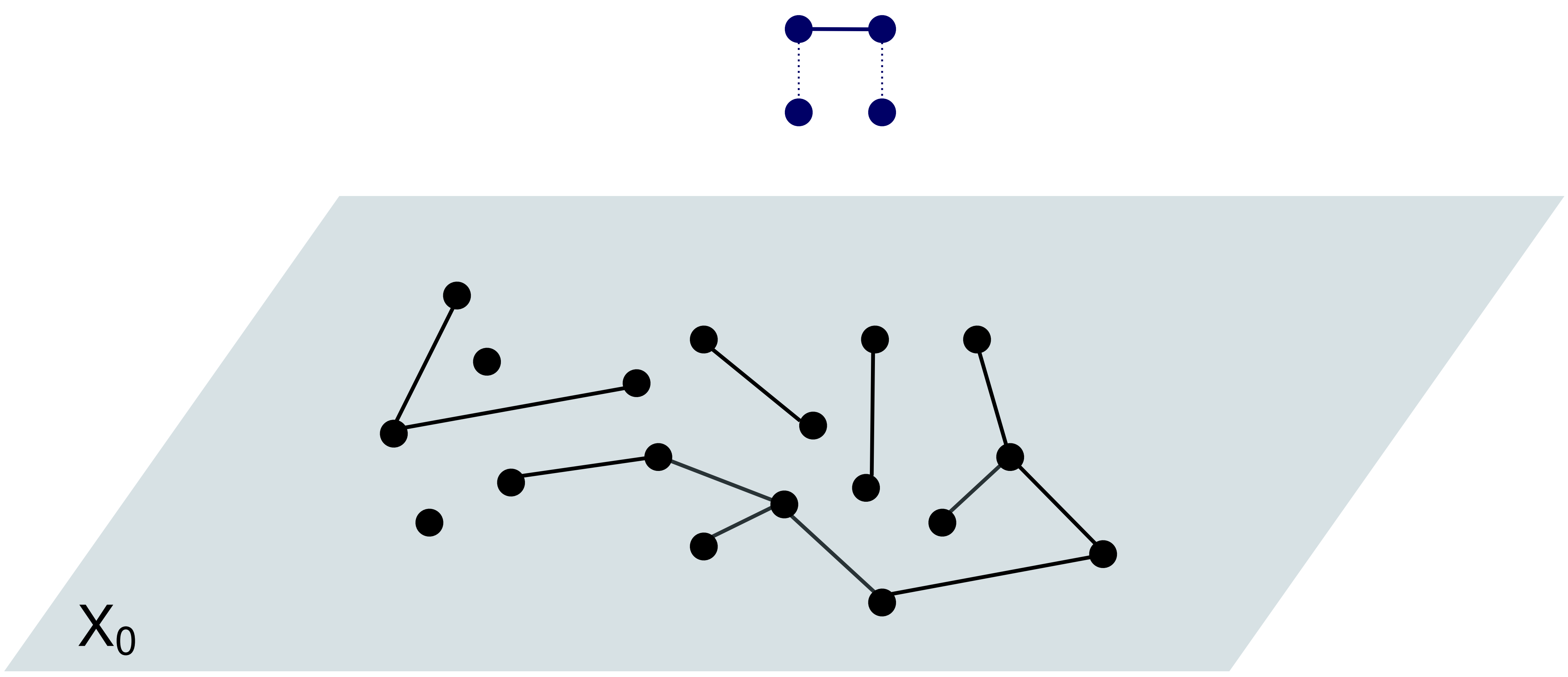


X_0



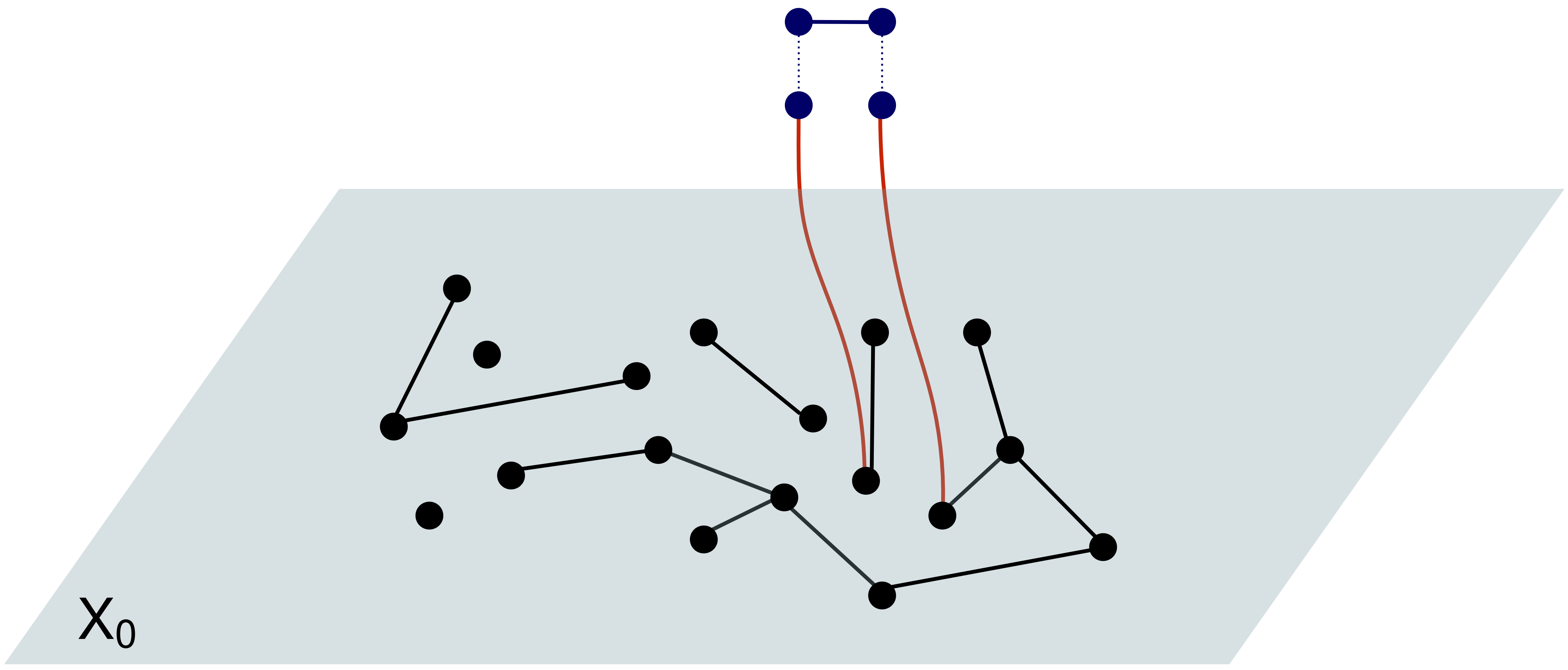
$O_1 \xleftarrow{r_1} I_1$

X_0

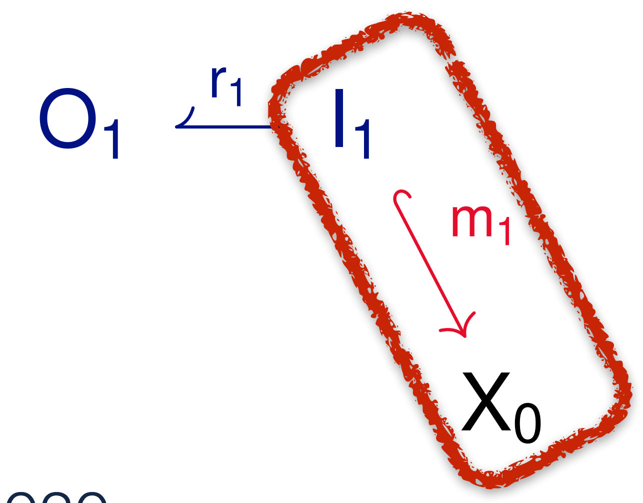


$$O_1 \xleftarrow{r_1} I_1$$

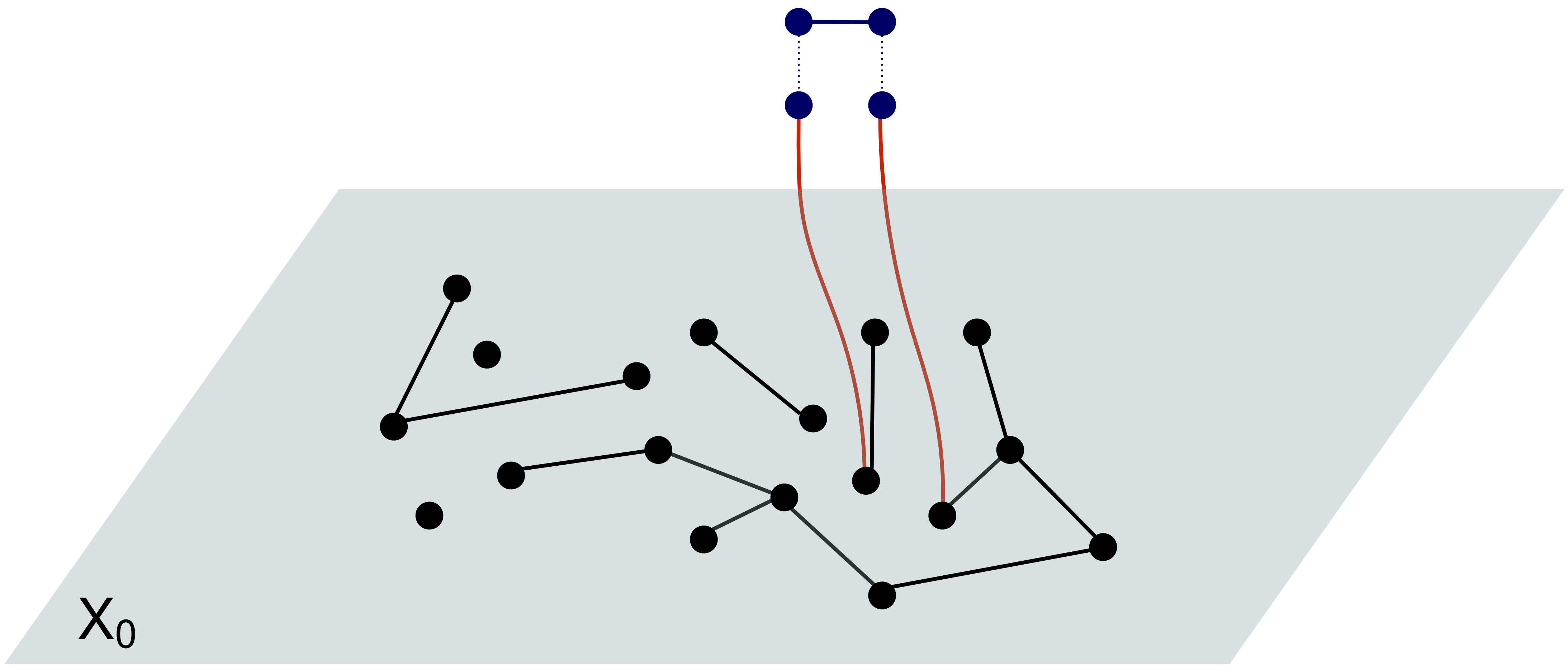
$$X_0$$



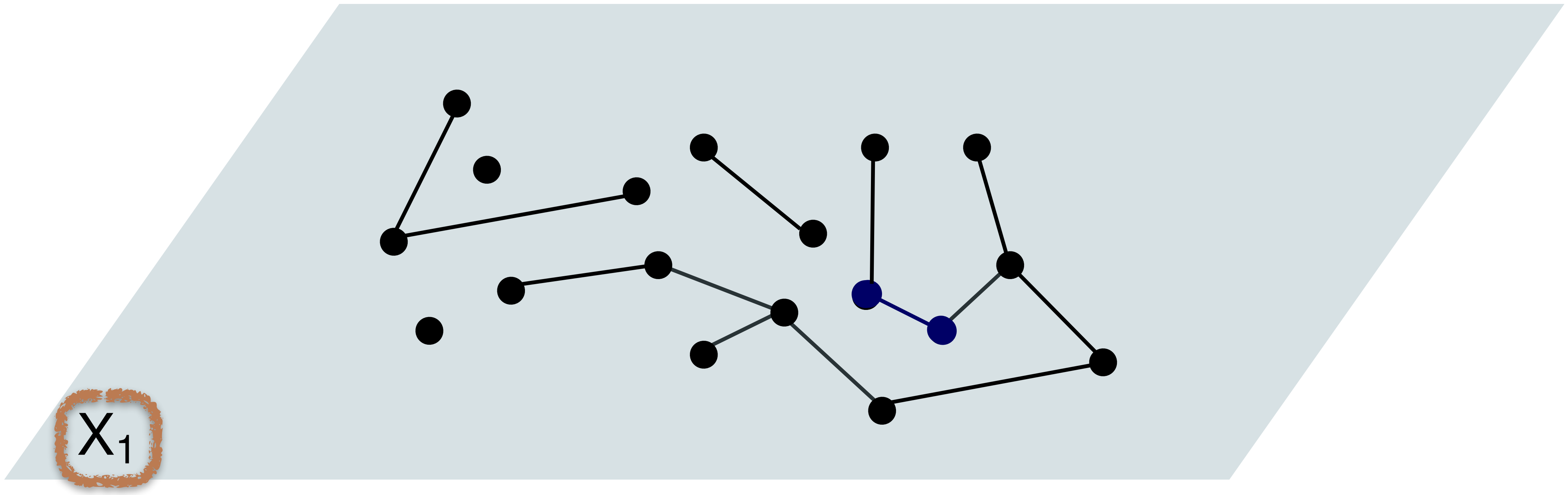
X0



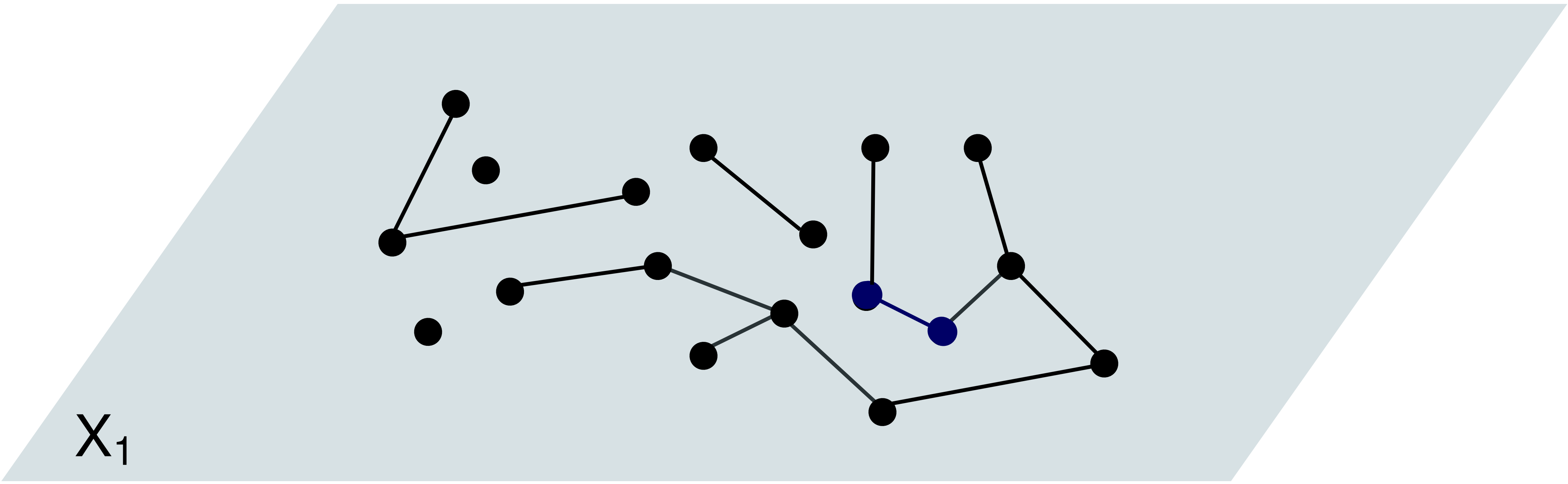
O1
r1
I1
m1
X0



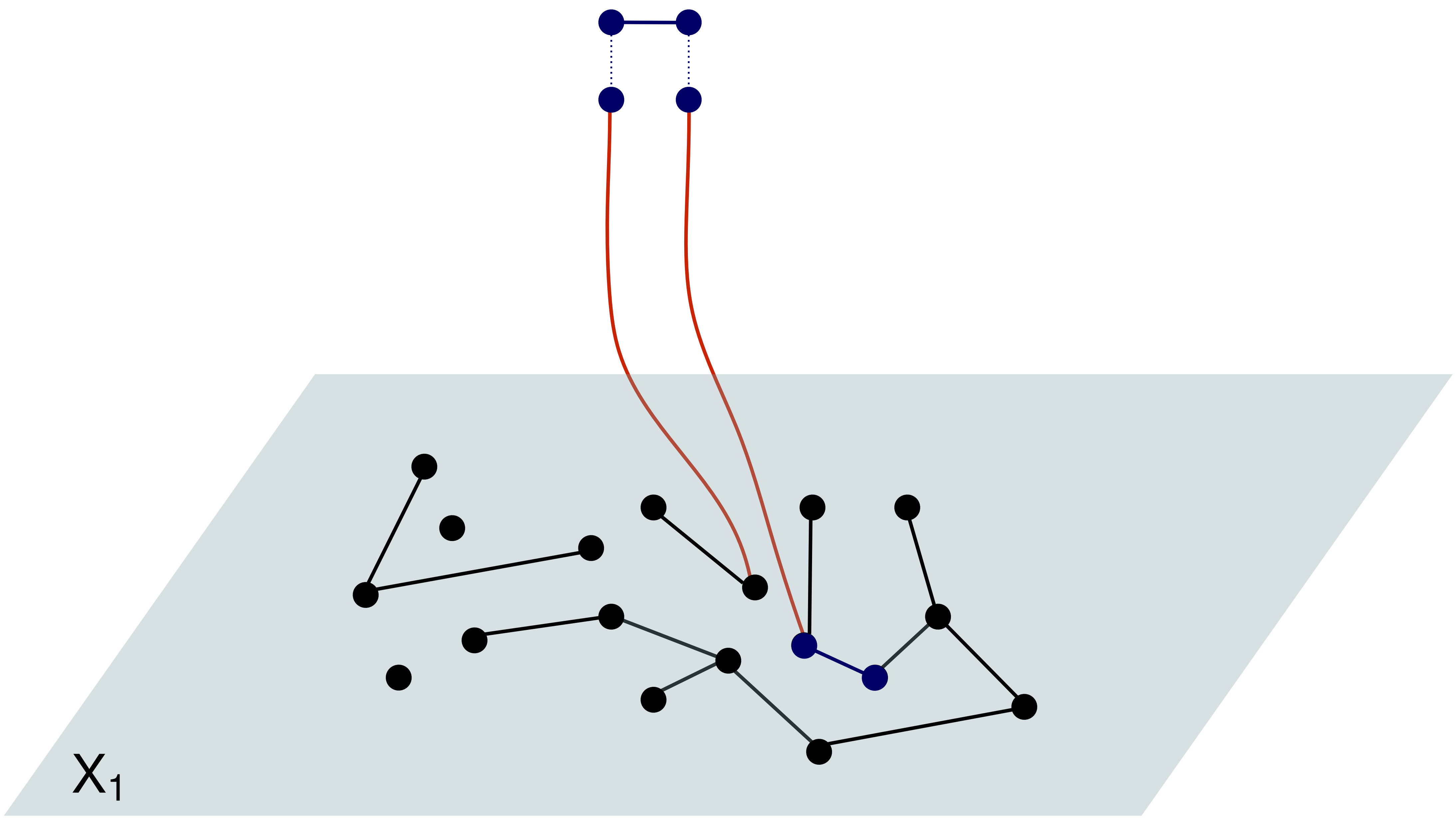
$$\begin{tikzcd} O_1 & I_1 \arrow[l, hook', "r_1"'] \arrow[d, hook, "m_1"] \\ & X_0 \end{tikzcd}$$

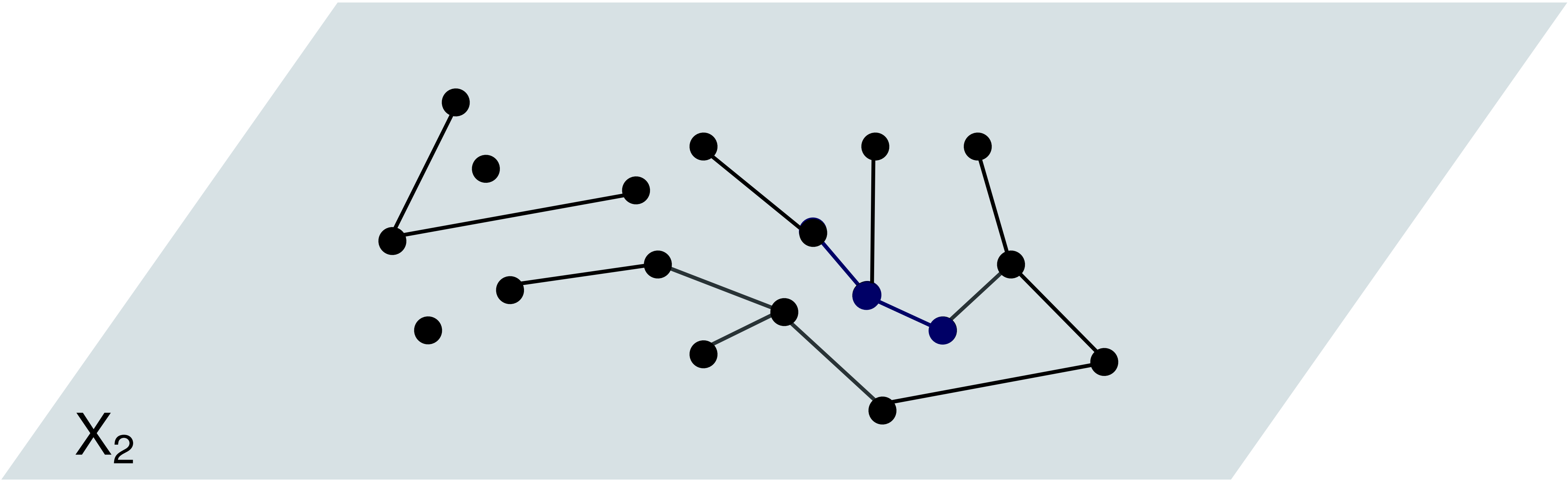


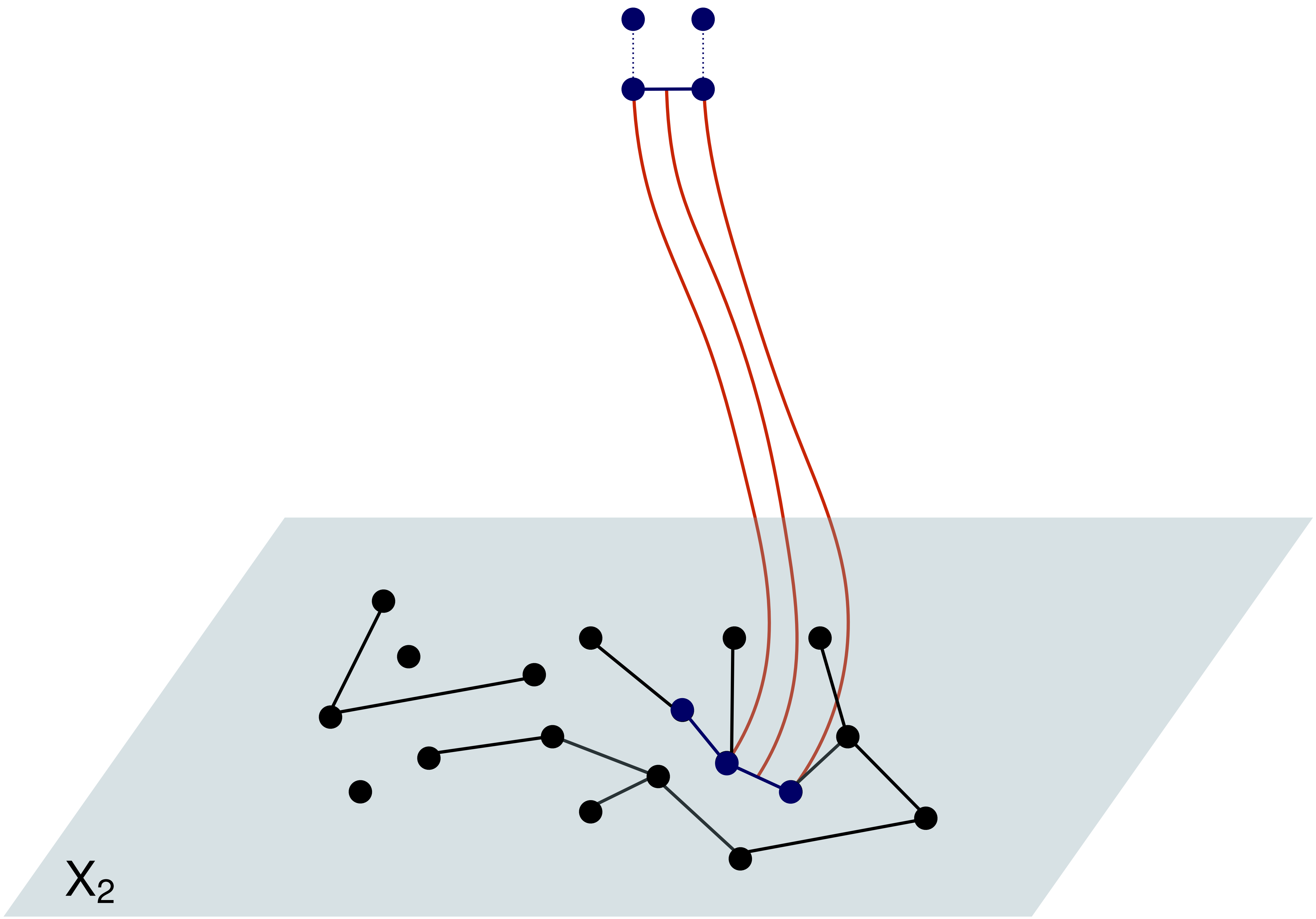
$$\begin{tikzcd}
O_1 & I_1 \arrow[l, "r_1"'] \arrow[d, "m_1"] \\
X_1 \arrow[from=u] & X_0 \arrow[l]
\end{tikzcd}$$

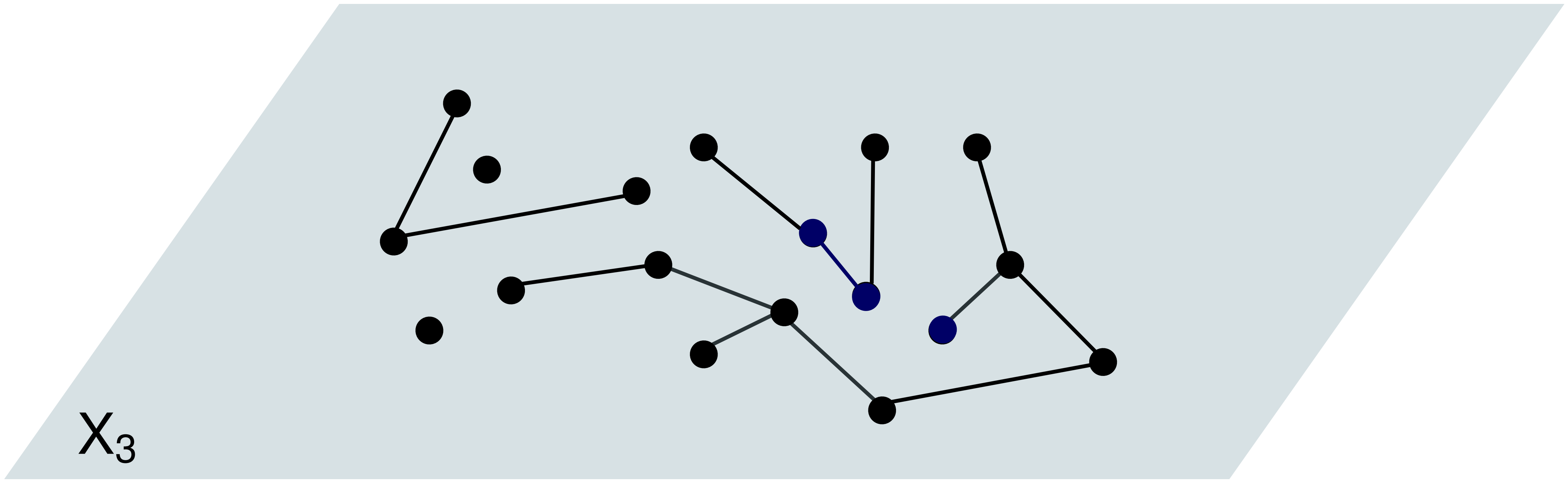


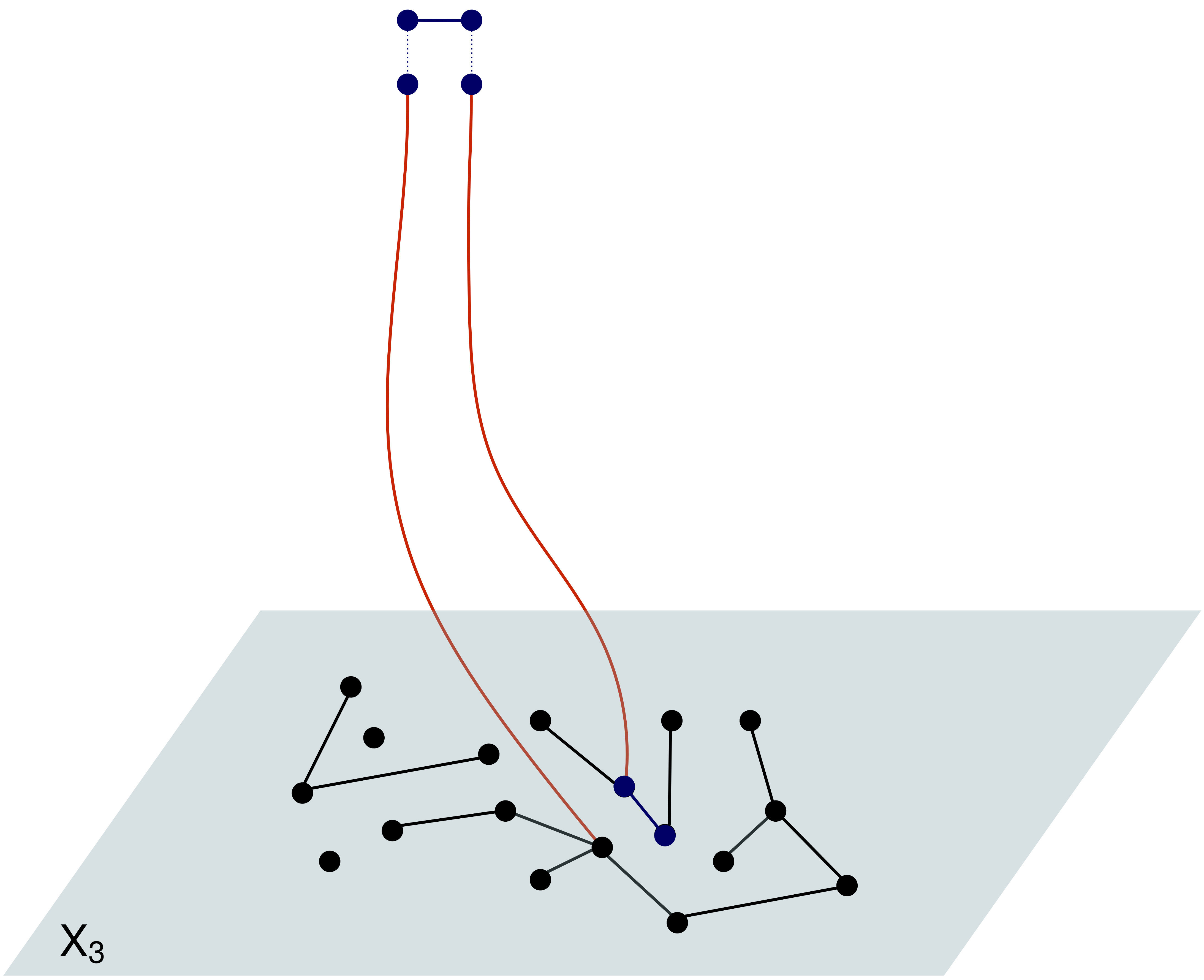
$$\begin{tikzcd} O_1 \arrow[d, hook] & I_1 \arrow[l, hook', "r_1"'] \arrow[d, hook, "m_1"] \\ X_1 & X_0 \arrow[l, hook'] \end{tikzcd}$$

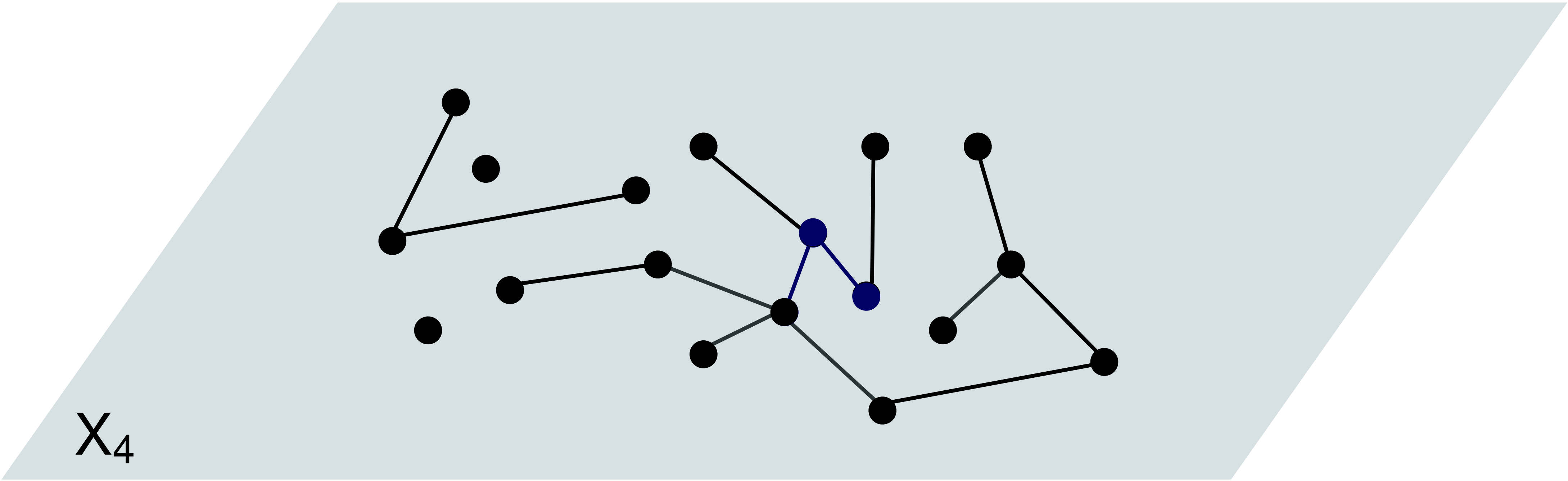

X_1

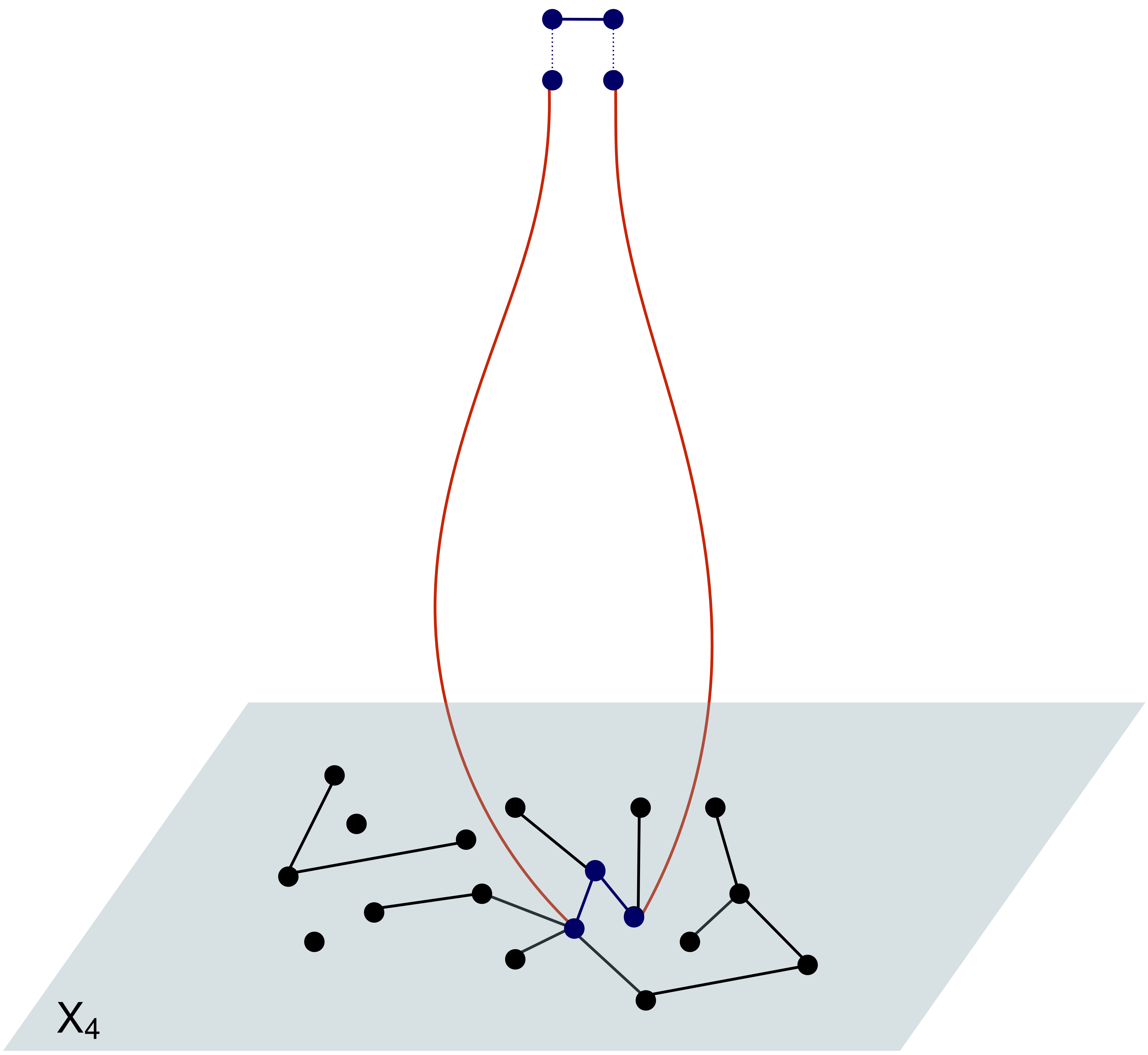

X_2

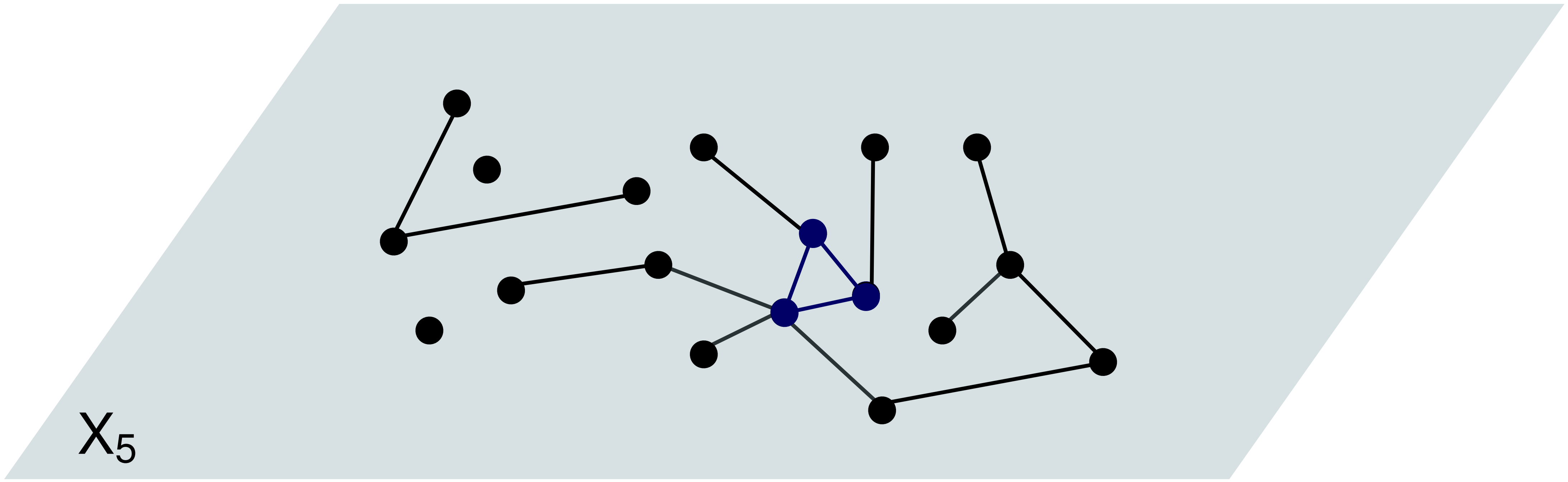

X_2

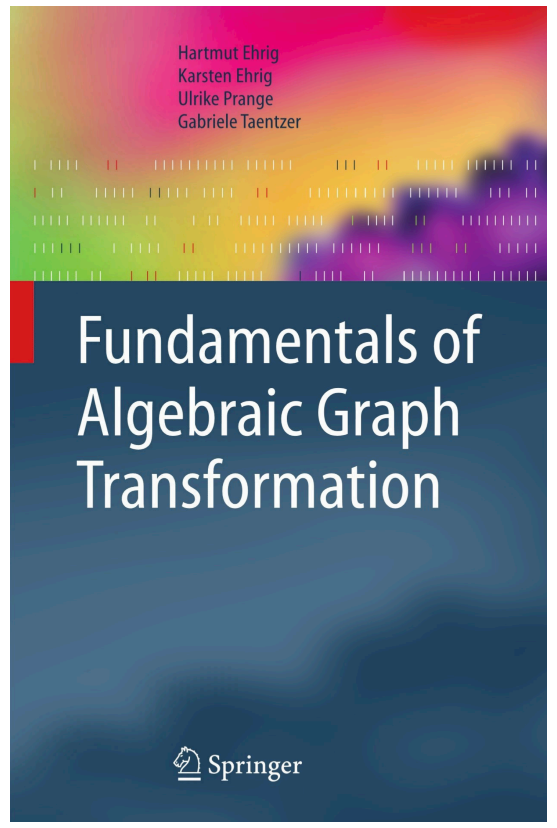

X_3


X_3

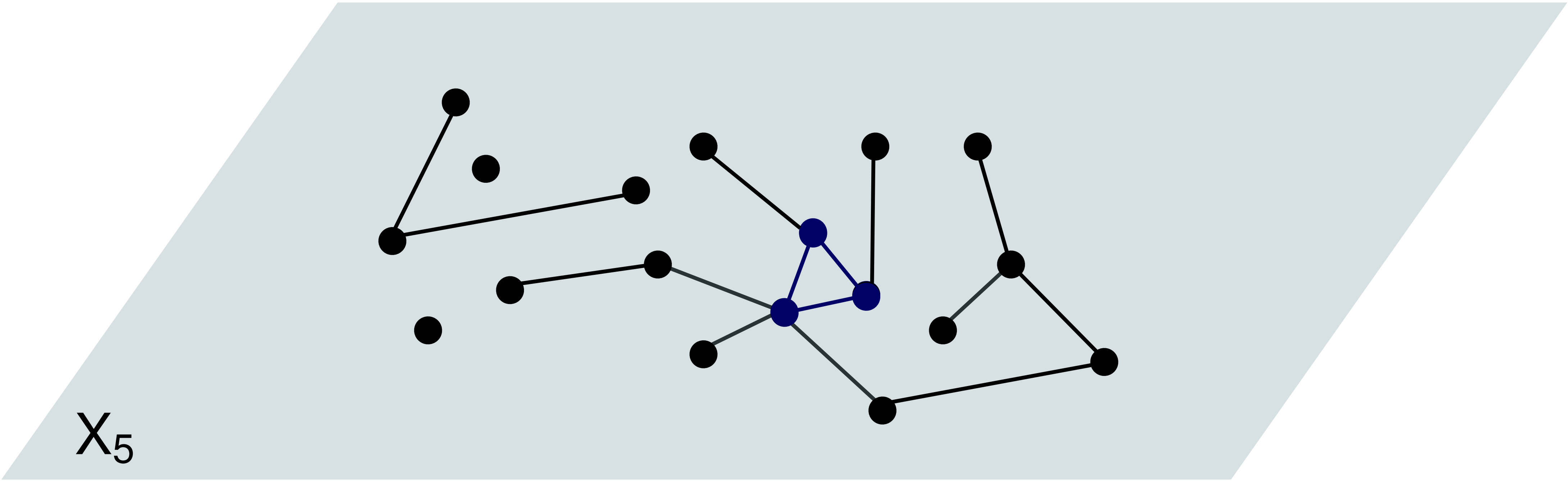

X4

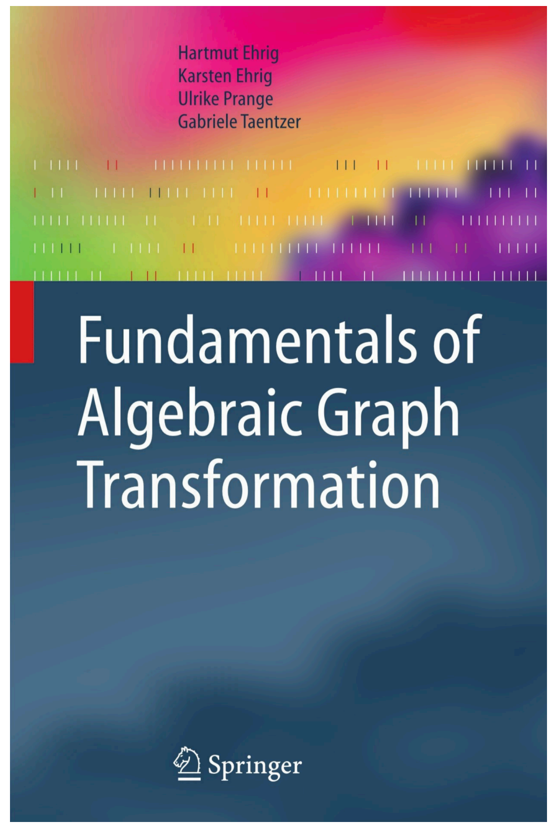

X_4


X5

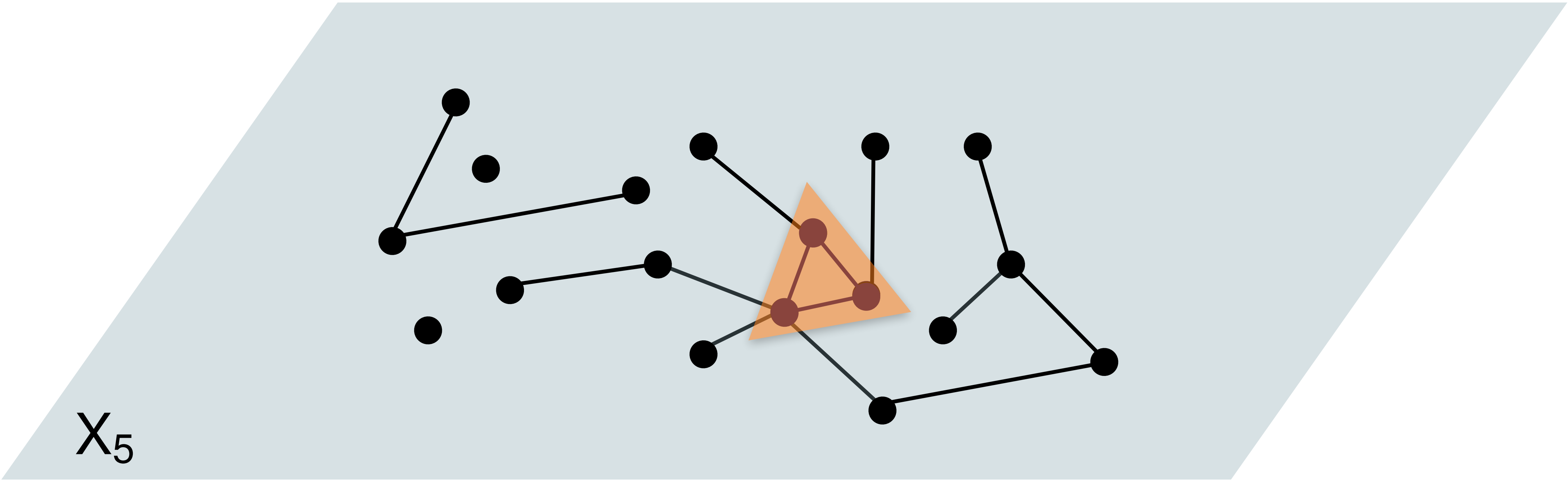


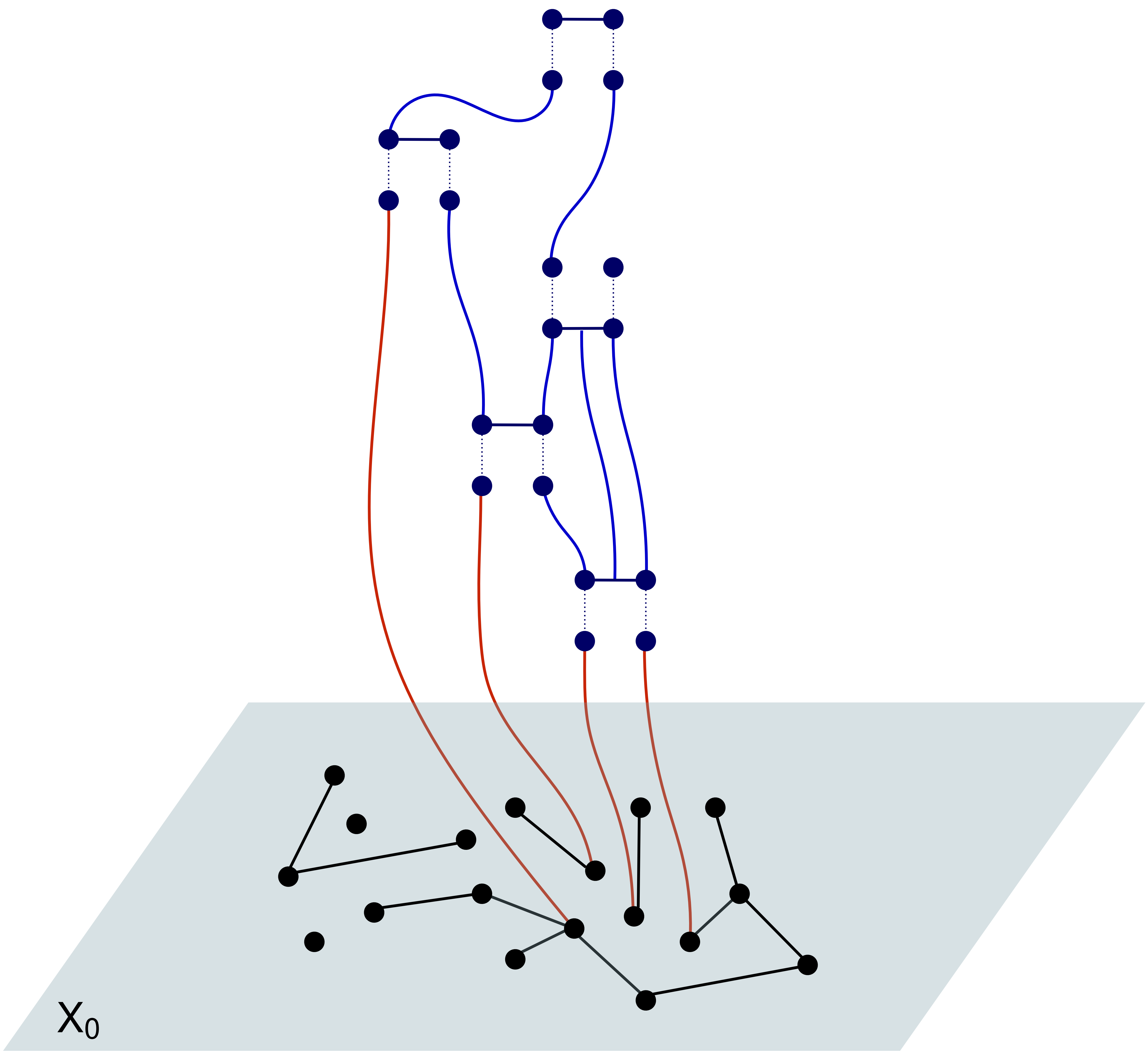
categorical rewriting theory
(since the mid-70's)

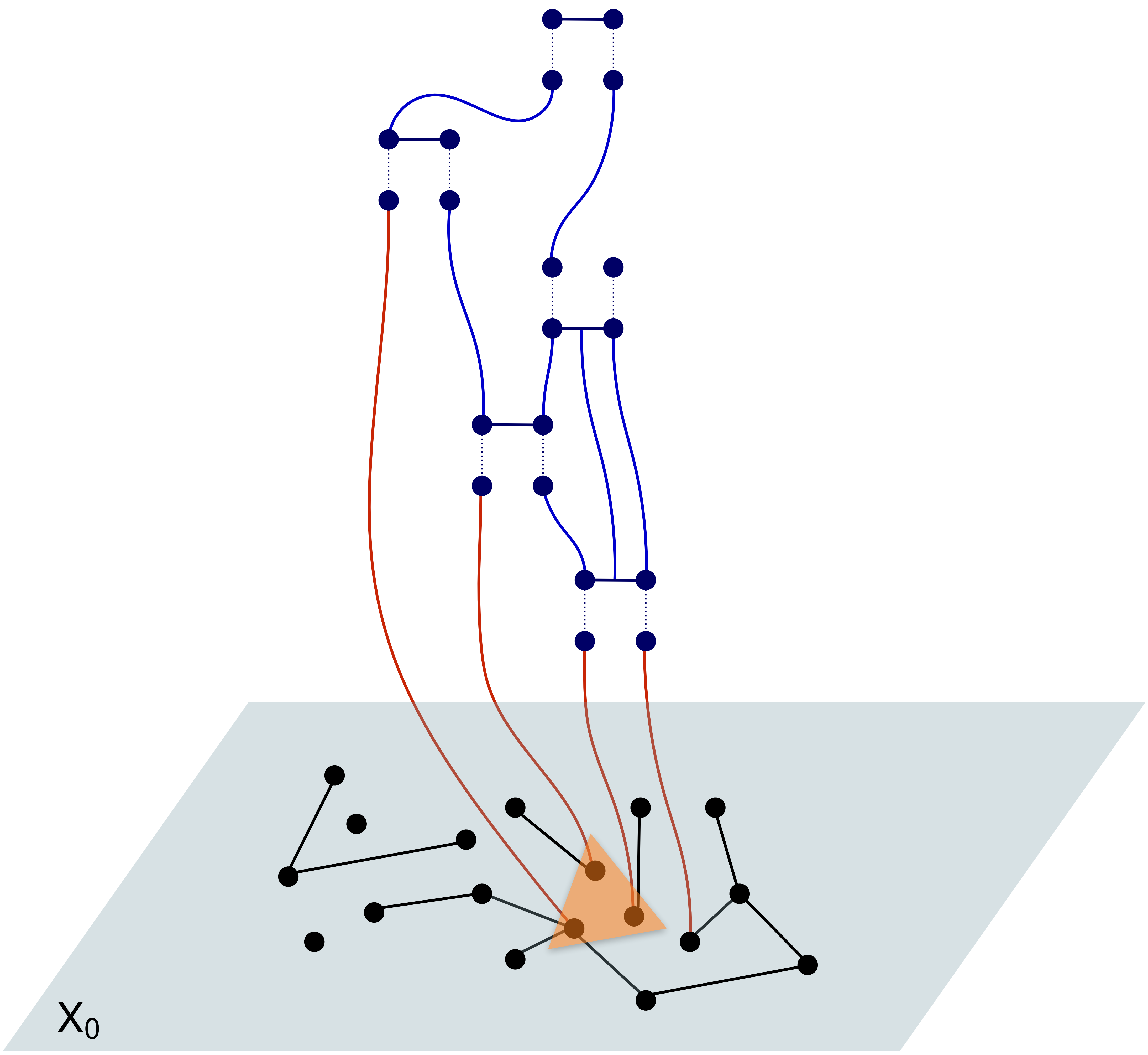


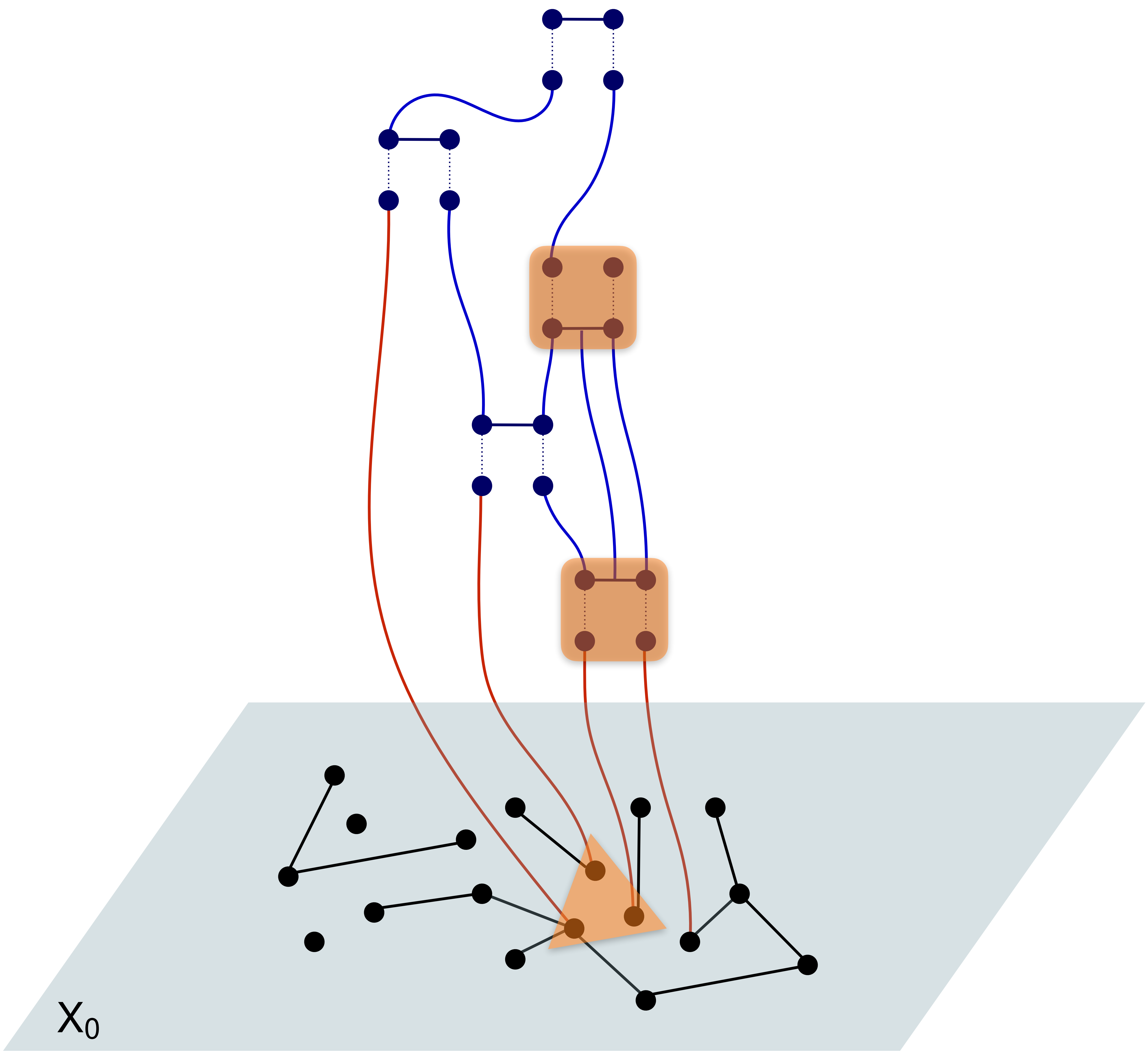


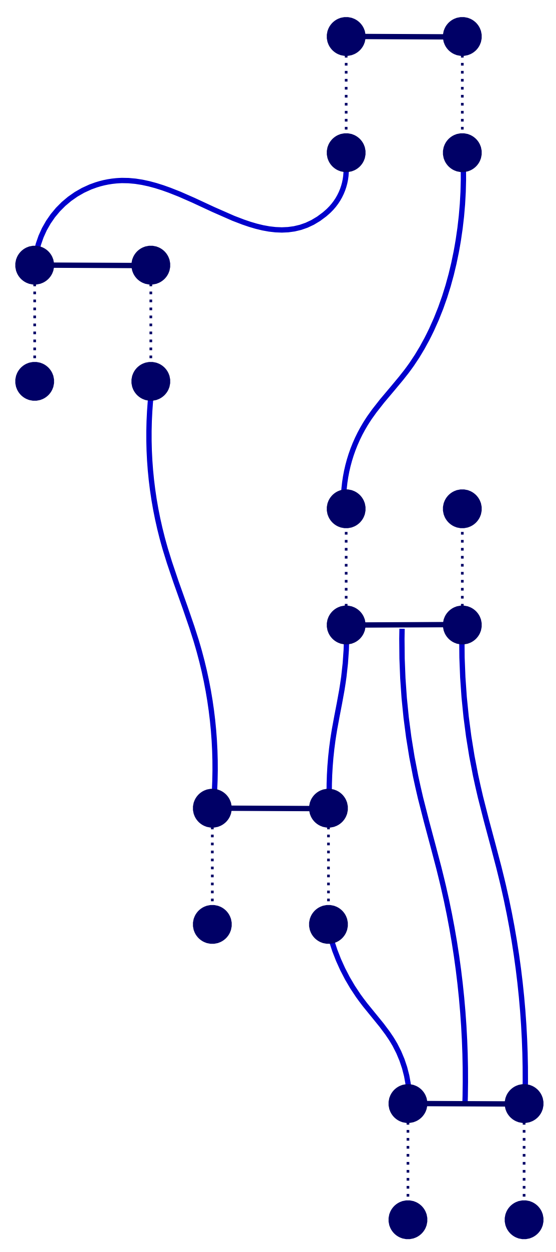
categorical rewriting theory
(since the mid-70's)

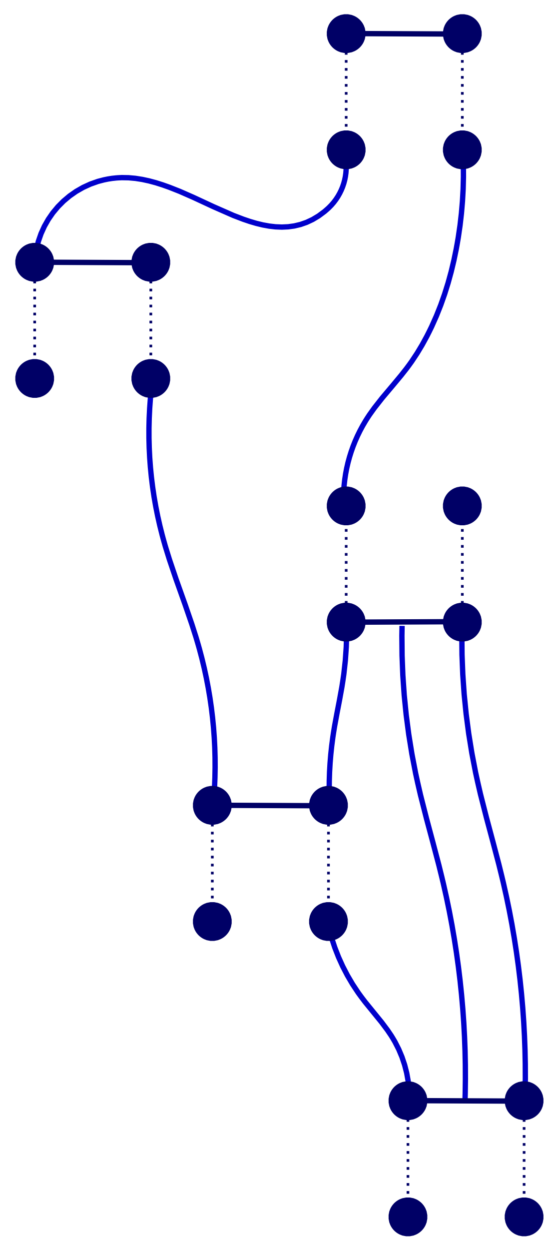



X_0


X0


X_0





a **TRACELET**
(of length 5)

The suitable notion of **rule compositions**

Definition 3. Let $\mathsf{R}_1, \mathsf{R}_2 \in \overline{\mathsf{Lin}}(\mathbf{C})_\sim$ be two equivalence classes of rules with conditions, and let $\mathsf{r}_j \in \mathsf{Lin}(\mathbf{C})$ and $\mathsf{c}_{\mathsf{I}_j}$ be concrete representatives of R_j (for $\mathsf{j} = 1, 2$). For $\mathbb{T} \in \{\mathsf{DPO}, \mathsf{SqPO}\}$, an $\mathcal{M}$-span $\mu = (\mathsf{I}_2 \hookleftarrow \mathsf{M}_{21} \hookrightarrow \mathsf{O}_1)$ (i.e. with $(\mathsf{M}_{21} \hookrightarrow \mathsf{O}_1), (\mathsf{M}_{21} \hookrightarrow \mathsf{I}_2) \in \mathcal{M}$) is a **$\mathbb{T}$-admissible match of** R_2 **into** R_1 if the diagram below is constructable (with N_{21} constructed by taking pushout)

$$O_2 \xleftarrow{r_2} I_2 \hookleftarrow M_{21} \hookrightarrow O_1 \xleftarrow{r_1} I_1 \tag{4}$$

The suitable notion of **rule compositions**

Definition 3. Let $\mathsf{R}_1, \mathsf{R}_2 \in \overline{\mathsf{Lin}}(\mathbf{C})_{\sim}$ be two equivalence classes of rules with conditions, and let $\mathsf{r}_j \in \mathsf{Lin}(\mathbf{C})$ and $\mathsf{c}_{\mathsf{I}_j}$ be concrete representatives of R_j (for $\mathsf{j} = 1, 2$). For $\mathbb{T} \in \{\mathsf{DPO}, \mathsf{SqPO}\}$, an $\mathcal{M}$-span $\mu = (\mathsf{I}_2 \hookleftarrow \mathsf{M}_{21} \hookrightarrow \mathsf{O}_1)$ (i.e. with $(\mathsf{M}_{21} \hookrightarrow \mathsf{O}_1), (\mathsf{M}_{21} \hookrightarrow \mathsf{I}_2) \in \mathcal{M}$) is a **$\mathbb{T}$-admissible match of R_2 into R_1** if the diagram below is constructable (with N_{21} constructed by taking pushout)

$$\begin{tikzcd}
O_2 & I_2 \arrow[l, hook', "r_2"'] \arrow[dr, hook] & M_{21} \arrow[l, hook'] \arrow[r, hook] \arrow[d, phantom, "\mathsf{PO}"] & O_1 \arrow[dl, hook'] & I_1 \arrow[l, hook', "r_1"'] \\
 & & N_{21} & &
\end{tikzcd} \tag{4}$$

The suitable notion of **rule compositions**

Definition 3. Let $R_1, R_2 \in \overline{\mathsf{Lin}}(\mathbf{C})_{\sim}$ be two equivalence classes of rules with conditions, and let $r_j \in \mathsf{Lin}(\mathbf{C})$ and c_{I_j} be concrete representatives of R_j (for $j = 1, 2$). For $\mathbb{T} \in \{\mathsf{DPO}, \mathsf{SqPO}\}$, an $\mathcal{M}$-span $\mu = (I_2 \hookleftarrow M_{21} \hookrightarrow O_1)$ (i.e. with $(M_{21} \hookrightarrow O_1), (M_{21} \hookrightarrow I_2) \in \mathcal{M}$) is a **$\mathbb{T}$-admissible match of** R_2 **into** R_1 if the diagram below is constructable (with N_{21} constructed by taking pushout)

$$
\begin{tikzcd}
O_2 \arrow[d, hook] & & I_2 \arrow[ll, hook', "r_2"'] \arrow[dr] & M_{21} \arrow[l, hook'] \arrow[r, hook] \arrow[d, phantom, "\mathsf{PO}"] & O_1 \arrow[dl] & & I_1 \arrow[ll, hook', "r_1"'] \\
O_{21} \arrow[urr, phantom, "\mathbb{T}"] & & & N_{21} \arrow[lll, hook'] & & &
\end{tikzcd}
\tag{4}
$$

The suitable notion of **rule compositions**

Definition 3. Let $\mathsf{R}_1, \mathsf{R}_2 \in \overline{\mathsf{Lin}}(\mathbf{C})_\sim$ be two equivalence classes of rules with conditions, and let $\mathsf{r}_\mathsf{j} \in \mathsf{Lin}(\mathbf{C})$ and $\mathsf{c}_{\mathsf{I}_\mathsf{j}}$ be concrete representatives of R_j (for $\mathsf{j} = 1, 2$). For $\mathbb{T} \in \{\mathsf{DPO}, \mathsf{SqPO}\}$, an $\mathcal{M}$-span $\mu = (\mathsf{I}_2 \hookleftarrow \mathsf{M}_{21} \hookrightarrow \mathsf{O}_1)$ (i.e. with $(\mathsf{M}_{21} \hookrightarrow \mathsf{O}_1), (\mathsf{M}_{21} \hookrightarrow \mathsf{I}_2) \in \mathcal{M}$) is a **$\mathbb{T}$-admissible match of** R_2 **into** R_1 if the diagram below is constructable (with N_{21} constructed by taking pushout)

$$\begin{tikzcd}
O_2 \arrow[d, hook] \arrow[dr, phantom, "\mathbb{T}"] & I_2 \arrow[l, hook', "r_2"'] \arrow[dr, bend right, hook] & & M_{21} \arrow[ll, hook'] \arrow[rr, hook] \arrow[d, phantom, "\mathsf{PO}"] & & O_1 \arrow[dll, bend left, hook'] \arrow[dr, phantom, "DPO^\dagger"] & I_1 \arrow[l, hook', "r_1"'] \arrow[d, hook] \\
O_{21} & & & N_{21} \arrow[lll, hook'] & & & I_{21} \arrow[lll, hook']
\end{tikzcd} \tag{4}$$

The suitable notion of **rule compositions**

Definition 3. Let $\mathsf{R}_1, \mathsf{R}_2 \in \overline{\mathsf{Lin}}(\mathbf{C})_{\sim}$ be two equivalence classes of rules with conditions, and let $\mathsf{r}_\mathsf{j} \in \mathsf{Lin}(\mathbf{C})$ and $\mathsf{c}_{\mathsf{I}_\mathsf{j}}$ be concrete representatives of R_j (for $\mathsf{j} = 1, 2$). For $\mathbb{T} \in \{\mathsf{DPO}, \mathsf{SqPO}\}$, an $\mathcal{M}$-span $\mu = (\mathsf{I}_2 \hookleftarrow \mathsf{M}_{21} \hookrightarrow \mathsf{O}_1)$ (i.e. with $(\mathsf{M}_{21} \hookrightarrow \mathsf{O}_1), (\mathsf{M}_{21} \hookrightarrow \mathsf{I}_2) \in \mathcal{M}$) is a **$\mathbb{T}$-admissible match of R_2 into R_1** if the diagram below is constructable (with N_{21} constructed by taking pushout)

$$
\begin{tikzcd}
O_2 \arrow[d] \arrow[dr, phantom, "\mathbb{T}"] & I_2 \arrow[l, "r_2"'] \arrow[dr, bend right] & M_{21} \arrow[l, hook'] \arrow[r, hook] \arrow[d, phantom, "\mathsf{PO}"] & O_1 \arrow[dl, bend left] & I_1 \arrow[l, "r_1"'] \arrow[d] \arrow[dl, phantom, "DPO^{\dagger}"] \\
O_{21} & & N_{21} \arrow[ll] & & I_{21} \arrow[ll]
\end{tikzcd}
\tag{4}
$$

and if $\mathsf{c}_{\mathsf{I}_{21}} \not\equiv \mathsf{false}$. Here, the condition $\mathsf{c}_{\mathsf{I}_{21}}$ is computed as

$$
\mathsf{c}_{I_{21}} := \mathsf{Shift}(I_1 \hookrightarrow I_{21}, \mathsf{c}_{I_1}) \;\wedge\; \mathsf{Trans}(N_{21} \leftharpoonup I_{21}, \mathsf{Shift}(I_2 \hookrightarrow N_{21}, \mathsf{c}_{I_2}))\,. \tag{5}
$$

The suitable notion of **rule compositions**

Definition 3. Let $\mathsf{R}_1, \mathsf{R}_2 \in \overline{\mathsf{Lin}}(\mathbf{C})_{\sim}$ be two equivalence classes of rules with conditions, and let $\mathsf{r}_j \in \mathsf{Lin}(\mathbf{C})$ and $\mathsf{c}_{\mathsf{I}_j}$ be concrete representatives of R_j (for $j = 1, 2$). For $\mathbb{T} \in \{\mathsf{DPO}, \mathsf{SqPO}\}$, an $\mathcal{M}$-span $\mu = (\mathsf{I}_2 \hookleftarrow \mathsf{M}_{21} \hookrightarrow \mathsf{O}_1)$ (i.e. with $(\mathsf{M}_{21} \hookrightarrow \mathsf{O}_1), (\mathsf{M}_{21} \hookrightarrow \mathsf{I}_2) \in \mathcal{M}$) is a **$\mathbb{T}$-admissible match of** R_2 **into** R_1 if the diagram below is constructable (with N_{21} constructed by taking pushout)

$$\begin{tikzcd}
O_2 \arrow[d] \arrow[dr, phantom, "\mathbb{T}"] & I_2 \arrow[l, "r_2"'] \arrow[drr, bend right] & & M_{21} \arrow[ll, hook'] \arrow[rr, hook] \arrow[d, phantom, "\mathsf{PO}"] & & O_1 \arrow[dll, bend left] \arrow[dr, phantom, "DPO^{\dagger}"] & I_1 \arrow[l, "r_1"'] \arrow[d] \\
O_{21} & & & N_{21} \arrow[lll] & & & I_{21} \arrow[lll]
\end{tikzcd} \tag{4}$$

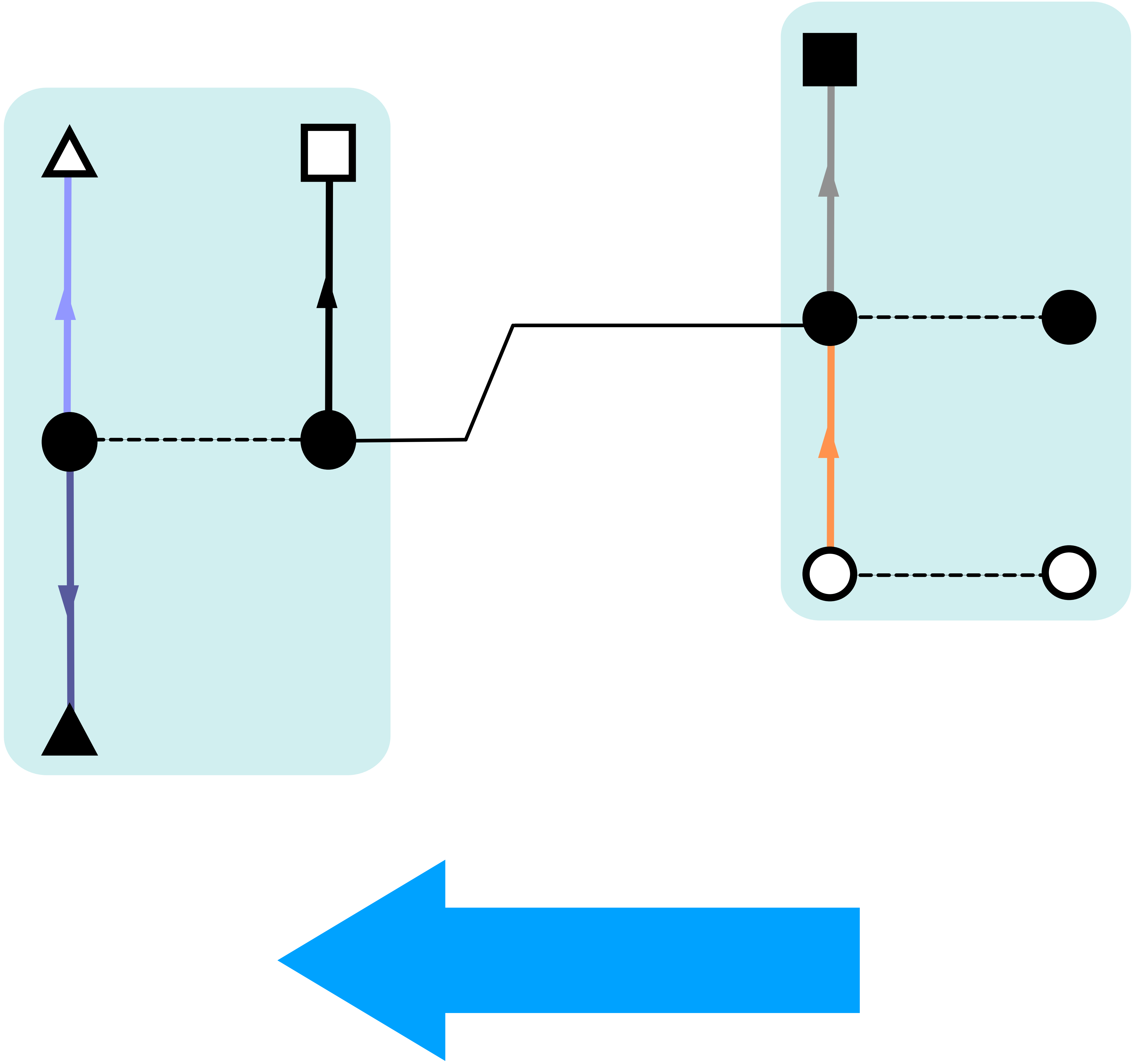
and if $\mathsf{c}_{\mathsf{I}_{21}} \not\equiv \mathsf{false}$. Here, the condition $\mathsf{c}_{\mathsf{I}_{21}}$ is computed as

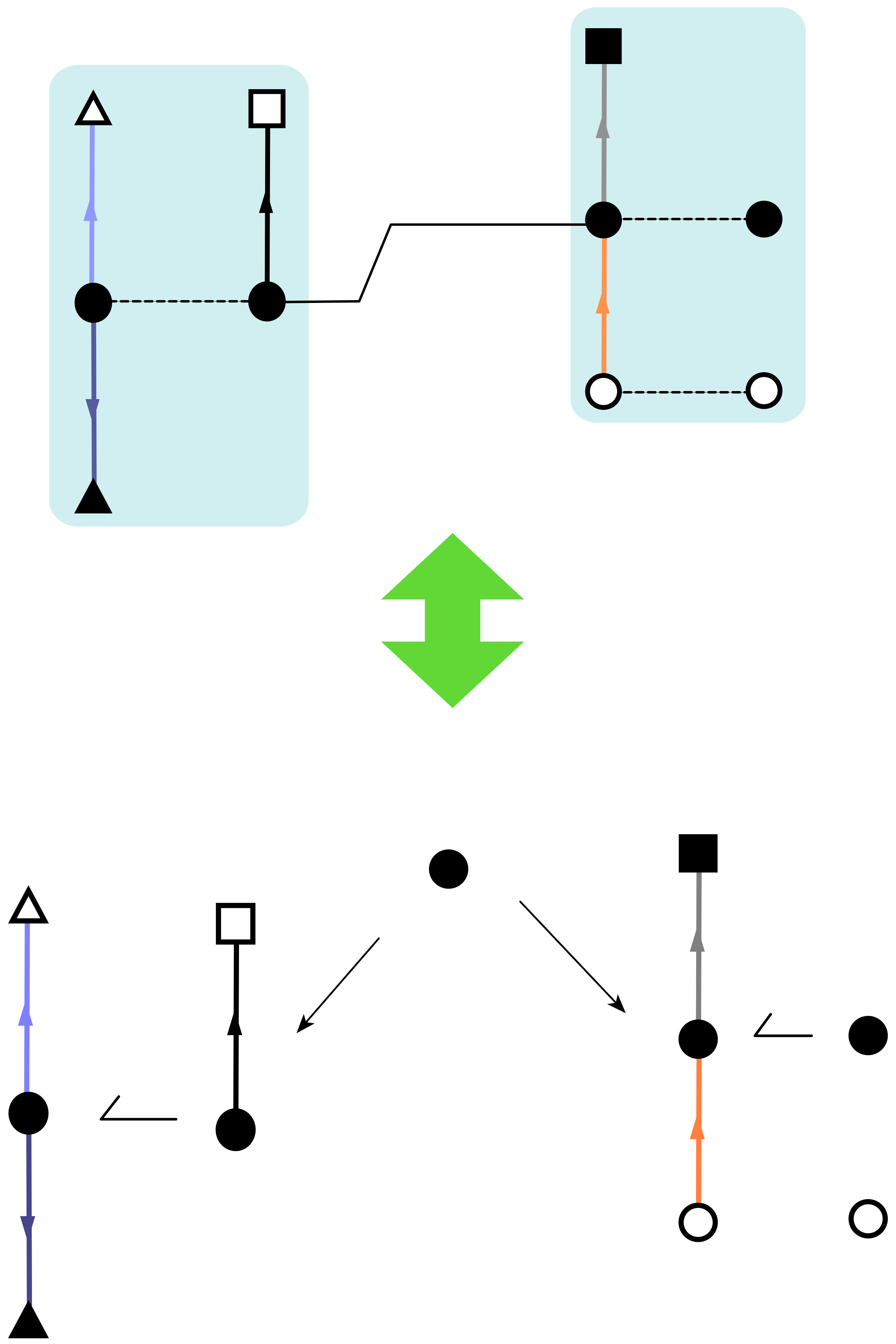
$$\mathsf{c}_{I_{21}} := \mathsf{Shift}(I_1 \hookrightarrow I_{21}, \mathsf{c}_{I_1}) \ \wedge \ \mathsf{Trans}(N_{21} \leftharpoonup I_{21}, \mathsf{Shift}(I_2 \hookrightarrow N_{21}, \mathsf{c}_{I_2}))\,. \tag{5}$$

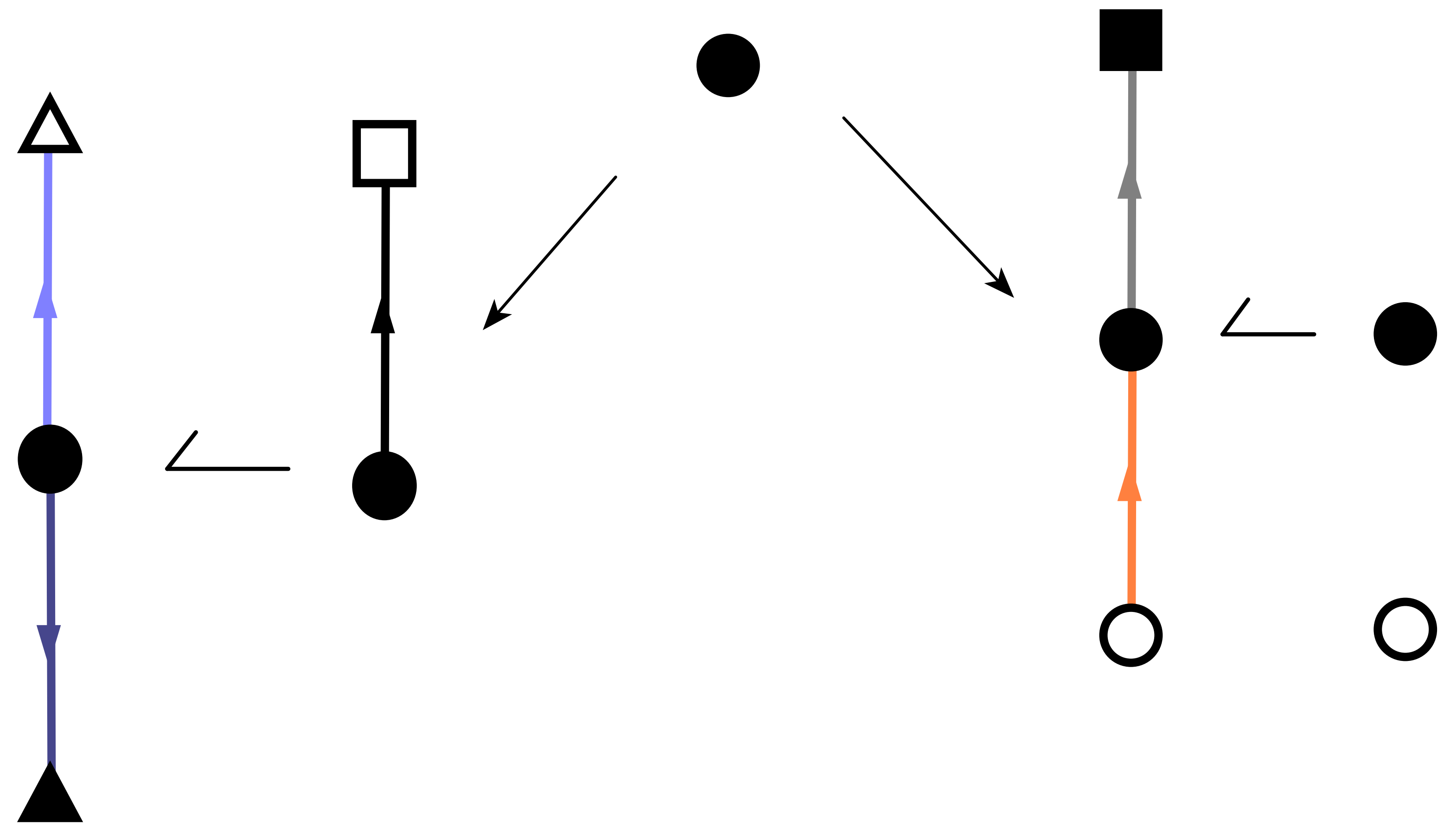
In this case, we define the **type $\mathbb{T}$ composition of** R_2 **with** R_1 **along** μ, denoted $\mathsf{R}_2{}^{\mu}\!\triangleleft_{\mathbb{T}}\mathsf{R}_1$, as

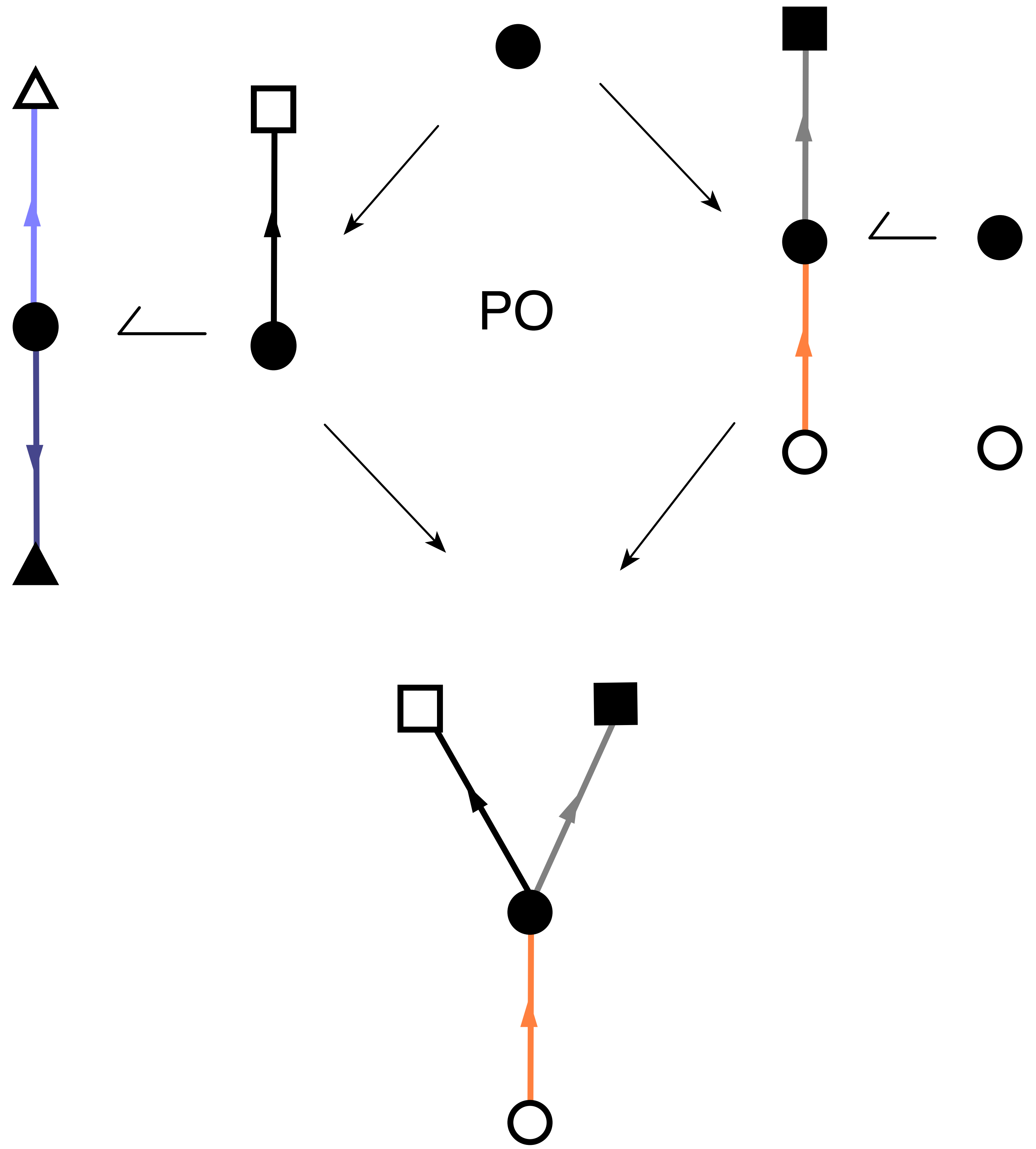
$$R_2{}^{\mu}\!\triangleleft_{\mathbb{T}} R_1 := [(O_{21} \leftharpoonup I_{21}; \mathsf{c}_{I_{21}})]_{\sim}\,, \tag{6}$$

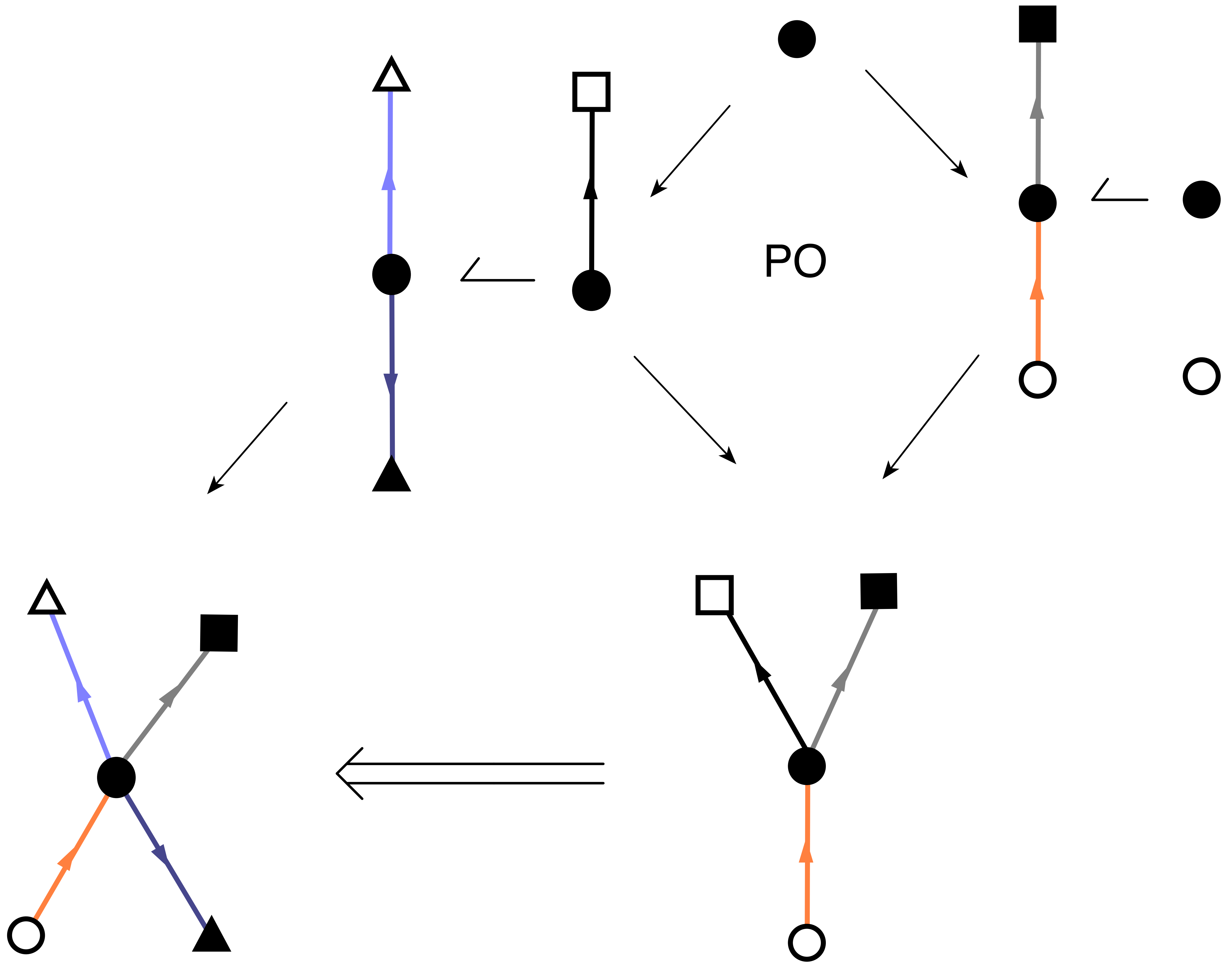
where $(\mathsf{O}_{21} \leftharpoonup \mathsf{I}_{21}) := (\mathsf{O}_{21} \leftharpoonup \mathsf{N}_{21}) \circ (\mathsf{N}_{21} \leftharpoonup \mathsf{I}_{21})$ (with $\circ$ the **span composition** operation).

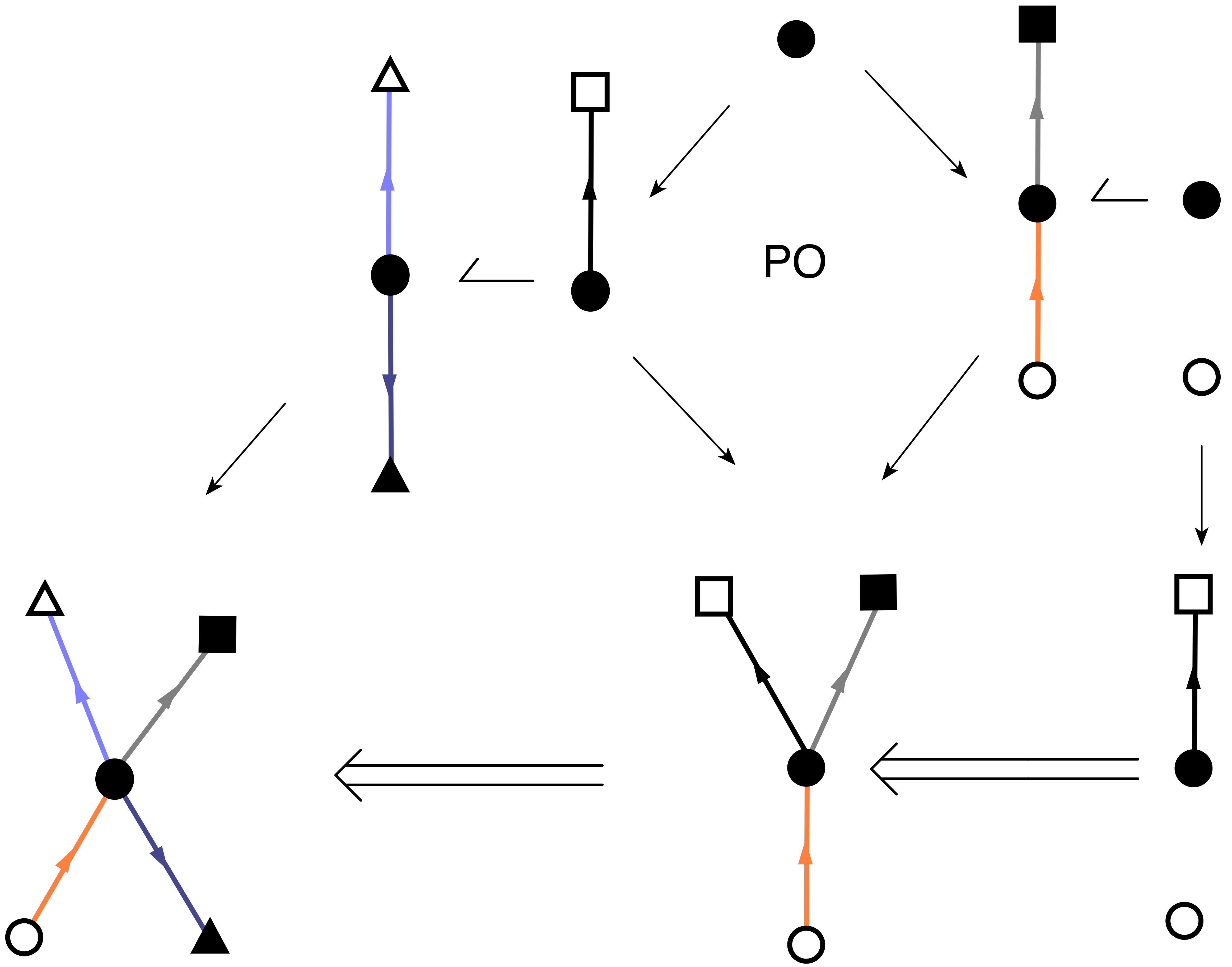







PO


PO


PO

I. Categorical Rewriting Theory

II. **Rule Algebra Framework**

III. Rule-Algebraic Combinatorics

Physics insight: the **rule algebra** formalism

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

a **rule**

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

Physics insight: the **rule algebra** formalism

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

a **rule**

a **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

$$\delta(r_2) *_{\mathcal{R}} \delta(r_1) := \sum_{\mu \in M_{r_2}(r_1)} \delta\left(r_2 \overset{\mu}{\blacktriangleleft} r_1\right)$$

"sum over ways to compose the rules"

Physics insight: the **rule algebra** formalism

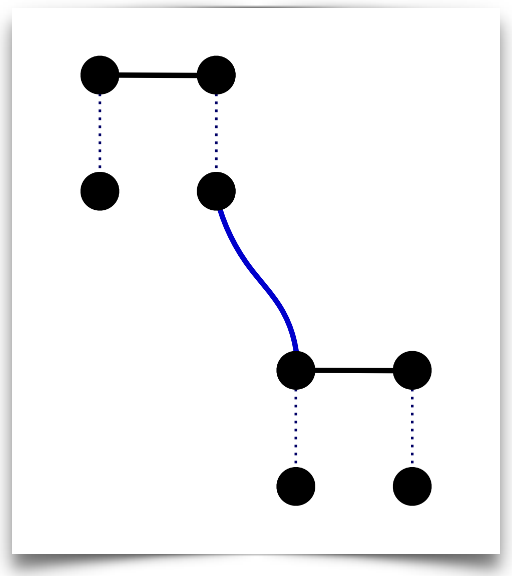
$$\left(O \xleftarrow{r} I\right) \rightsquigarrow \delta\left(O \xleftarrow{r} I\right)$$

a **rule** — a **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

$$\delta(r_2) *_{\mathcal{R}} \delta(r_1) := \sum_{\mu \in M_{r_2}(r_1)} \delta\left(r_2 \overset{\mu}{\blacktriangleleft} r_1\right)$$

"sum over ways to compose the rules"



$$\begin{tikzcd}
O_2 \arrow[d] & I_2 \arrow[l, hook', "r_2"'] \arrow[dr] & & O_1 \arrow[ll, hook', "\mu"'] \arrow[dl] & I_1 \arrow[l, hook', "r_1"'] \arrow[d] \\
O_{21} & & N_{21} \arrow[ll, hook', "r_2'"'] & & I_{21} \arrow[ll, hook', "r_1'"'] \arrow[llll, bend left, "r_2 \overset{\mu}{\blacktriangleleft} r_1 := r_2' \circ r_1'"]
\end{tikzcd}$$

Physics insight: the **rule algebra** formalism

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

$$\delta(\mathrm{r}_2) *_{\mathcal{R}} \delta(\mathrm{r}_1) := \sum_{\mu \in \mathsf{M}_{\mathrm{r}_2}(\mathrm{r}_1)} \delta\left(\mathrm{r}_2 \overset{\mu}{\blacktriangleleft} \mathrm{r}_1\right)$$

"sum over ways to compose the rules"

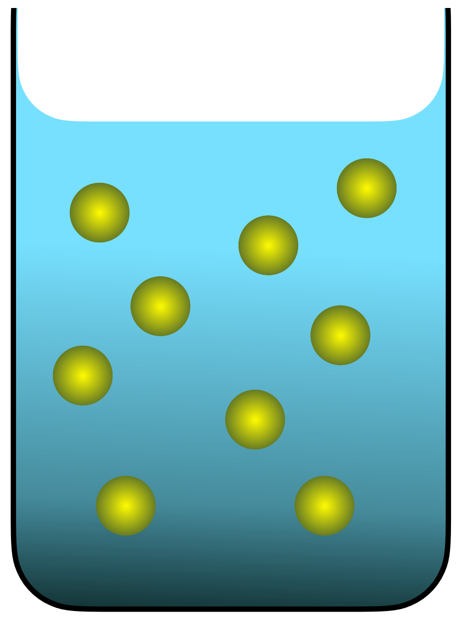
Theorem LiCS 2016, CSL 2018, GCM 2019, LMCS 2020, ICGT 2020

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

$\Rightarrow$ a new fundamental tool in **rewriting theory**, **combinatorics** and **concurrency theory**

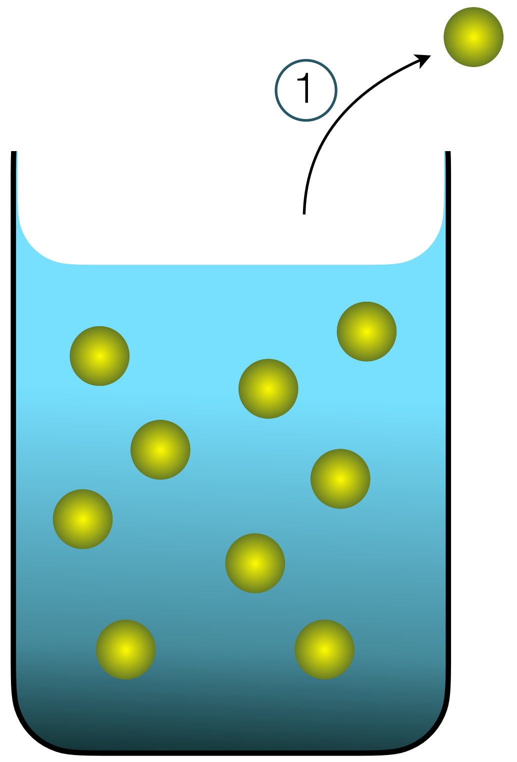
Mathematics of **chemical reactions**

Example: $2X \xleftharpoondown{\alpha} X \qquad (\alpha \in \mathbb{R}_{>0})$



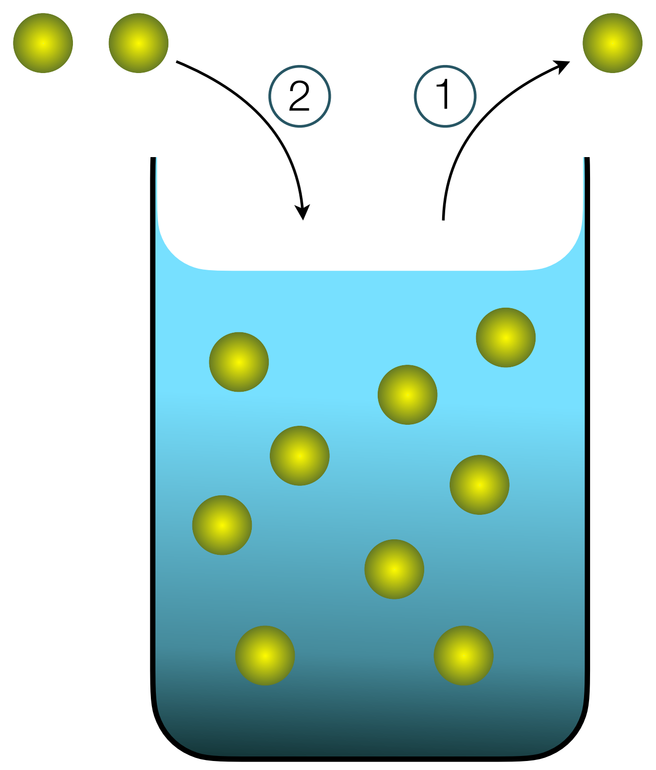
Mathematics of **chemical reactions**

Example: $2X \xleftharpoondown{\alpha} X \qquad (\alpha \in \mathbb{R}_{>0})$



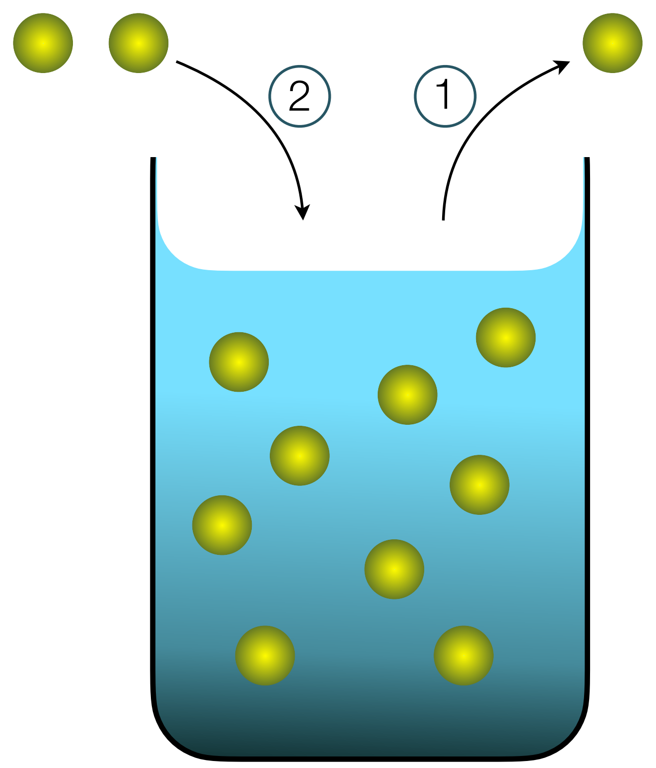
Mathematics of **chemical reactions**

Example: $2X \xleftharpoondown{\alpha} X \qquad (\alpha \in \mathbb{R}_{>0})$



Mathematics of **chemical reactions**

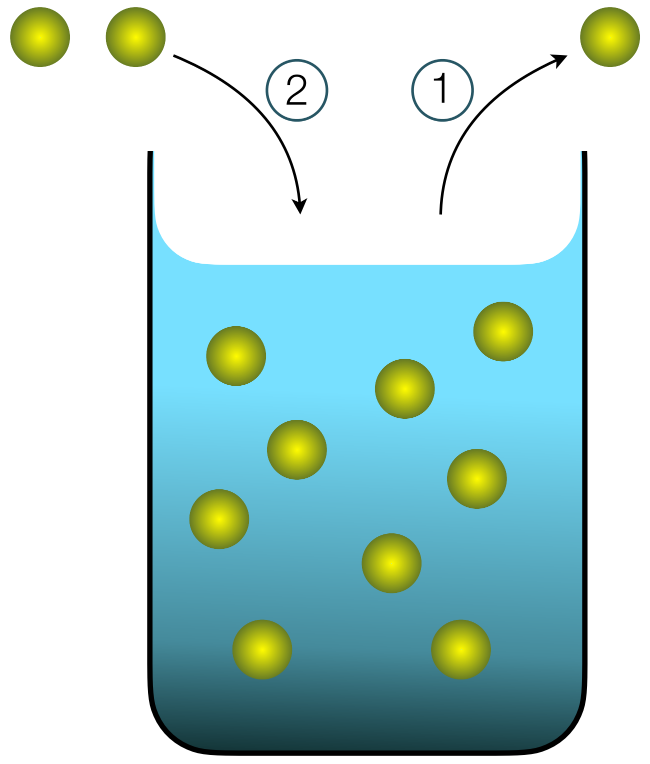
Example: $2X \xleftarrow{\alpha} X \qquad (\alpha \in \mathbb{R}_{>0})$



$p_n(t) := \Pr(\#X = n \text{ at time } t) = ?$

Mathematics of **chemical reactions**

Example: $2X \xleftarrow{\alpha} X \qquad (\alpha \in \mathbb{R}_{>0})$

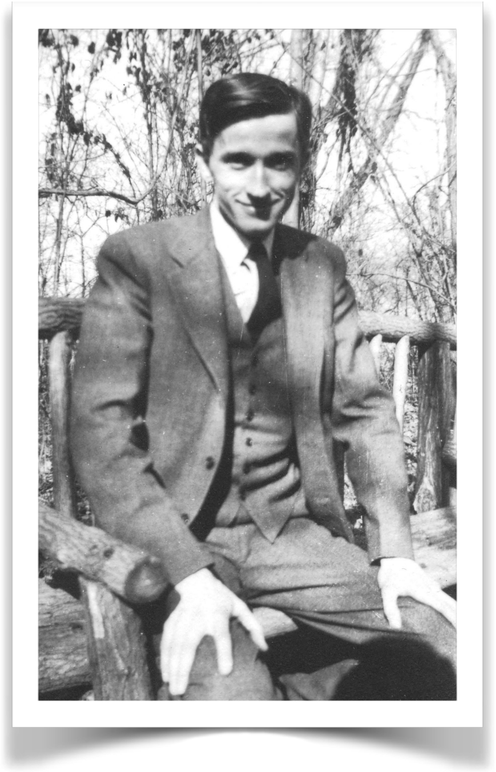


$$p_n(t) \;:=\; \Pr(\#X = n \text{ at time } t) \;=\; ?$$

Delbrück (1940): $$P(t;x) \;:=\; \sum_{n\geq 0} p_n(t)\, x^n$$

$$\partial_t P(t;x) = \boxed{\left[\alpha\left(\hat{x}^2\partial_x - \hat{x}\partial_x\right)\right]}\, P(t;x)$$

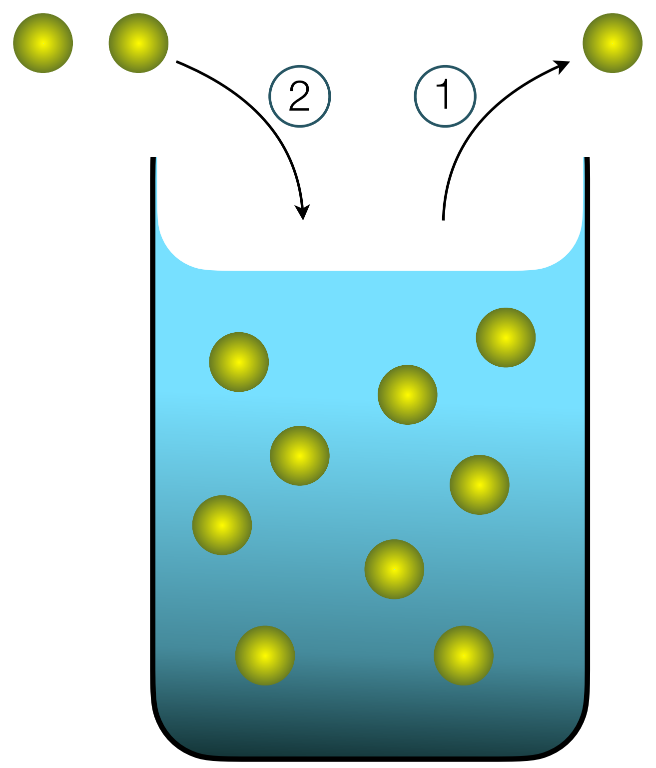
a **linear operator...**



Max Delbrück (1906-1981)
1969 **Nobel Prize** laureate
(medicine and physiology)

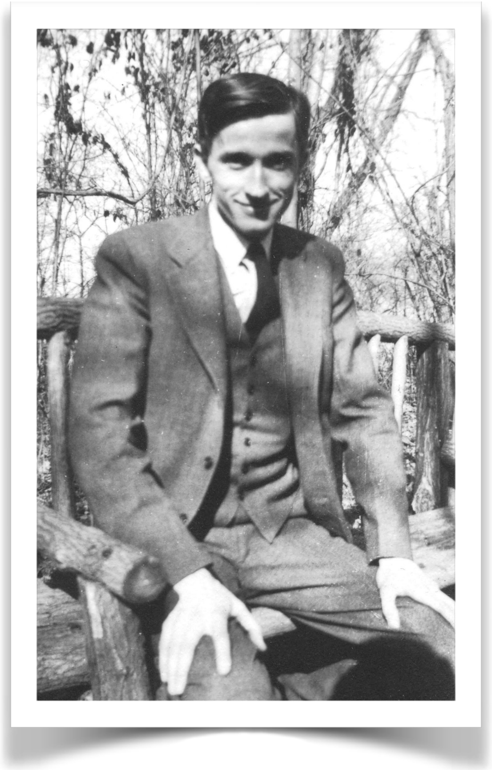
Mathematics of **chemical reactions**

Example: $2X \xleftarrow{\alpha} X \qquad (\alpha \in \mathbb{R}_{>0})$



$$p_n(t) \;:=\; \Pr(\#X = n \text{ at time } t) \;=\; ?$$

Delbrück (1940): $$P(t;x) \;:=\; \sum_{n\geq 0} p_n(t)\, x^n$$



Max Delbrück (1906-1981)
1969 **Nobel Prize** laureate
(medicine and physiology)

$$\partial_t P(t;x) = \boxed{\left[\alpha\left(\hat{x}^2\partial_x - \hat{x}\partial_x\right)\right]} \; P(t;x)$$

a **linear operator…**

$$\hat{x}(x^n) := x^{n+1}\,, \qquad \partial_x(x^n) := \begin{cases} 0 & \text{if } n = 0 \\ n \cdot x^{n-1} & \text{if } n > 0 \end{cases}$$

Rule algebra framework (Part II)

Observation: x^n — **basis vector** (of the vector space of polynomials in x)

Rule algebra framework (Part II)

Observation: x^n — **basis vector** (of the vector space of polynomials in x)

$\Rightarrow$ **analogous concept** for **rewriting theory**:

$|X\rangle$ — **basis vector** (of a vector space of **configurations** $\hat{\mathbf{C}}$, e.g. **graphs**, **trees**, **molecules**, ….)

Rule algebra framework (Part II)

Observation: x^n — **basis vector** (of the vector space of polynomials in x)

$\Rightarrow$ **analogous concept** for **rewriting theory**:

$|X\rangle$ — **basis vector** (of a vector space of **configurations** $\hat{\mathbf{C}}$,
e.g. **graphs**, **trees**, **molecules**, ….)

Example: $|n\rangle := \underbrace{|\bullet \ldots \bullet\rangle}_{\mathbf{n}\text{ vertices}}$ (here: configuration = **iso-class** of graph)

Rule algebra framework (Part II)

Observation: x^n — **basis vector** (of the vector space of polynomials in x)

$\Rightarrow$ **analogous concept** for **rewriting theory**:

$|X\rangle$ — **basis vector** (of a vector space of **configurations** $\hat{\mathbf{C}}$,
e.g. **graphs**, **trees**, **molecules**, ….)

Example: $|n\rangle := |\underbrace{\bullet \ldots \bullet}_{\mathbf{n}\text{ vertices}}\rangle$ (here: configuration = **iso-class** of graph)

Key step: from **rules** to **linear operators** on $\hat{\mathbf{C}}$

$$\rho(\delta(r))\,|X\rangle := \sum_{m \in M_r(X)} |r_m(X)\rangle$$

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

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

Rule algebra framework (Part II)

$$\rho\big(\delta(r)\big)\,|X\rangle := \sum_{m \in M_r(X)} |r_m(X)\rangle$$

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

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

$$\rho\big(\delta(r_2)\big)\rho\big(\delta(r_1)\big)\,|X\rangle = \rho\big(\delta(r_2) *_{\mathcal{R}} \delta(r_1)\big)\,|X\rangle$$

Rule algebra framework (Part II)

$$\rho(\delta(r))\,|X\rangle := \sum_{m\in M_r(X)} |r_m(X)\rangle$$

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

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

$$\rho\big(\delta(r_2)\big)\rho\big(\delta(r_1)\big)\,|X\rangle = \rho\big(\delta(r_2) *_{\mathcal{R}} \delta(r_1)\big)\,|X\rangle$$

Example:

$$|n\rangle := \underbrace{|\bullet \ldots \bullet\rangle}_{\mathbf{n}\text{ vertices}} \quad \leftrightarrow \quad x^n$$

$$\rho\big(\delta(\bullet \leftharpoonup \varnothing)\big)\,|n\rangle = |n+1\rangle \quad \leftrightarrow \quad \hat{x}\big(x^n\big) = x^{n+1}$$

$$\rho\big(\delta(\varnothing \leftharpoonup \bullet)\big)\,|n\rangle = \begin{cases} 0 & \text{if } n = 0 \\ n \cdot |n-1\rangle & \text{if } n > 0 \end{cases} \quad \leftrightarrow \quad \partial_x\big(x^n\big) = \begin{cases} 0 & \text{if } n = 0 \\ n \cdot x^{n-1} & \text{if } n > 0 \end{cases}$$

Rule algebra framework (Part II)

$$\rho(\delta(r))\,|X\rangle := \sum_{m \in M_r(X)} |r_m(X)\rangle$$

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

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

$$\rho\big(\delta(r_2)\big)\rho\big(\delta(r_1)\big)\,|X\rangle = \rho\big(\delta(r_2) *_{\mathcal{R}} \delta(r_1)\big)\,|X\rangle$$

$$|n\rangle := \underbrace{|\bullet \ldots \bullet\rangle}_{\mathbf{n}\text{ vertices}}$$

Example: $\leftrightarrow$ x^n

$$\rho\big(\delta(\bullet \leftharpoonup \varnothing)\big)\,|n\rangle = |n+1\rangle \quad \leftrightarrow \quad \hat{x}\big(x^n\big) = x^{n+1}$$

$$\rho\big(\delta(\varnothing \leftharpoonup \bullet)\big)\,|n\rangle = \begin{cases} 0 & \text{if } n = 0 \\ n \cdot |n-1\rangle & \text{if } n > 0 \end{cases} \quad \leftrightarrow \quad \partial_x\big(x^n\big) = \begin{cases} 0 & \text{if } n = 0 \\ n \cdot x^{n-1} & \text{if } n > 0 \end{cases}$$

Application to the case of the reaction $2X \xleftharpoonup{\alpha} X$ $\quad (\alpha \in \mathbb{R}_{>0})$

$$\alpha\left(\hat{x}^2\partial_x - \hat{x}\partial_x\right)$$

Rule algebra framework (Part II)

$$\rho(\delta(r))\,|X\rangle := \sum_{m \in M_r(X)} |r_m(X)\rangle$$

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

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

$$\rho(\delta(r_2))\rho(\delta(r_1))\,|X\rangle = \rho(\delta(r_2) *_{\mathcal{R}} \delta(r_1))\,|X\rangle$$

$$|n\rangle := \underbrace{|\bullet \ldots \bullet\rangle}_{\mathbf{n}\text{ vertices}} \qquad \textbf{Example:} \quad \leftrightarrow \quad x^n$$

$$\rho(\delta(\bullet \leftharpoonup \varnothing))\,|n\rangle = |n+1\rangle \quad \leftrightarrow \quad \hat{x}(x^n) = x^{n+1}$$

$$\rho(\delta(\varnothing \leftharpoonup \bullet))\,|n\rangle = \begin{cases} 0 & \text{if } n = 0 \\ n \cdot |n-1\rangle & \text{if } n > 0 \end{cases} \quad \leftrightarrow \quad \partial_x(x^n) = \begin{cases} 0 & \text{if } n = 0 \\ n \cdot x^{n-1} & \text{if } n > 0 \end{cases}$$

Application to the case of the reaction $2X \xleftharpoonup{\alpha} X \qquad (\alpha \in \mathbb{R}_{>0})$

$$\alpha\left(\rho(\delta(\bullet\,\bullet \leftharpoonup \bullet)) - \rho(\delta(\bullet \leftharpoonup \bullet))\right) \quad \leftrightarrow \quad \alpha\left(\hat{x}^2\partial_x - \hat{x}\partial_x\right)$$

$\Rightarrow$ Delbrück's evolution operator **explained via rewriting theory!**

I. Categorical Rewriting Theory

II. Rule Algebra Framework

III. Rule-Algebraic Combinatorics

Defining combinatorial structures via **generators**

Definition 4. Consider a rewriting system over some suitable category $\mathbf{C}$ that consists of a **finite set of rules with conditions** $R_1, \ldots, R_n \in \overline{\mathsf{Lin}}(\mathbf{C})$. For some choice of parameters $\gamma_1, \ldots, \gamma_n \in \mathbb{R}$, define a **linear operator**

$$\hat{G} := \sum_{j=1}^{n} \gamma_j \rho(\delta(R_j))\,. \tag{7}$$

Defining combinatorial structures via **generators**

Definition 4. Consider a rewriting system over some suitable category $\mathbf{C}$ that consists of a **finite set of rules with conditions** $R_1, \dots, R_n \in \overline{\mathsf{Lin}}(\mathbf{C})$. For some choice of parameters $\gamma_1, \dots, \gamma_n \in \mathbb{R}$, define a **linear operator**

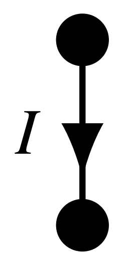
$$\hat{G} := \sum_{j=1}^{n} \gamma_j \rho(\delta(R_j))\,. \tag{7}$$

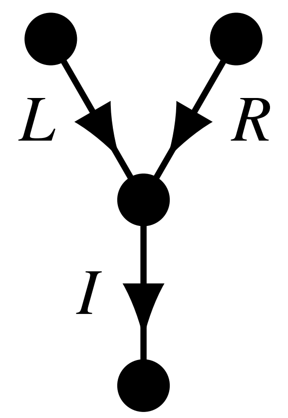
Note: $\widehat{G}$ has a natural interpretation as a linear operator that encodes **"application of the rules** $R_1, \dots, R_n$ **in all possible ways, and weighted by the parameters** $\gamma_1, \dots, \gamma_n$**"**, i.e. one may view $\widehat{G}$ as the (weighted) **generator** of a (countable) **set of structures** $S_{\widehat{G}}$,

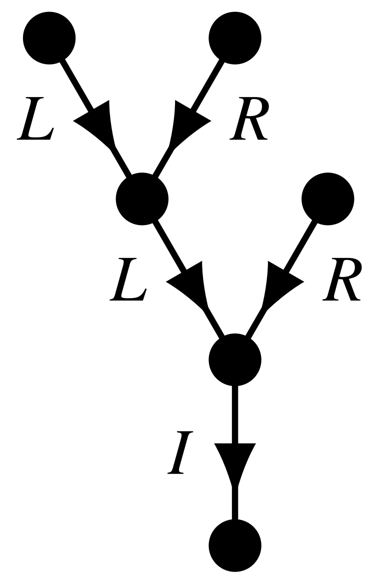
$$S_{\hat{G}} := \bigcup_{n>0} S_{\hat{G}}^{(n)}\,, \quad S_{\hat{G}}^{(n)} := \begin{cases} \{X_0\} & \text{if } n = 0 \\ \{X \in \mathsf{obj}(\mathbf{C})_{\cong} \mid X_0 \Rightarrow_{(n)} X\} & \text{if } n > 0\,. \end{cases} \tag{8}$$

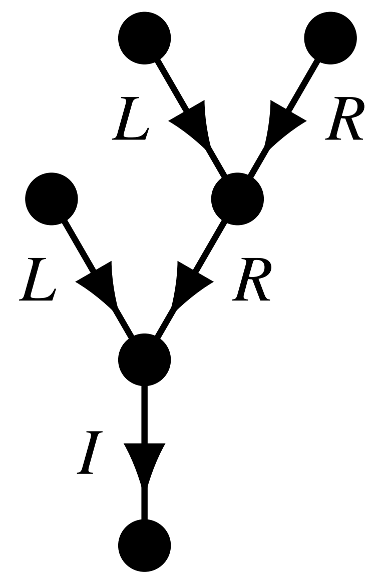
Here, $\{\Rightarrow_{(i)}\}_{i>0}$ denotes the **reachability relation** on $\mathsf{obj}(\mathbf{C})_{\cong}^{\times 2}$ with respect to the **initial configuration** $X_0 \in \mathsf{obj}(\mathbf{C})_{\cong}$ and the **rule-set** $\{R_j\}_{h=1}^{n}$ used to define $\widehat{G}$.

Running example: **planar rooted binary trees (PRBTs)**

$\mathcal{T}_0 := \{$ 

$\}$, $\mathcal{T}_1 := \{$ 

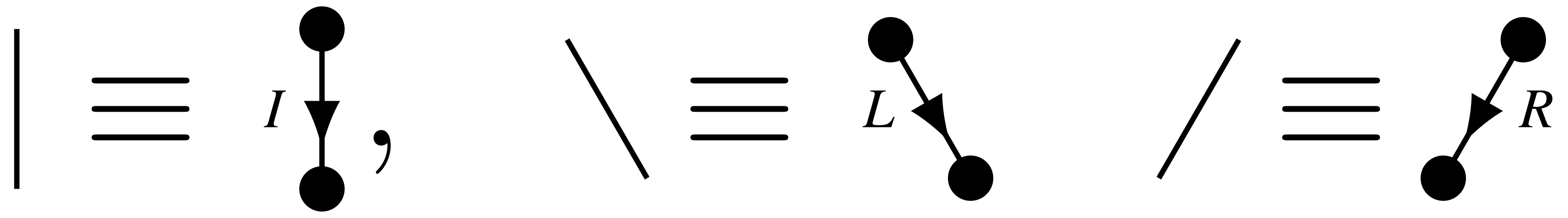
$\}$, $\mathcal{T}_2 := \{$ 

, 

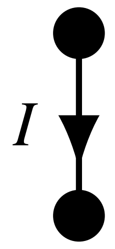
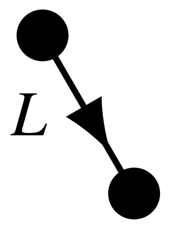
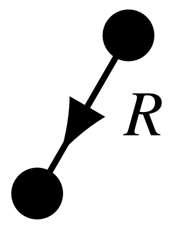
$\}$, $\ldots$

Running example: **planar rooted binary trees (PRBTs)**

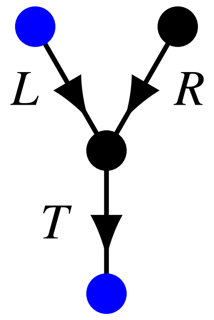
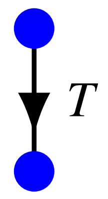
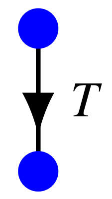
Notational convention:

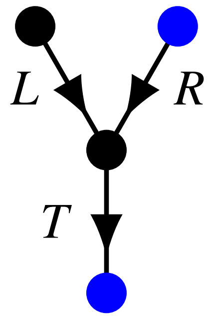
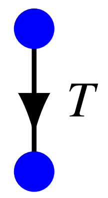
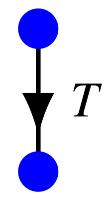


Running example: **planar rooted binary trees (PRBTs)**

Notational convention: $\vert \equiv$  , $\backslash \equiv$  $/ \equiv$ 

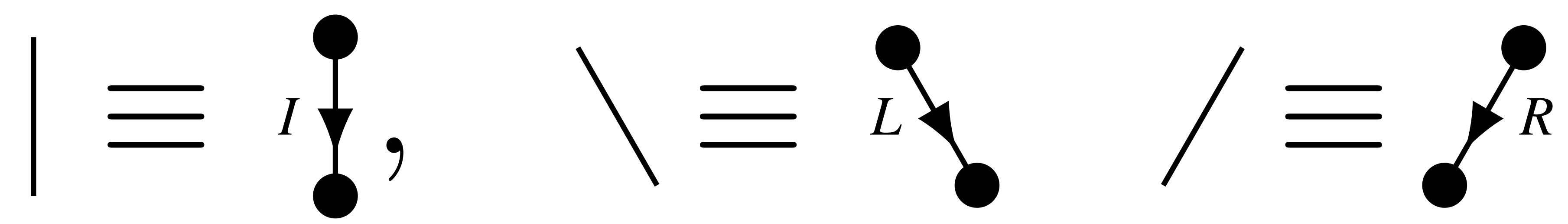
The Rémy uniform generator in the rule-algebra formalism

$\hat{G} := \hat{G}_L + \hat{G}_R\,,\ \hat{G}_L :=$ $\sum_{T\in\{I,L,R\}} \rho\Big(\delta\Big($  $\hookleftarrow$  $\hookrightarrow$  $;\ \mathsf{Shift}\big(\varnothing \hookrightarrow$  $, \mathsf{c}_{PBRT}\big)\Big)\Big)$

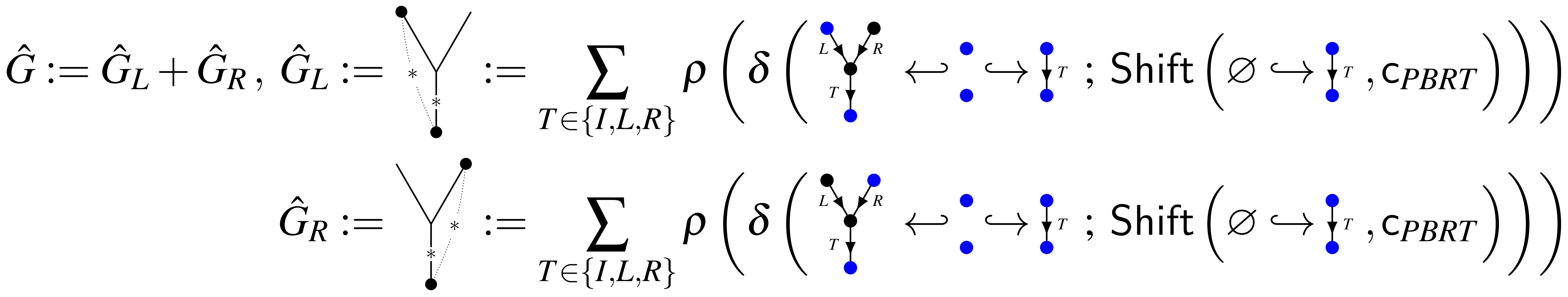
$\hat{G}_R :=$ $\sum_{T\in\{I,L,R\}} \rho\Big(\delta\Big($  $\hookleftarrow$  $\hookrightarrow$  $;\ \mathsf{Shift}\big(\varnothing \hookrightarrow$  $, \mathsf{c}_{PBRT}\big)\Big)\Big)$

Running example: **planar rooted binary trees (PRBTs)**

Notational convention:



The Rémy uniform generator in the rule-algebra formalism



Running example: **planar rooted binary trees (PRBTs)**

Notational convention: $\vert \equiv \bullet \xrightarrow{I} \bullet, \quad \backslash \equiv \bullet \xrightarrow{L} \bullet \quad / \equiv \bullet \xrightarrow{R} \bullet$

The Rémy uniform generator in the rule-algebra formalism

$$\hat{G} := \hat{G}_L + \hat{G}_R\,,\ \hat{G}_L := \quad := \sum_{T\in\{I,L,R\}} \rho\left(\delta\left(\quad \hookleftarrow \bullet\ \bullet \hookrightarrow \bullet \xrightarrow{T} \bullet\ ;\ \mathsf{Shift}\left(\varnothing \hookrightarrow \bullet \xrightarrow{T} \bullet\,, \mathsf{c}_{PBRT}\right)\right)\right)$$

$$\hat{G}_R := \quad := \sum_{T\in\{I,L,R\}} \rho\left(\delta\left(\quad \hookleftarrow \bullet\ \bullet \hookrightarrow \bullet \xrightarrow{T} \bullet\ ;\ \mathsf{Shift}\left(\varnothing \hookrightarrow \bullet \xrightarrow{T} \bullet\,, \mathsf{c}_{PBRT}\right)\right)\right)$$

$$\hat{G}\,|\,|\,\rangle = \sum_{t\in\mathcal{T}_1} 2!\,|t\rangle\,,\ \forall t\in\mathcal{T}_1 : \hat{G}\,|t\rangle = \sum_{t'\in\mathcal{T}_2} 3!\,\left|t'\right\rangle\,,\ \ldots\,,\ \forall t\in\mathcal{T}_n : \hat{G}\,|t\rangle = \sum_{t'\in\mathcal{T}_{n+1}} (n+2)!\,\left|t'\right\rangle$$

Counting **patterns** in combinatorial structures

Definition 5. Let $\mathbb{T} \in \{\mathsf{DPO}, \mathsf{SqPO}\}$ denote the rewriting semantics utilized. Then **pattern count observables** are defined as follows:

$$\hat{O}_{P,q;\mathsf{c}_P} := \rho_{\mathbf{C}}^{DPO}\left(\delta\left(P \xleftarrow{q} Q \xrightarrow{q} P; \mathsf{c}_P\right)\right)\,, \qquad \hat{O}_{P;\mathsf{c}_P} := \rho_{\mathbf{C}}^{SqPO}\left(\delta\left(P \xleftarrow{id_P} P \xrightarrow{id_P} P; \mathsf{c}_P\right)\right) \tag{9}$$

Counting **patterns** in combinatorial structures

Definition 5. Let $\mathbb{T} \in \{\mathsf{DPO}, \mathsf{SqPO}\}$ denote the rewriting semantics utilized. Then **pattern count observables** are defined as follows:

$$\hat{O}_{P,q;c_P} := \rho_{\mathbf{C}}^{DPO}\left(\delta\left(P \overset{q}{\leftarrow} Q \overset{q}{\rightarrow} P; c_P\right)\right)\,, \qquad \hat{O}_{P;c_P} := \rho_{\mathbf{C}}^{SqPO}\left(\delta\left(P \xleftarrow{id_P} P \xrightarrow{id_P} P; c_P\right)\right) \tag{9}$$

To better understand the meaning of the above definitions, it is important to note the so-called **jump-closure properties** of DPO- and SqPO-types, respectively:

$$\begin{aligned}
&\forall \mathsf{R} = \left(\mathsf{O} \overset{\mathsf{o}}{\leftarrow} \mathsf{K} \overset{\mathsf{i}}{\rightarrow} \mathsf{I}, \mathsf{c_I}\right) \in \overline{\mathsf{Lin}}(\mathbf{C}) : \quad \langle| \rho_{\mathbf{C}}^{\mathbb{T}}(\delta(\mathsf{R})) = \langle| \widehat{\mathbb{O}}_{\mathbb{T}}(\delta(\mathsf{R}))\\
&\widehat{\mathbb{O}}_{\mathsf{DPO}}(\delta(\mathsf{R})) := \widehat{\mathsf{O}}_{\mathsf{I},\mathsf{k};\mathsf{c_I}}\,, \qquad \widehat{\mathbb{O}}_{\mathsf{SqPO}}(\delta(\mathsf{R})) := \widehat{\mathsf{O}}_{\mathsf{I};\mathsf{c_I}}\,, \qquad \langle| : \widehat{\mathbf{C}} \rightarrow \mathbb{R} : \ |\mathsf{X}\rangle \mapsto \langle| \mathsf{X}\rangle := 1_{\mathbb{R}}\,.
\end{aligned} \tag{10}$$

Counting **patterns** in combinatorial structures

Definition 5. Let $\mathbb{T} \in \{\mathsf{DPO}, \mathsf{SqPO}\}$ denote the rewriting semantics utilized. Then **pattern count observables** are defined as follows:

$$\hat{O}_{P,q;c_P} := \rho_{\mathbf{C}}^{DPO}\left(\delta\left(P \xleftarrow{q} Q \xrightarrow{q} P; c_P\right)\right), \qquad \hat{O}_{P;c_P} := \rho_{\mathbf{C}}^{SqPO}\left(\delta\left(P \xleftarrow{id_P} P \xrightarrow{id_P} P; c_P\right)\right) \tag{9}$$

To better understand the meaning of the above definitions, it is important to note the so-called **jump-closure properties** of DPO- and SqPO-types, respectively:

$$\begin{gathered}\forall R = \left(O \xleftarrow{o} K \xrightarrow{i} I, c_I\right) \in \overline{\mathsf{Lin}}(\mathbf{C}): \quad \langle| \rho_{\mathbf{C}}^{\mathbb{T}}(\delta(R)) = \langle| \widehat{\mathbb{O}}_{\mathbb{T}}(\delta(R)) \\ \widehat{\mathbb{O}}_{\mathsf{DPO}}(\delta(R)) := \widehat{O}_{I,k;c_I}, \qquad \widehat{\mathbb{O}}_{\mathsf{SqPO}}(\delta(R)) := \widehat{O}_{I;c_I}, \qquad \langle| : \widehat{\mathbf{C}} \to \mathbb{R} : \ |X\rangle \mapsto \langle|X\rangle := 1_{\mathbb{R}}.\end{gathered} \tag{10}$$

In other words, $\widehat{O}_{I,k;c_I}$ and $\widehat{O}_{I;c_I}$ permit to **count** the number of matches of the rewriting rule $R = (O \xleftarrow{o} K \xrightarrow{i} I, c_I)$ in DPO- and SqPO-semantics, respectively. More explicitly, we find that

$$\langle| \hat{O}_{P,q;c_I} |X\rangle = |\mathcal{M}^{DPO}_{(P \xleftarrow{q} Q \xrightarrow{q} P; c_I)}(X)|, \quad \langle| \hat{O}_{P;c_I} |X\rangle = |\mathcal{M}^{SqPO}_{(P \xleftarrow{id_P} P \xrightarrow{id_P} P; c_I)}(X)|. \tag{11}$$

Counting **patterns** in combinatorial structures

Example 1. The simplest type of observables encountered in practice are the **"plain" pattern-counting observables** $\widehat{O}_{P} = \widehat{O}_{P,id_P;\mathsf{true}} = \widehat{O}_{P;\mathsf{true}}$, with typical examples including

Counting **patterns** in combinatorial structures

Example 1. The simplest type of observables encountered in practice are the **"plain" pattern-counting observables** $\hat{O}_P = \hat{O}_{P,id_P;\mathsf{true}} = \hat{O}_{P;\mathsf{true}}$, with typical examples including

- $\hat{O}_\bullet$ (counting **vertices**),

Counting **patterns** in combinatorial structures

Example 1. The simplest type of observables encountered in practice are the **"plain" pattern-counting observables** $\hat{O}_P = \hat{O}_{P,id_P;true} = \hat{O}_{P;true}$, with typical examples including

- $\hat{O}_{\bullet}$ (counting **vertices**),
- $\hat{O}_{\bullet\,\bullet}$ (counting **pairs of vertices**), and

Counting **patterns** in combinatorial structures

Example 1. The simplest type of observables encountered in practice are the **“plain” pattern-counting observables** $\widehat{O}_{\mathsf{P}} = \widehat{O}_{\mathsf{P},\mathsf{id}_{\mathsf{P}};\mathsf{true}} = \widehat{O}_{\mathsf{P};\mathsf{true}}$, with typical examples including

- $\widehat{O}_{\bullet}$ (counting **vertices**),
- $\widehat{O}_{\bullet\,\bullet}$ (counting **pairs of vertices**), and
- $\widehat{O}_{\bullet\!-\!\bullet}$ (counting **undirected edges**).

Counting **patterns** in combinatorial structures

Example 1. The simplest type of observables encountered in practice are the **"plain" pattern-counting observables** $\hat{O}_P = \hat{O}_{P,id_P;true} = \hat{O}_{P;true}$, with typical examples including

- $\hat{O}_{\bullet}$ (counting **vertices**),
- $\hat{O}_{\bullet\,\bullet}$ (counting **pairs of vertices**), and
- $\hat{O}_{\bullet\!-\!\bullet}$ (counting **undirected edges**).

More generally, for example in DPO rewriting the variant $\hat{O}_{\varnothing\hookrightarrow\bullet,true}$ effectively counts **vertices that are not linked to any other vertices** via incident edges, while in both DPO- and SqPO-rewriting the linear operator

$$\hat{O}_{\bullet\,\bullet\hookrightarrow\bullet\,\bullet;\nexists(\bullet\,\bullet\hookrightarrow\bullet\!-\!\bullet)} = \hat{O}_{\bullet\,\bullet;\nexists(\bullet\,\bullet\hookrightarrow\bullet\!-\!\bullet)}$$

counts **pairs of vertices not linked by an edge**.

A rule-algebraic **generating-functionology**

Definition 6.

(12)

A rule-algebraic **generating-functionology**

Definition 6.

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

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

A rule-algebraic **generating-functionology**

Definition 6. 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 6. Let $\widehat{\mathsf{G}}$ be a linear operator (the **generator**), let $\widehat{\mathsf{O}}_1, \dots, \widehat{\mathsf{O}}_m$ 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 6. Let $\widehat{G}$ be a linear operator (the **generator**), let $\widehat{O}_1, \ldots, \widehat{O}_m$ 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 $\underline{\omega} \cdot \underline{\widehat{O}} := \sum_{j=1}^{m} \omega_j \widehat{O}_j$, and λ as well as $\omega_1, \ldots, \omega_m$ are **formal variables**.

A rule-algebraic **generating-functionology**

Definition 6. 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 λ as well as $\omega_1, \ldots, \omega_m$ are **formal variables**.

A rule-algebraic **generating-functionology**

Definition 6. Let $\widehat{\mathsf{G}}$ be a linear operator (the **generator**), let $\widehat{\mathsf{O}}_1, \dots, \widehat{\mathsf{O}}_m$ 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 λ as well as $\omega_1, \dots, \omega_m$ are **formal variables**.

- Since we assume X_0 to be a **finite object**, clearly each of the sets $S^{(n)}_{\hat{G}}$ is of finite cardinality.
- The coefficients $g_n = \langle| \, \widehat{G}^n \, |X_0\rangle$ are evidently of finite value as well, which in summary permits the following **repartition** of the formal power series $\mathcal{G}(\lambda; \underline{0})$:

$$\mathcal{G}(\lambda; \underline{0}) = \sum_{n \geq 0} \frac{\lambda^n}{n!} \sum_{X \in S^{(n)}_{\hat{G}}} g_n(X) \,, \qquad g_n(X) := \langle X | \, \hat{G}^n \, |X_0\rangle \tag{13}$$

Consequently, the configurations $X \in S^{(n)}_{\hat{G}}$ may be seen as the **combinatorial structures contained in the n-th generation**, with $g_n(X)$ the **weight** of a configuration X in the n-th generation.

A rule-algebraic **generating-functionology**

Definition 6. 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 λ as well as $\omega_1, \ldots, \omega_m$ are **formal variables**.

- For generic values of $\underline{\omega}$, $\mathcal{G}(\lambda;\underline{\omega})$ evaluates as follows:

$$\mathcal{G}(\lambda;\underline{\omega}) = \sum_{n\geq 0}\tfrac{\lambda^n}{n!}\langle|\, e^{\underline{\omega}\cdot\underline{\hat{O}}}\hat{G}^n\,|X_0\rangle = \sum_{n\geq 0}\tfrac{\lambda^n}{n!}\sum_{X\in S^{(n)}_{\hat{G}}} g_n(X)e^{\underline{\omega}\cdot\underline{N(X)}}\,,\quad N_i(X) := \langle|\,\hat{O}_i\,|X\rangle\,. \tag{14}$$

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:

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

Combinatorial evolution equations

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

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

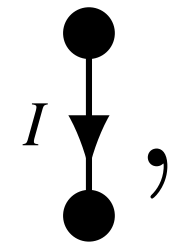
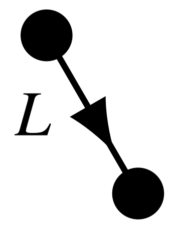
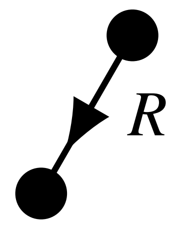
Applying the version of the **jump-closure theorem** appropriate for the chosen rewriting semantics (DPO or SqPO), the above formal evolution equation may be converted into a proper **evolution equation on formal power series** if the following **polynomial jump-closure** holds:

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

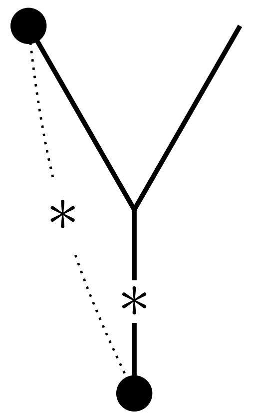
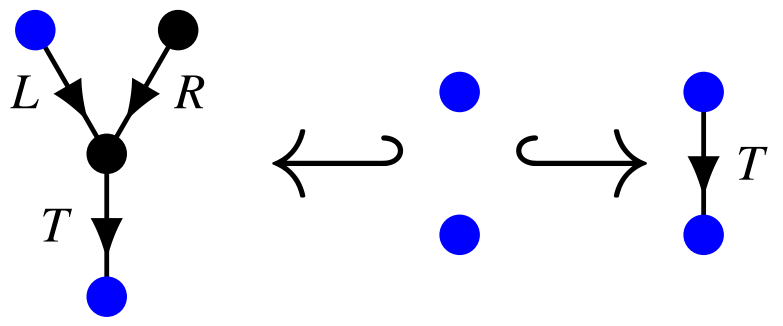
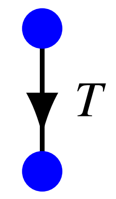
If a given set of observables satisfies (PJC'), the **formal evolution equation** (12) for the EMGF $\mathcal{G}(\lambda;\underline{\omega})$ may be refined into

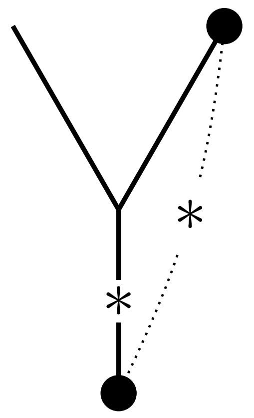
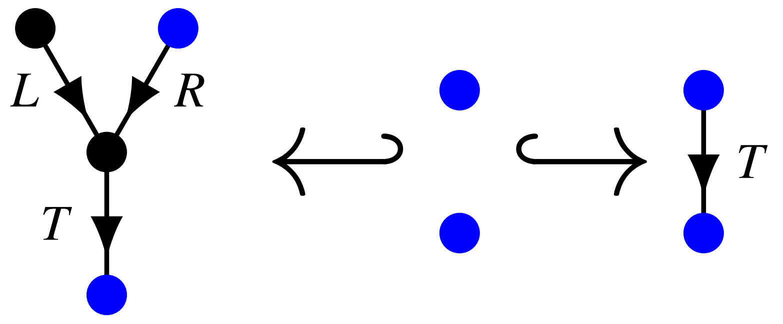
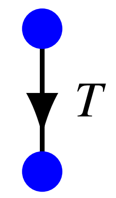
$$\tfrac{\partial}{\partial\lambda}\mathcal{G}(\lambda;\underline{\omega}) = \mathbb{G}(\underline{\omega},\underline{\partial\omega})\mathcal{G}(\lambda;\underline{\omega}), \quad \mathbb{G}(\underline{\omega},\underline{\partial\omega}) = \left(\langle| \, e^{ad_{\underline{\omega}\cdot\underline{\hat{O}}}}(\hat{G})\right)\Big|_{\underline{\hat{O}}\mapsto\unde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

Running example: **planar rooted binary trees (PRBTs)**

Notational convention: $|\equiv$  , $\backslash \equiv$  $/ \equiv$ 

The Rémy uniform generator in the rule-algebra formalism

$\hat{G} := \hat{G}_L + \hat{G}_R\,,\ \hat{G}_L :=$  $:= \sum_{T\in\{I,L,R\}} \rho\Big(\delta\Big($  $;\ \mathsf{Shift}\big(\varnothing \hookrightarrow$  $, \mathsf{c}_{PBRT}\big)\Big)\Big)$

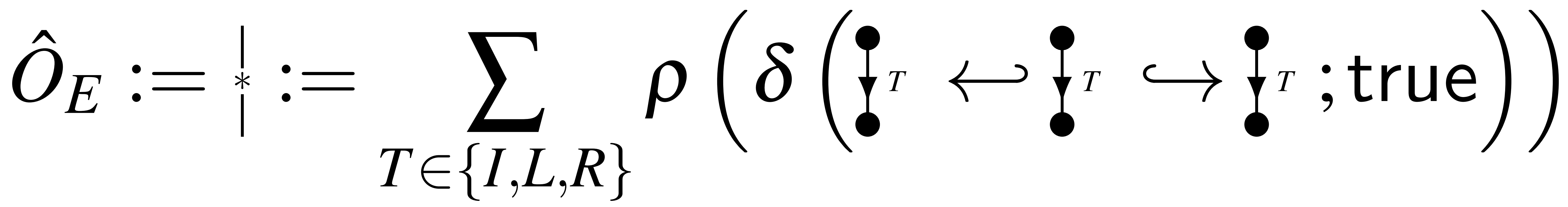
$\hat{G}_R :=$  $:= \sum_{T\in\{I,L,R\}} \rho\Big(\delta\Big($  $;\ \mathsf{Shift}\big(\varnothing \hookrightarrow$  $, \mathsf{c}_{PBRT}\big)\Big)\Big)$

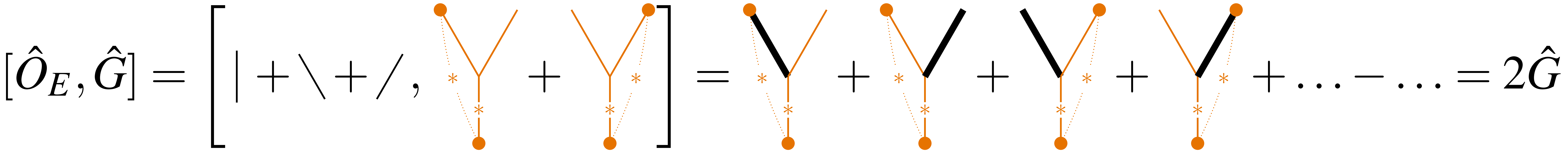
$$\hat{G}\,|\,|\,\rangle = \sum_{t\in\mathscr{T}_1} 2!\,|t\rangle\,,\ \forall t\in\mathscr{T}_1 : \hat{G}\,|t\rangle = \sum_{t'\in\mathscr{T}_2} 3!\,|t'\rangle\,,\ \ldots\,,\ \forall t\in\mathscr{T}_n : \hat{G}\,|t\rangle = \sum_{t'\in\mathscr{T}_{n+1}} (n+2)!\,|t'\rangle$$

Running example: **planar rooted binary trees (PRBTs)**

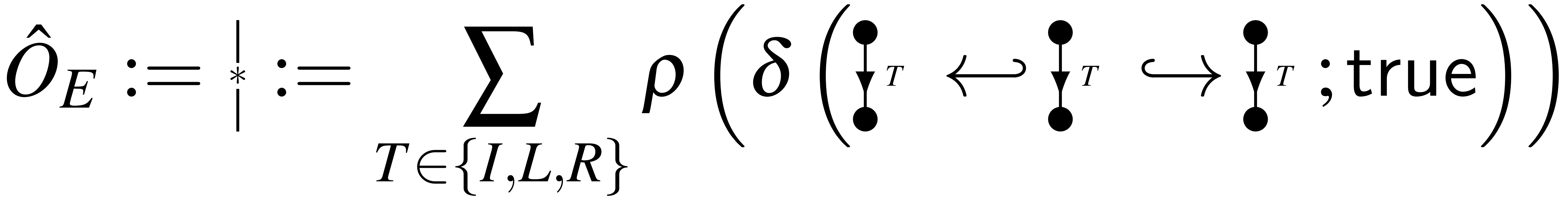
$$\hat{O}_E := \overset{|}{\underset{|}{*}} := \sum_{T\in\{I,L,R\}} \rho\left(\delta\left(\begin{smallmatrix}\bullet\\ \downarrow\\ \bullet\end{smallmatrix}{}_{T} \hookleftarrow \begin{smallmatrix}\bullet\\ \downarrow\\ \bullet\end{smallmatrix}{}_{T} \hookrightarrow \begin{smallmatrix}\bullet\\ \downarrow\\ \bullet\end{smallmatrix}{}_{T}\,;\mathsf{true}\right)\right)$$

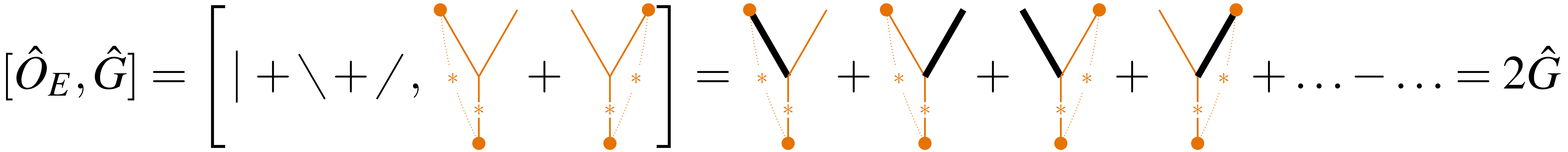
Running example: **planar rooted binary trees (PRBTs)**





Running example: **planar rooted binary trees (PRBTs)**

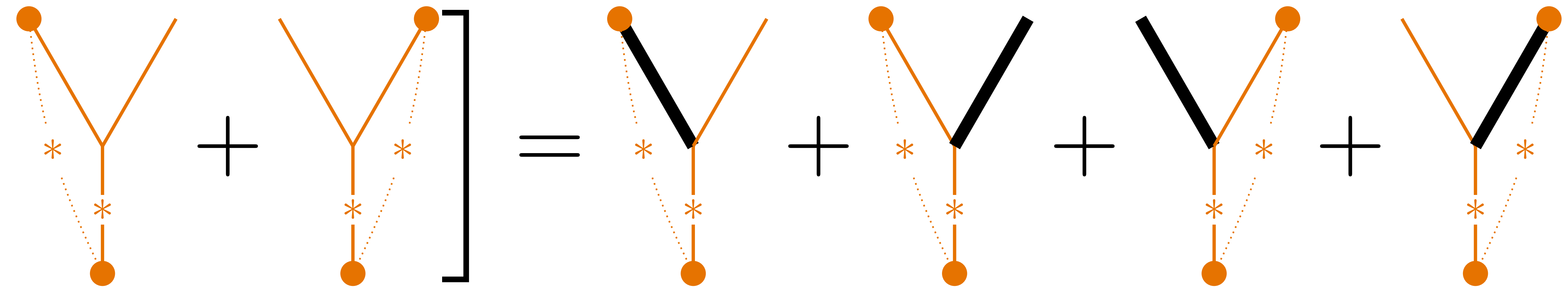




$$\langle | \hat{G} = 2 \langle | \hat{O}_E$$

Running example: **planar rooted binary trees (PRBTs)**

$$\hat{O}_E := \overset{|}{\underset{|}{*}} := \sum_{T\in\{I,L,R\}} \rho\left(\delta\left(\bullet\!\downarrow_T\!\bullet \hookleftarrow \bullet\!\downarrow_T\!\bullet \hookrightarrow \bullet\!\downarrow_T\!\bullet \,; \mathsf{true}\right)\right)$$

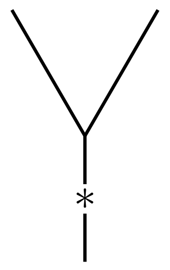


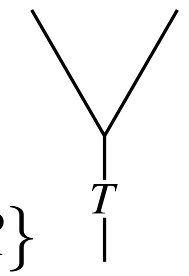
$$[\hat{O}_E, \hat{G}] = \left[\, | + \backslash + /\,, \quad + \quad \right] = \quad + \quad + \quad + \quad + \ldots - \ldots = 2\hat{G}$$

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

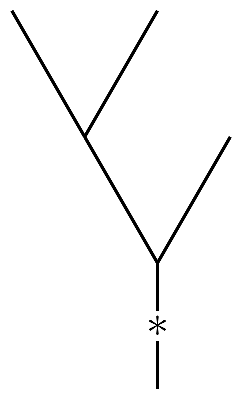
$$\left\{\begin{aligned} \frac{\partial}{\partial\lambda}\mathscr{G}(\lambda;\varepsilon) &= 2e^{2\varepsilon}\frac{\partial}{\partial\varepsilon}\mathscr{G}(\lambda;\varepsilon) \\ \mathscr{G}(0;\varepsilon) &= \langle | \, e^{\varepsilon \hat{O}_E} \, | | \rangle = e^{\varepsilon} \end{aligned}\right.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

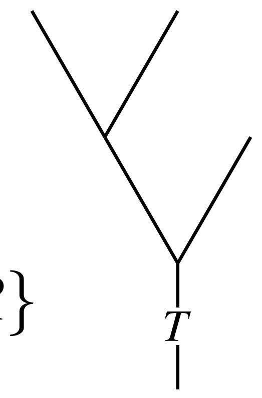
Running example: **planar rooted binary trees (PRBTs)**

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

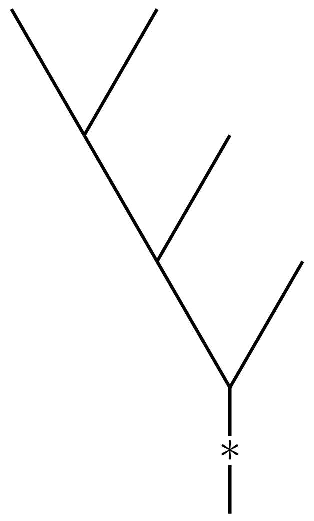
$\equiv \sum_{T \in \{I,L,R\}}$ 

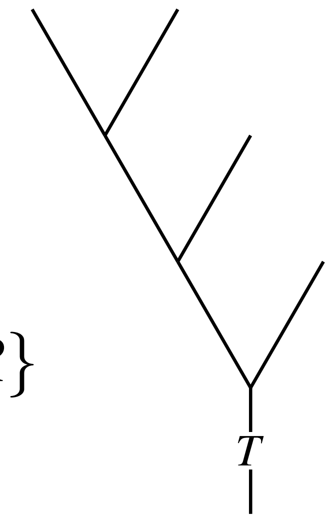
,

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

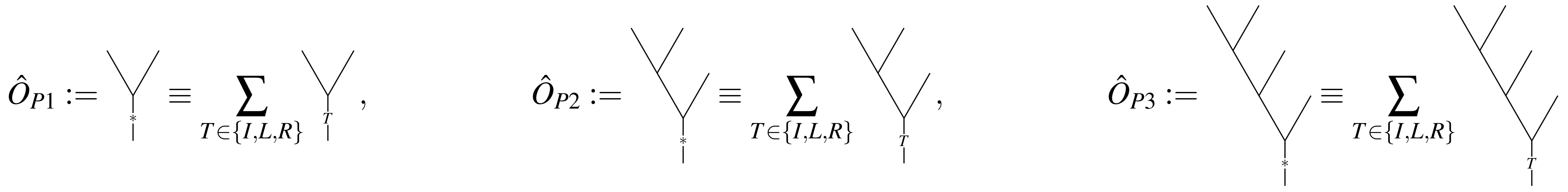
$\equiv \sum_{T \in \{I,L,R\}}$ 

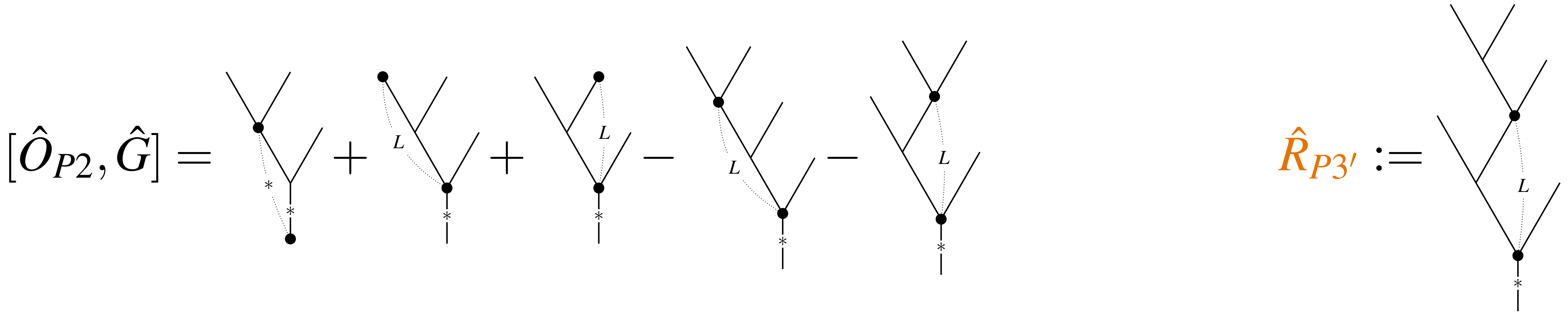
,

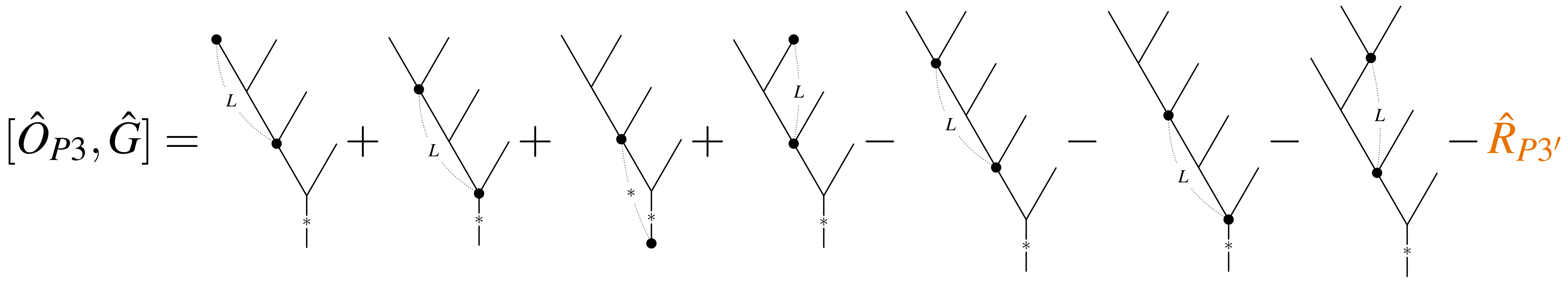
$\hat{O}_{P3} :=$ 

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

Running example: **planar rooted binary trees (PRBTs)**





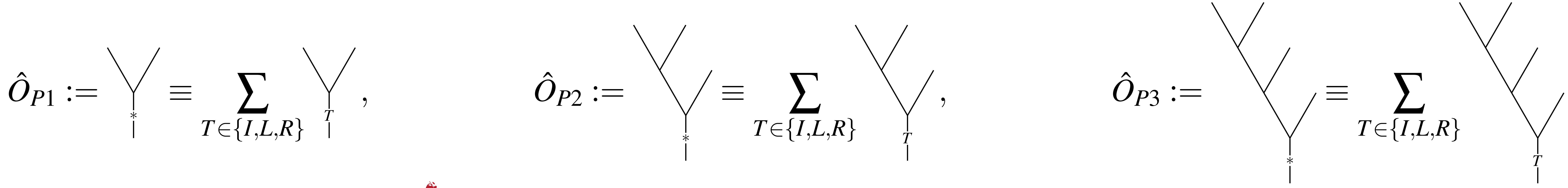


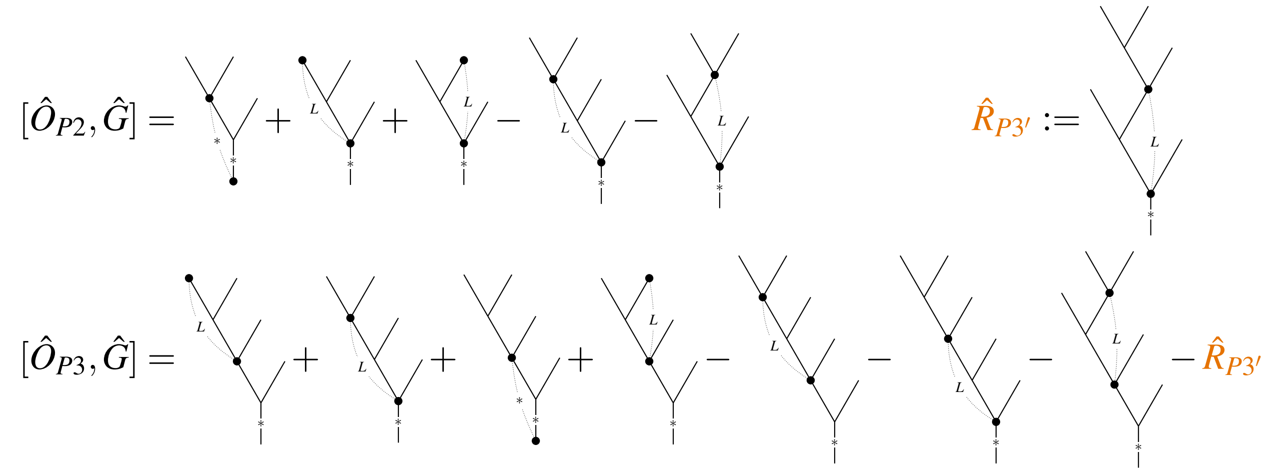
$$[\hat{O}_{P2},[\hat{O}_{P2},\hat{G}]] = [\hat{O}_{P2},\hat{G}]\,, \quad [\hat{O}_{P2},[\hat{O}_{P3},\hat{G}]] = [\hat{O}_{P3},\hat{G}] + \hat{R}_{P3}$$

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

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

Running example: **planar rooted binary trees (PRBTs)**





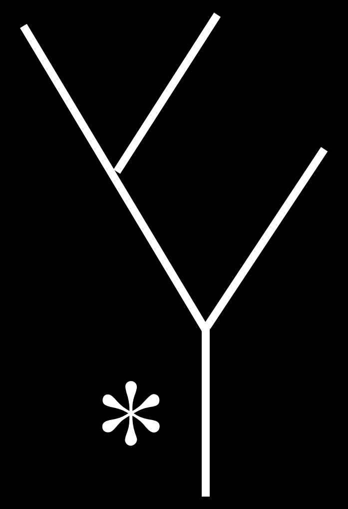
$$[\hat{O}_{P2},[\hat{O}_{P2},\hat{G}]] = [\hat{O}_{P2},\hat{G}]\,, \quad [\hat{O}_{P2},[\hat{O}_{P3},\hat{G}]] = [\hat{O}_{P3},\hat{G}] + \hat{R}_{P3}$$

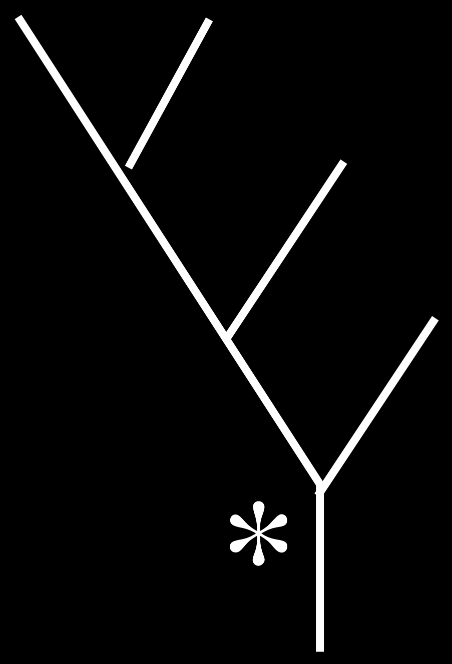
$$[\hat{O}_{P3},[\hat{O}_{P3},\hat{G}]] = [\hat{O}_{P3},\hat{G}] + 2\hat{R}_{P3'}\,, \quad [\hat{O}_{P2},\hat{R}_{P3'}] = 0\,, \quad [\hat{O}_{P3},\hat{R}_{P3'}] = -\hat{R}_{P3'}$$

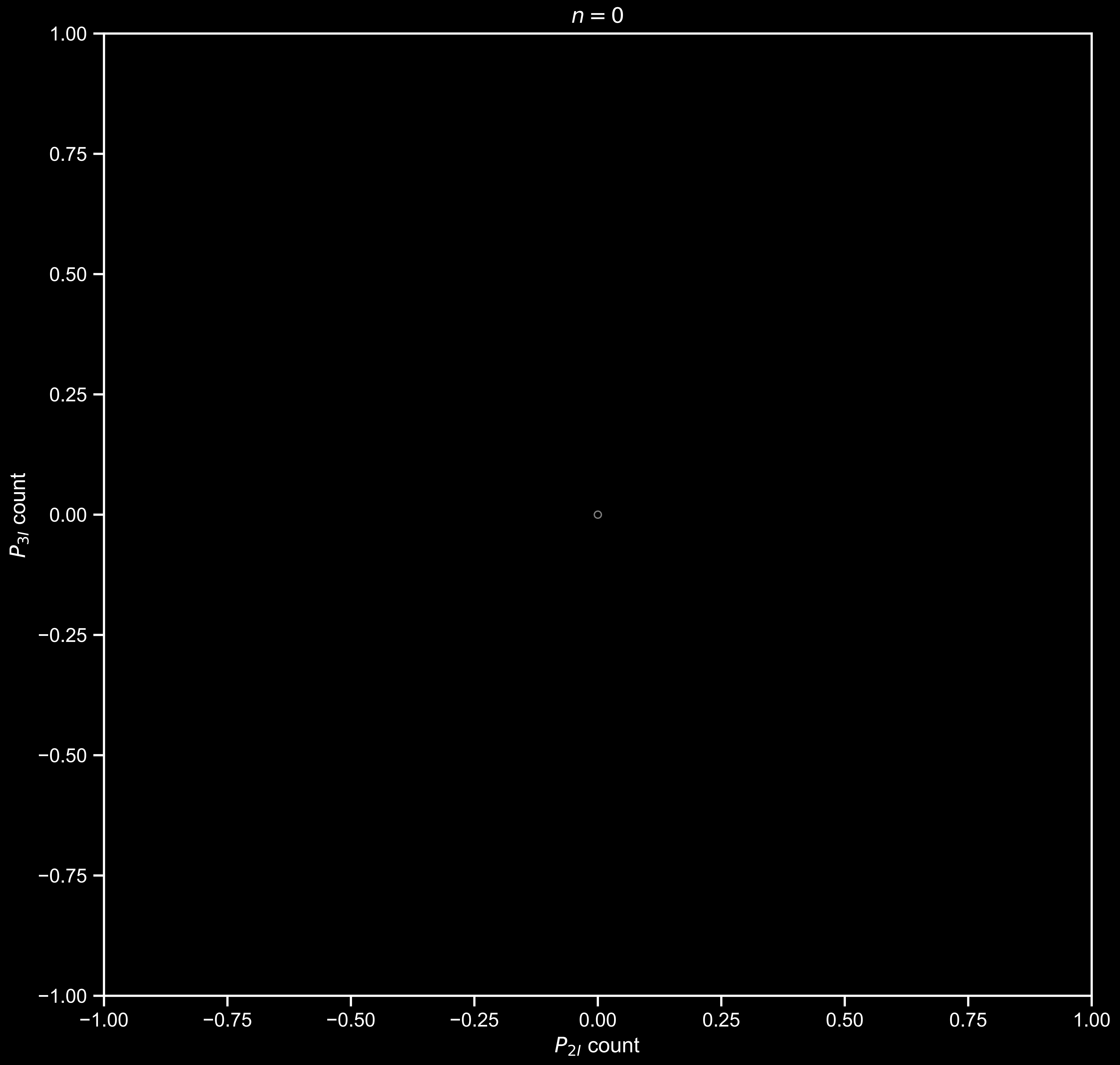
$$\langle|\,[\hat{O}_{P2},\hat{G}] = \langle|\,(3\hat{O}_{P1} - 2\hat{O}_{P2})\,, \quad \langle|\,[\hat{O}_{P3},\hat{G}] = \langle|\,(4\hat{O}_{P2} - 3\hat{O}_{P3})\,, \quad \langle|\,\hat{R}_{P3'} = \langle|\,\hat{O}_{P3}$$

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

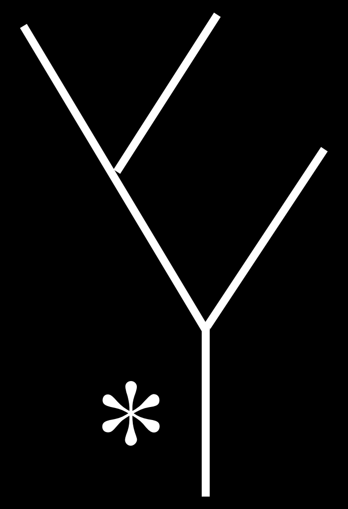
$$\begin{aligned}
\frac{\partial}{\partial\lambda}\mathscr{G}(\lambda;\underline{\omega}) &= \langle|\left(e^{ad_{\underline{\omega}\cdot\underline{\hat{O}}}}(\hat{G})\right) e^{\underline{\omega}\cdot\underline{\hat{O}}} e^{\lambda\hat{G}}\,|\rangle \overset{(*)}{=} \langle|\left(e^{ad_{\nu\hat{O}_{P3}}}\left(e^{ad_{\mu\hat{O}_{P2}}}\left(e^{ad_{\varepsilon\hat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

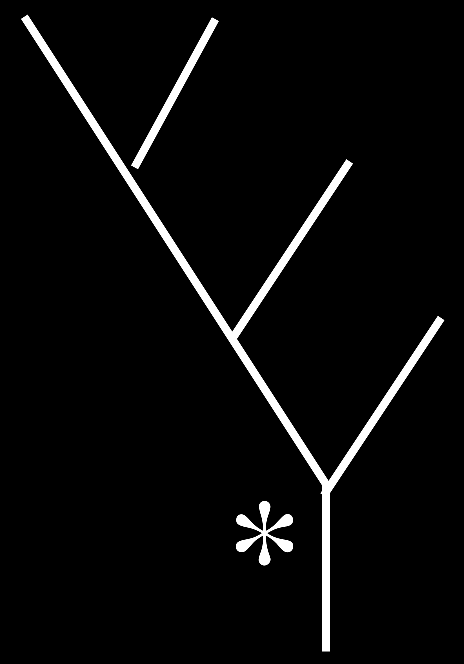
$P_{2I} :=$ 

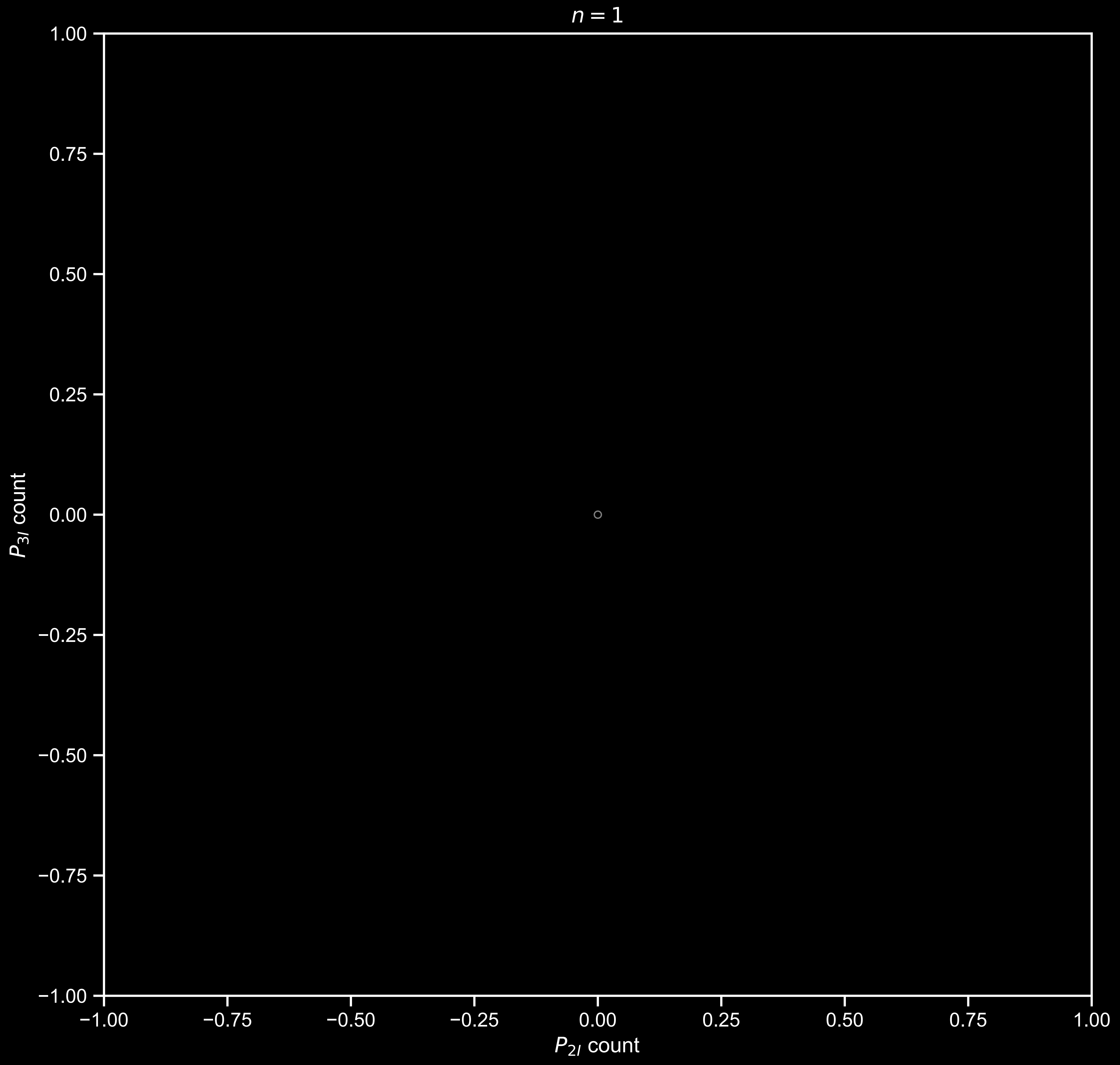
$P_{3I} :=$ 

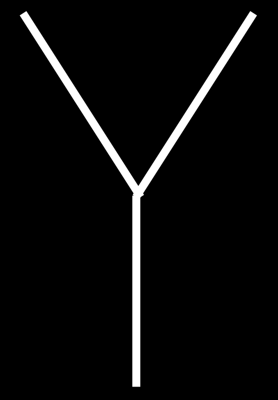


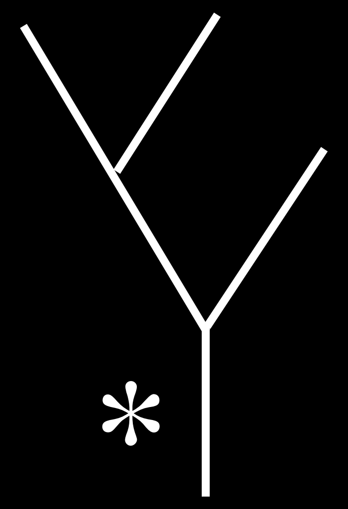
$$\mathscr{T}_0 = \{ \mid \}$$

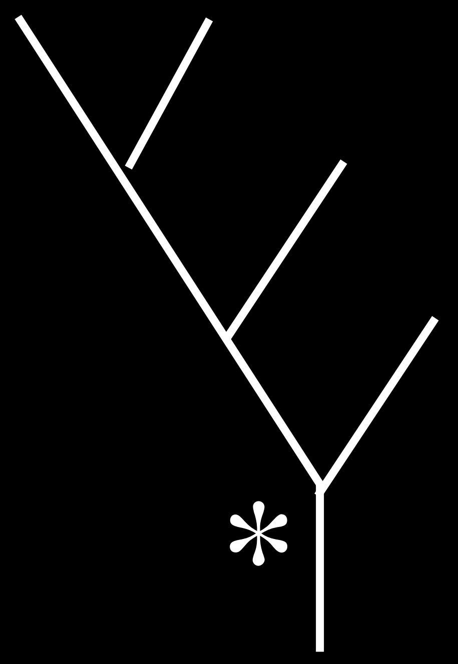
$P_{2I} :=$ 

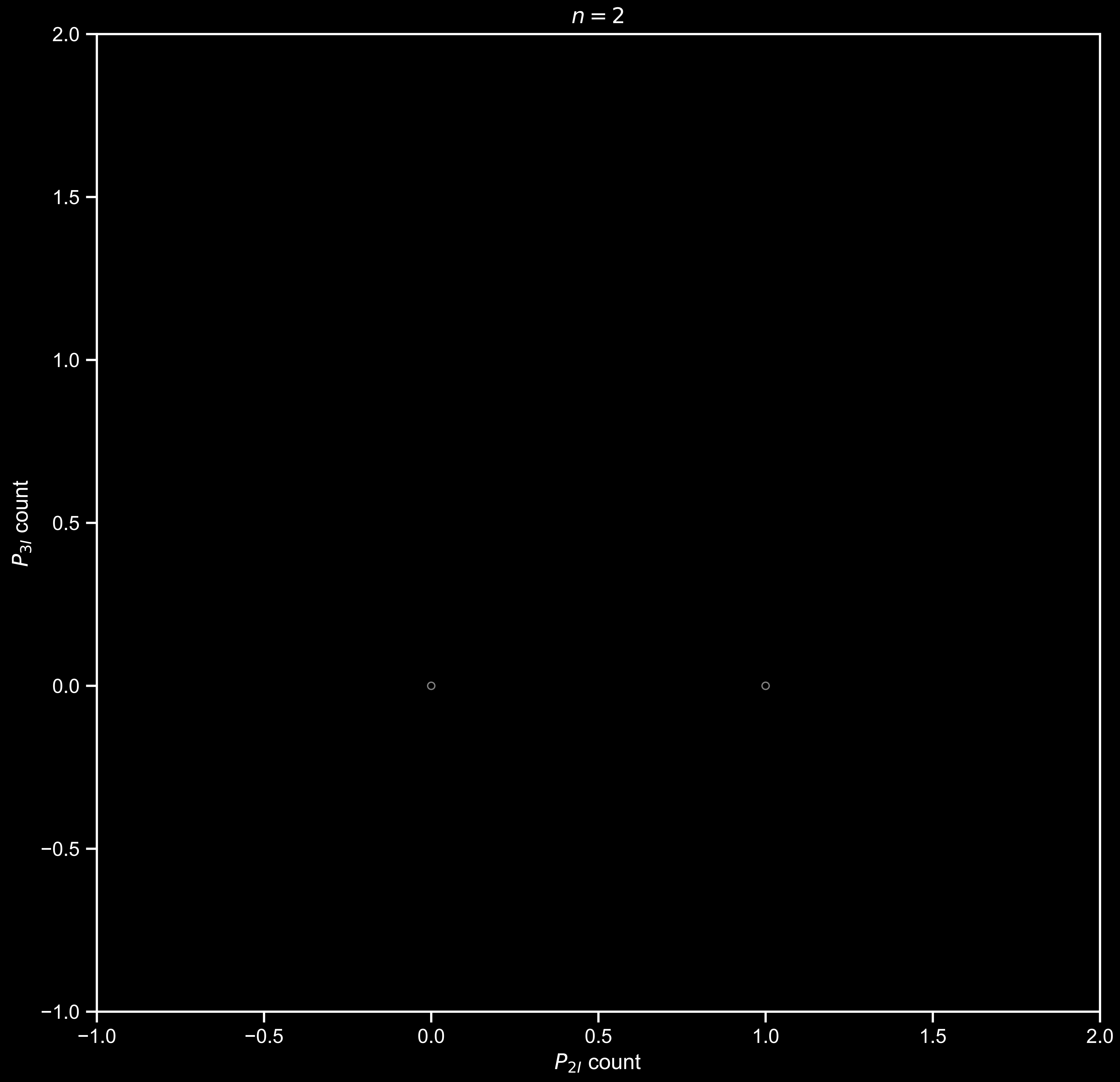
$P_{3I} :=$ 

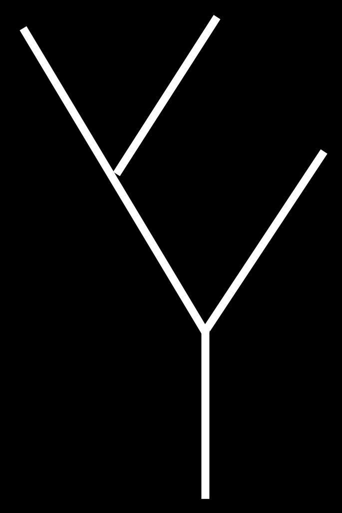


$\mathscr{T}_1 = \{$  $\}$

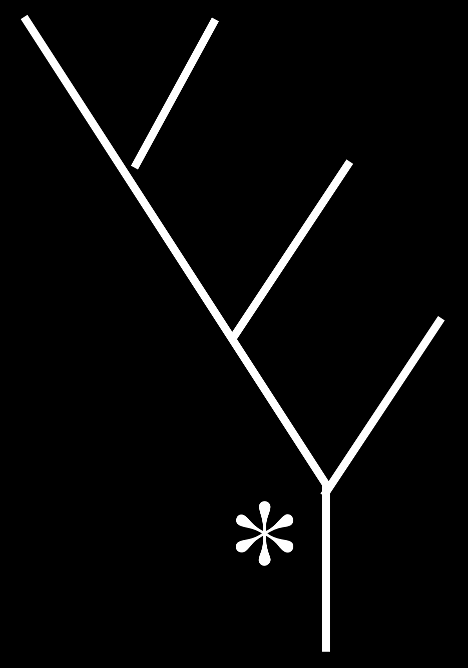
$P_{2I} :=$ 

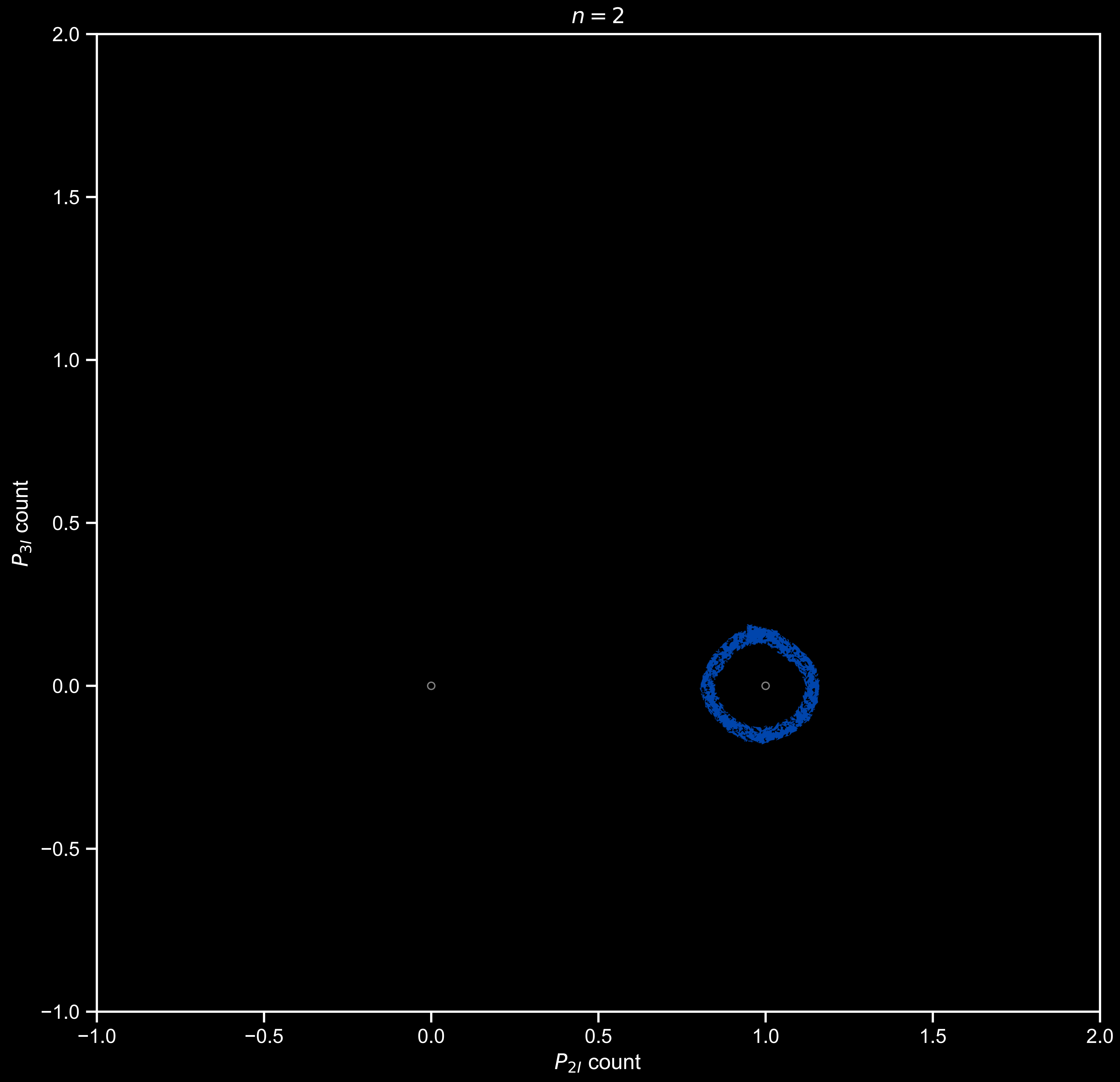
$P_{3I} :=$ 



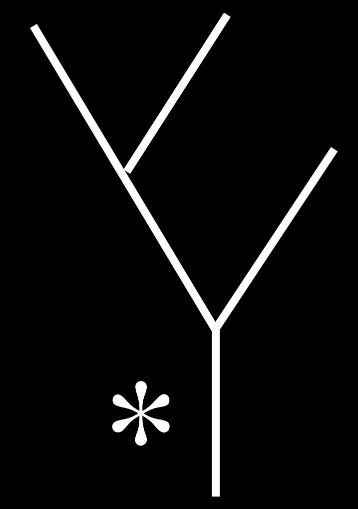
$\mathscr{T}_2 = \{$  ,  $\}$

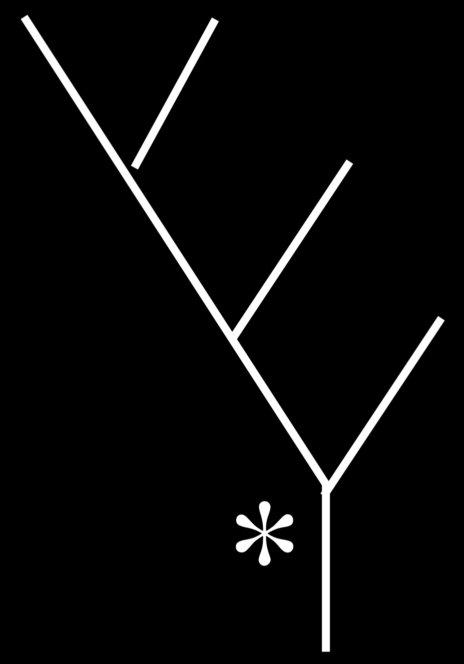
$P_{2I} :=$ 

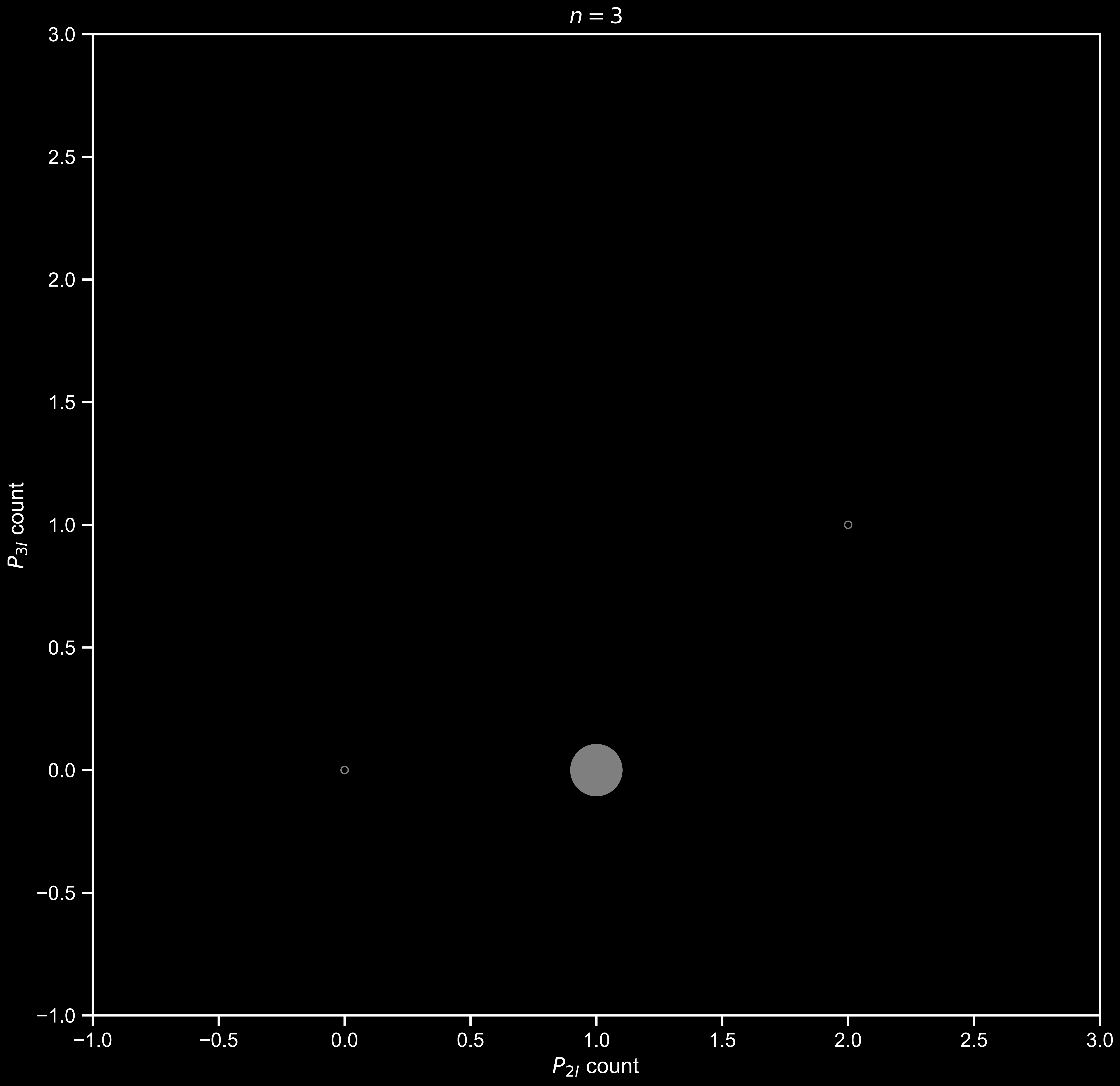
$P_{3I} :=$ 

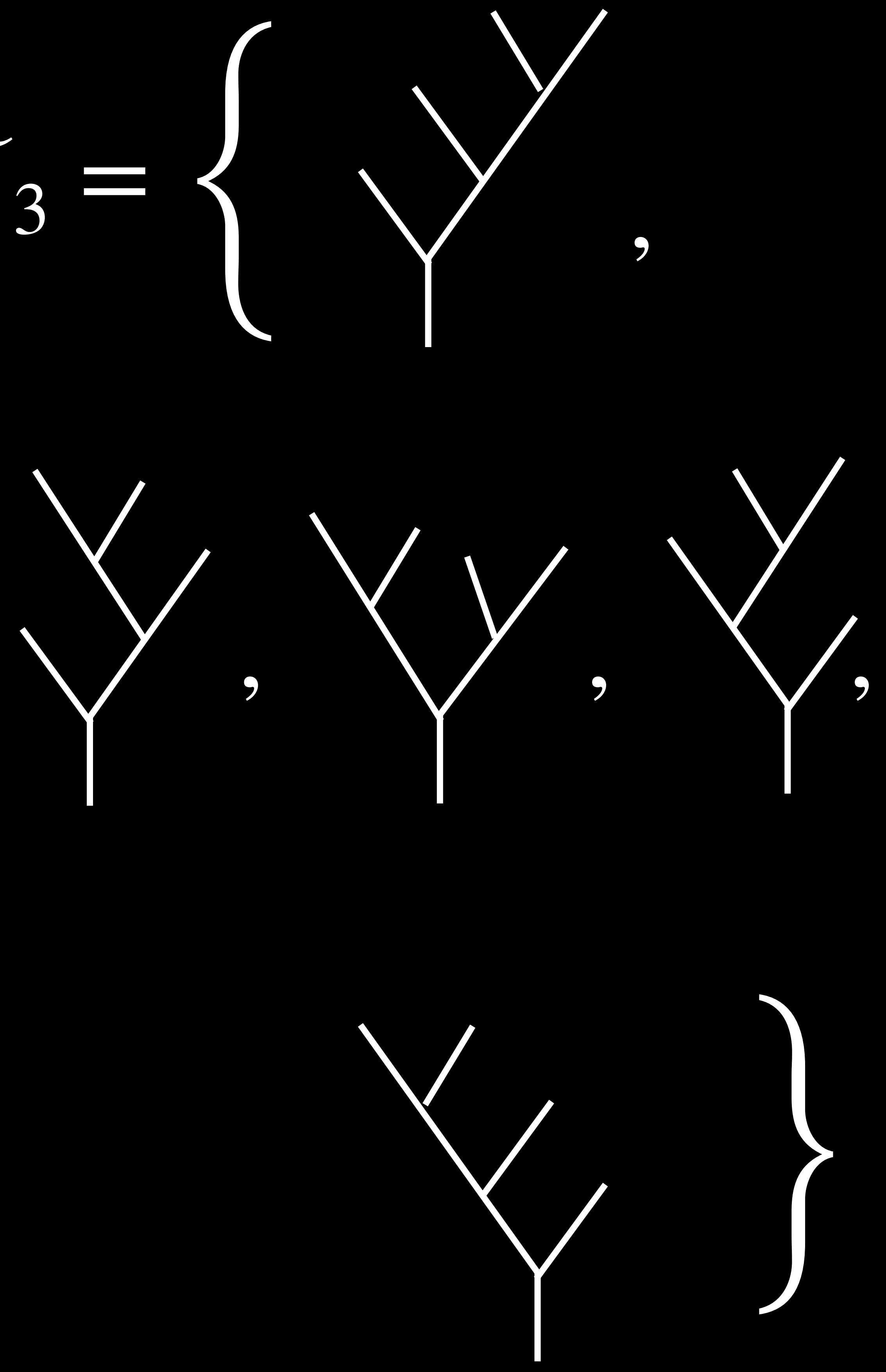


$\mathcal{T}_2 = \{$  $, $  $\}$

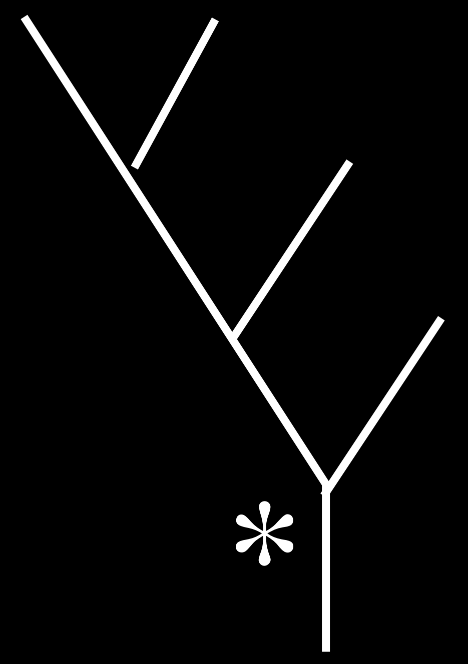
$P_{2I} :=$ 

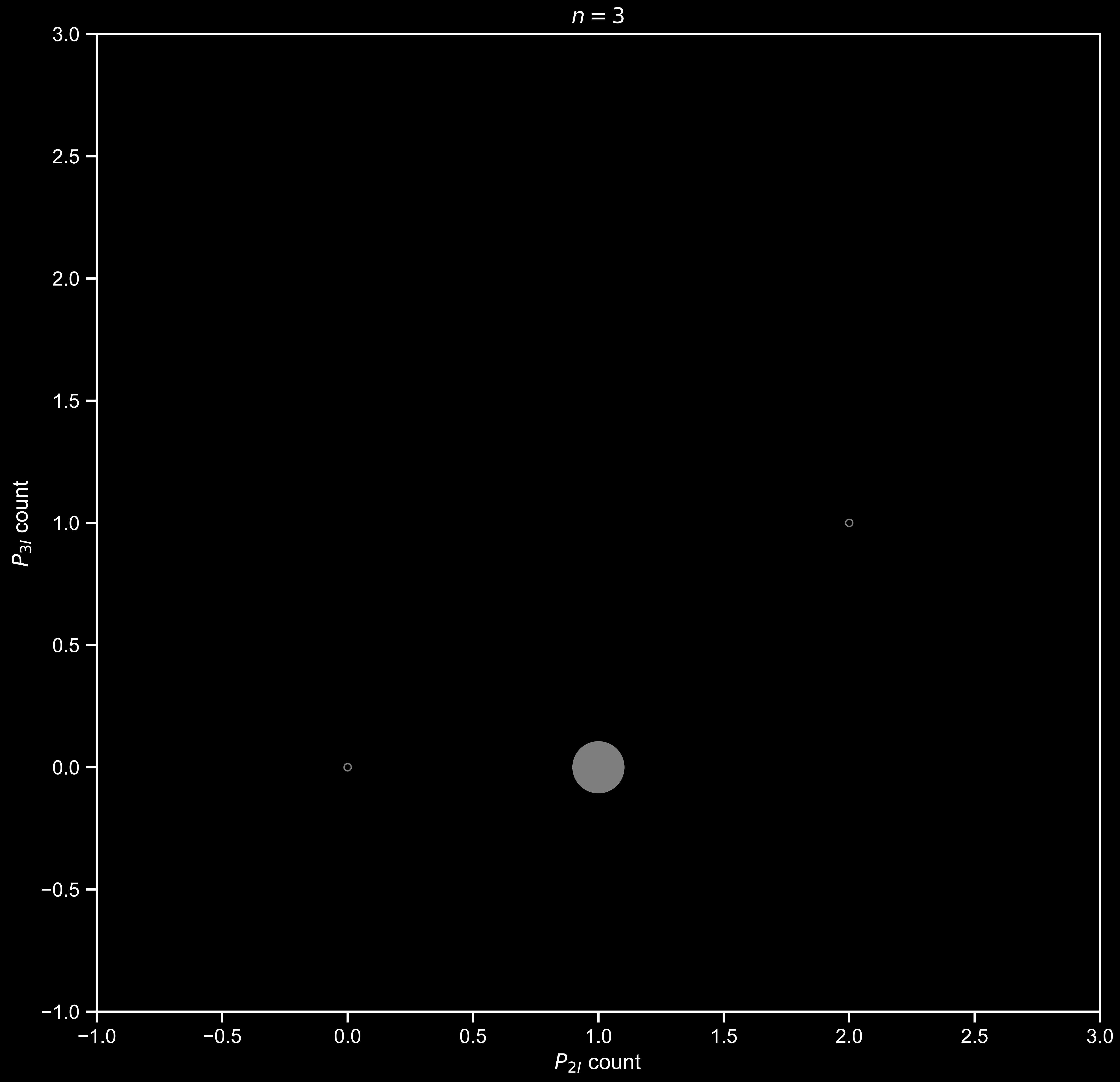
$P_{3I} :=$ 

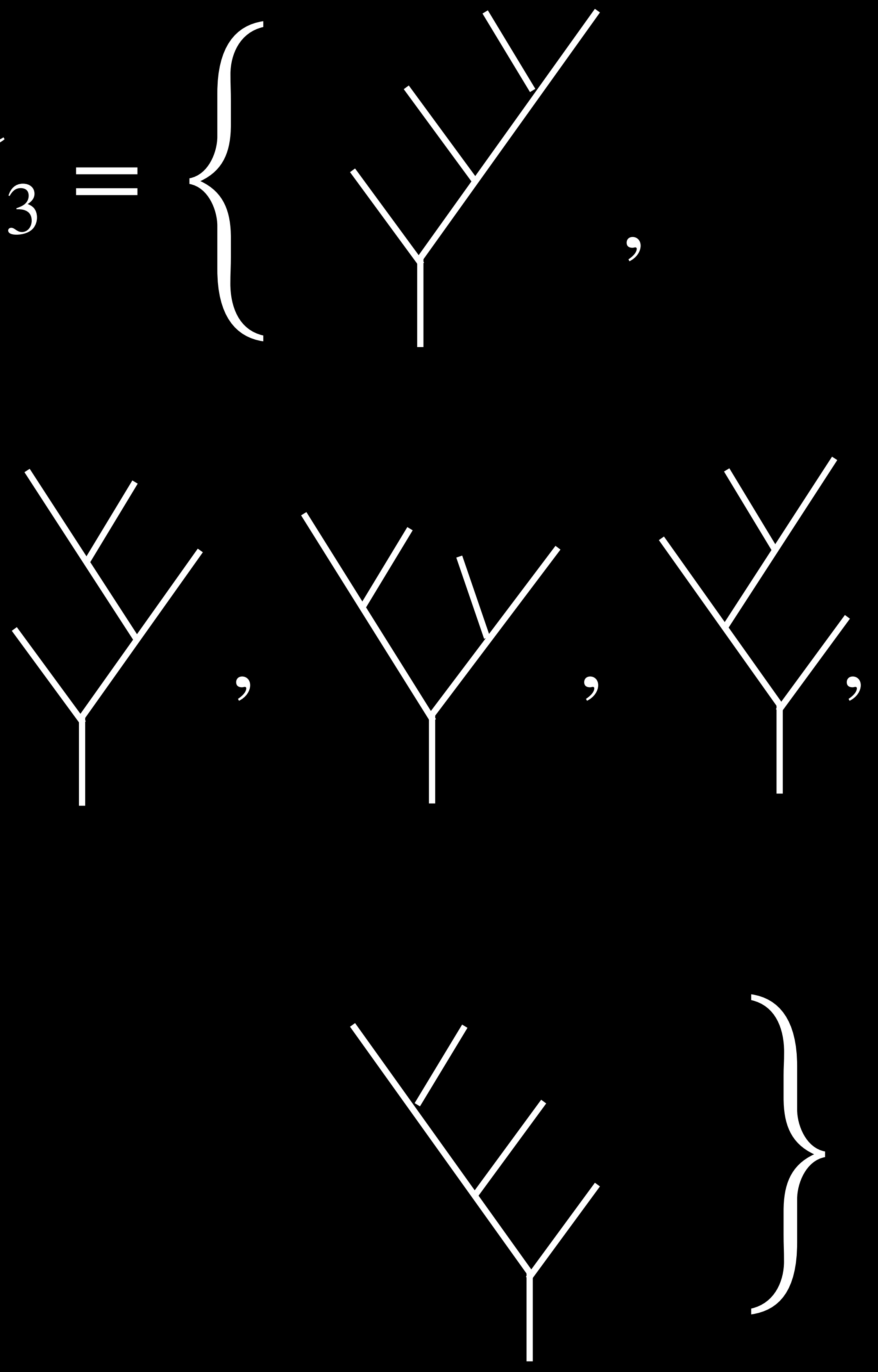


$\mathcal{T}_3 = \{$  $\}$

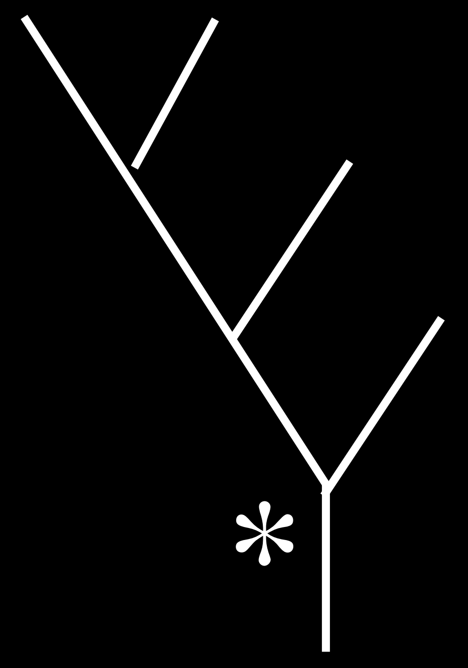
$P_{2I} :=$ 

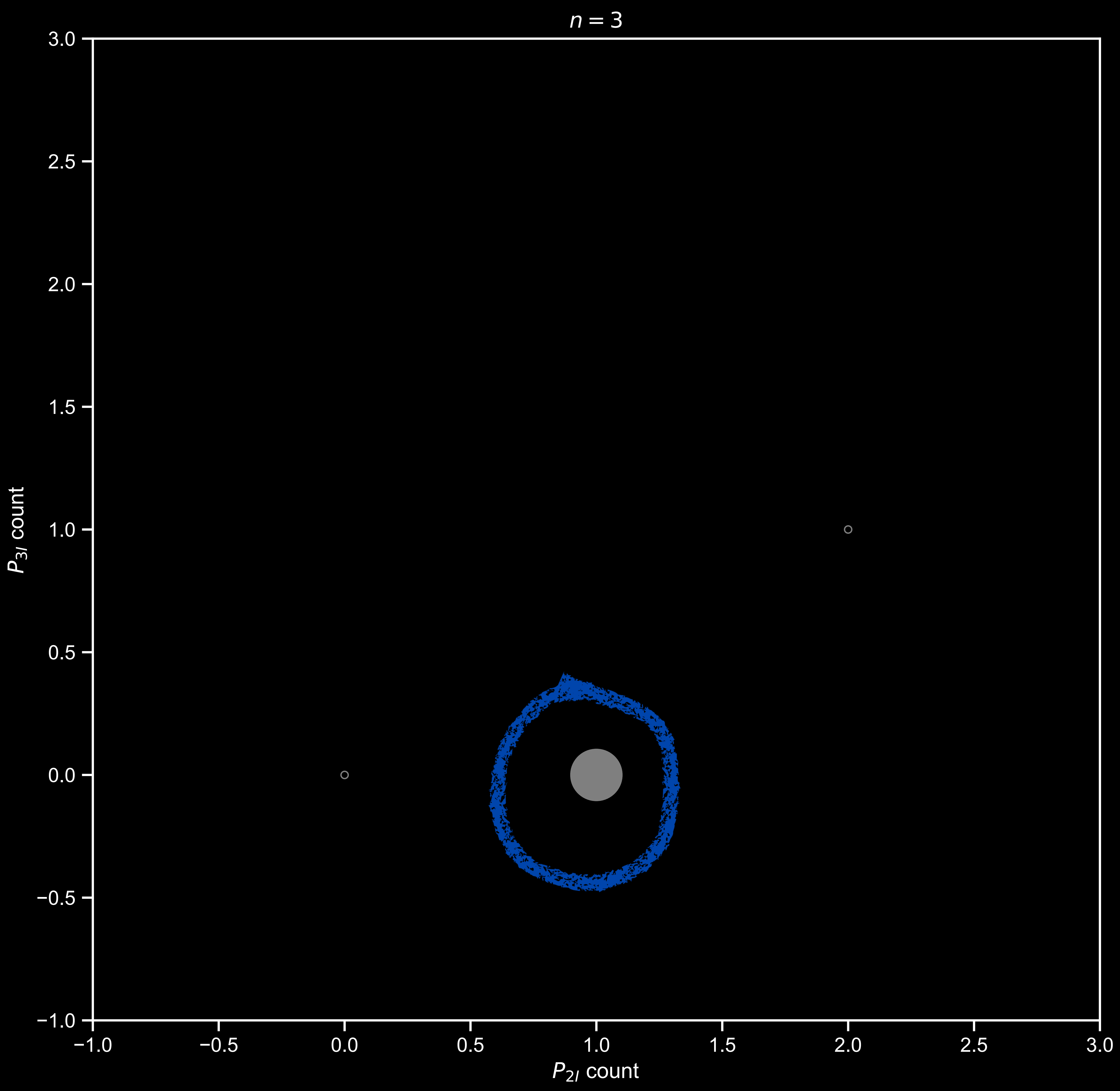
$P_{3I} :=$ 



$\mathscr{T}_3 =$ 

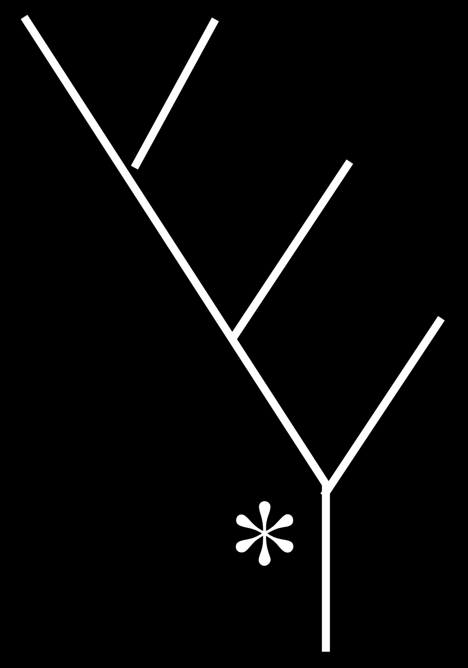
$P_{2I} :=$ 

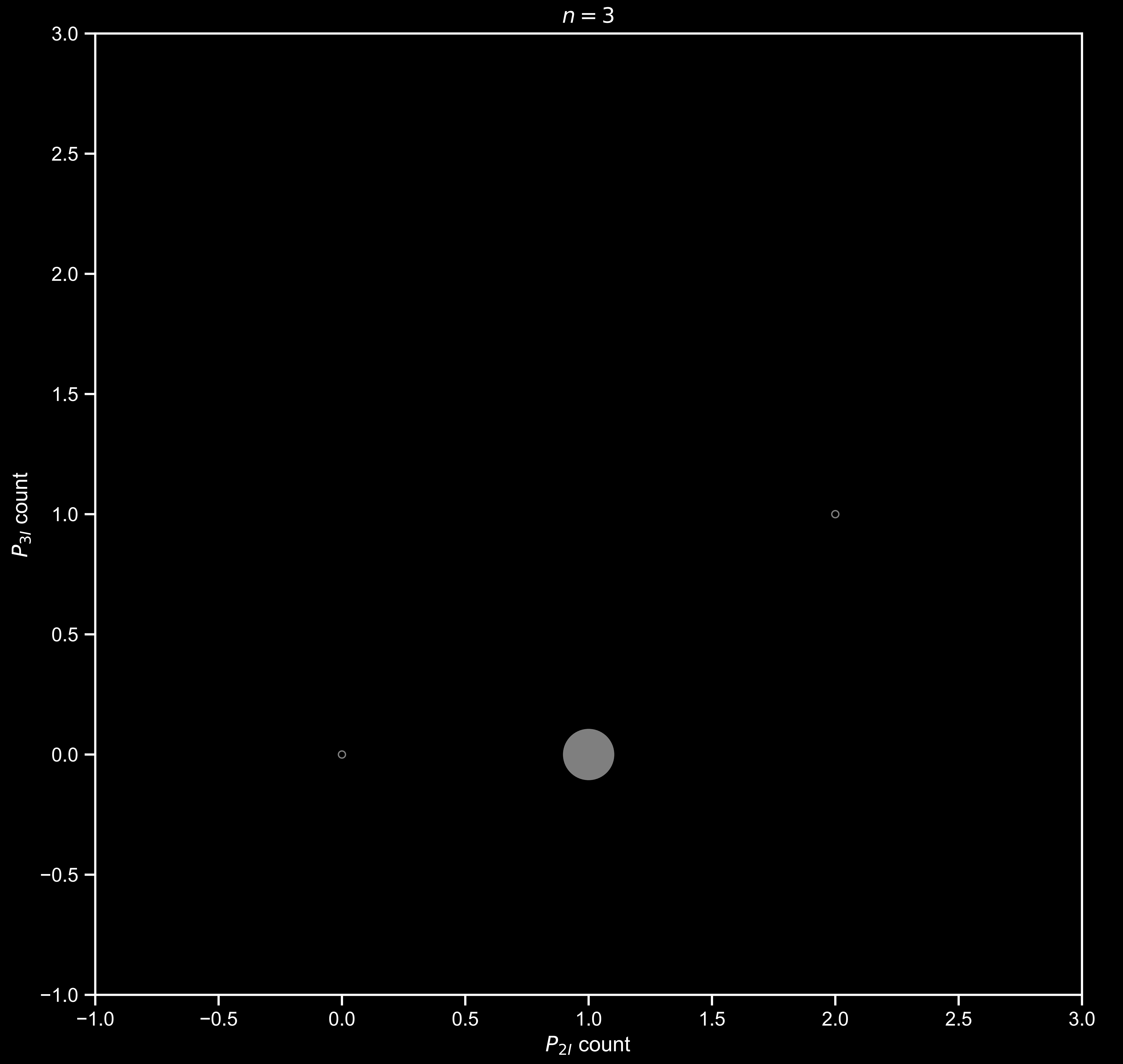
$P_{3I} :=$ 

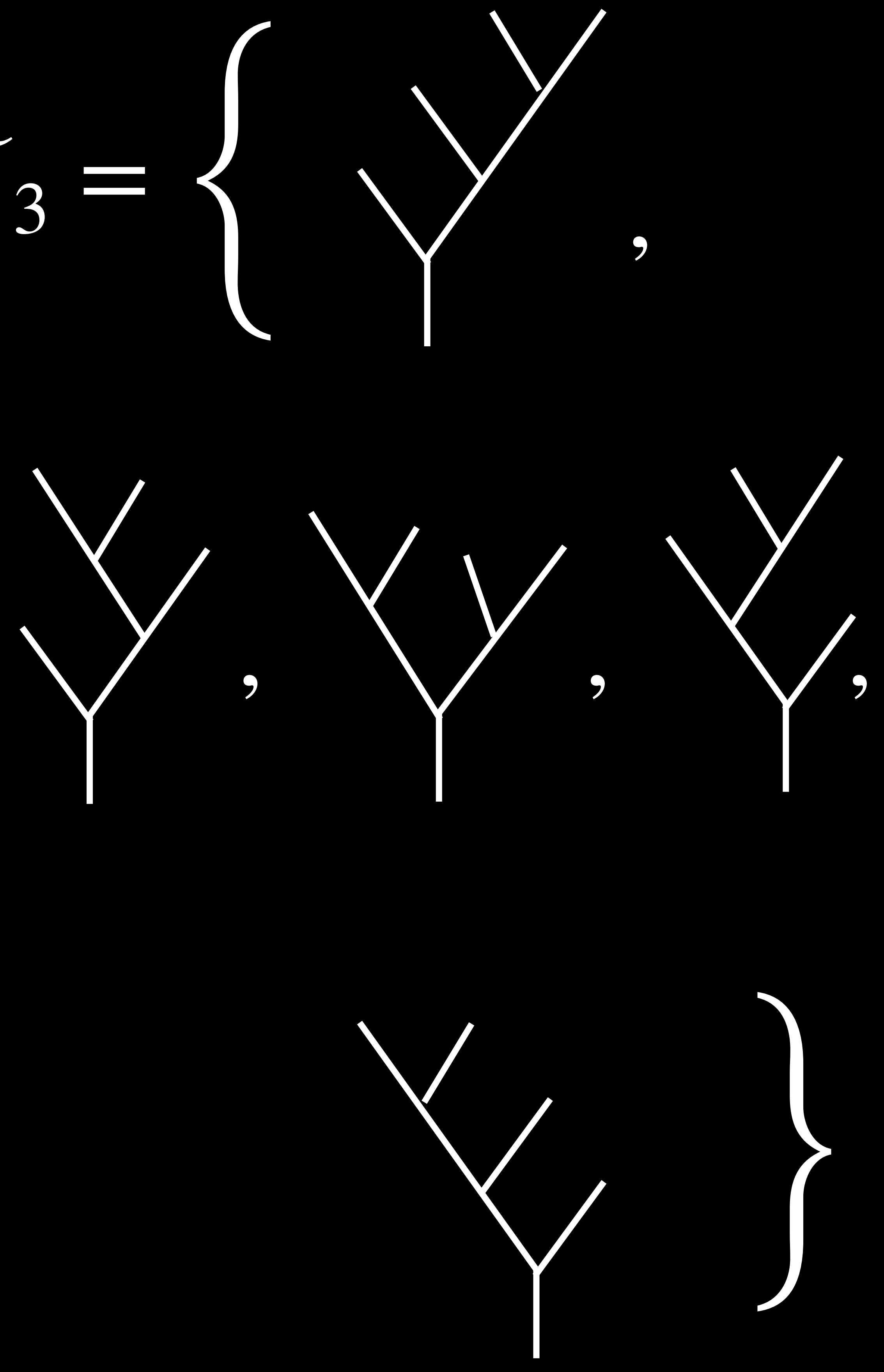


$\mathcal{T}_3 =$ 

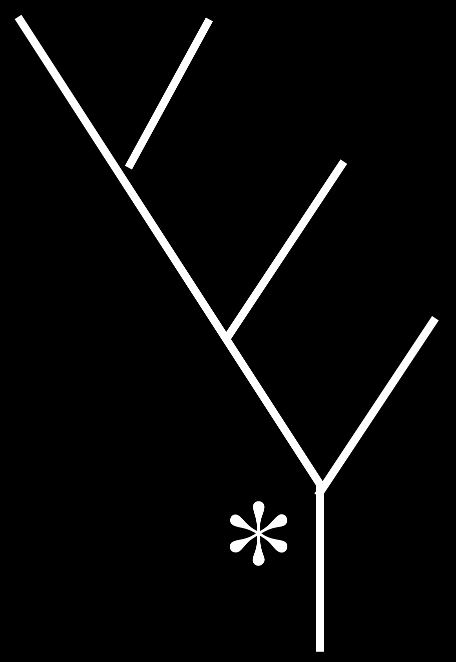
$P_{2I} :=$ 

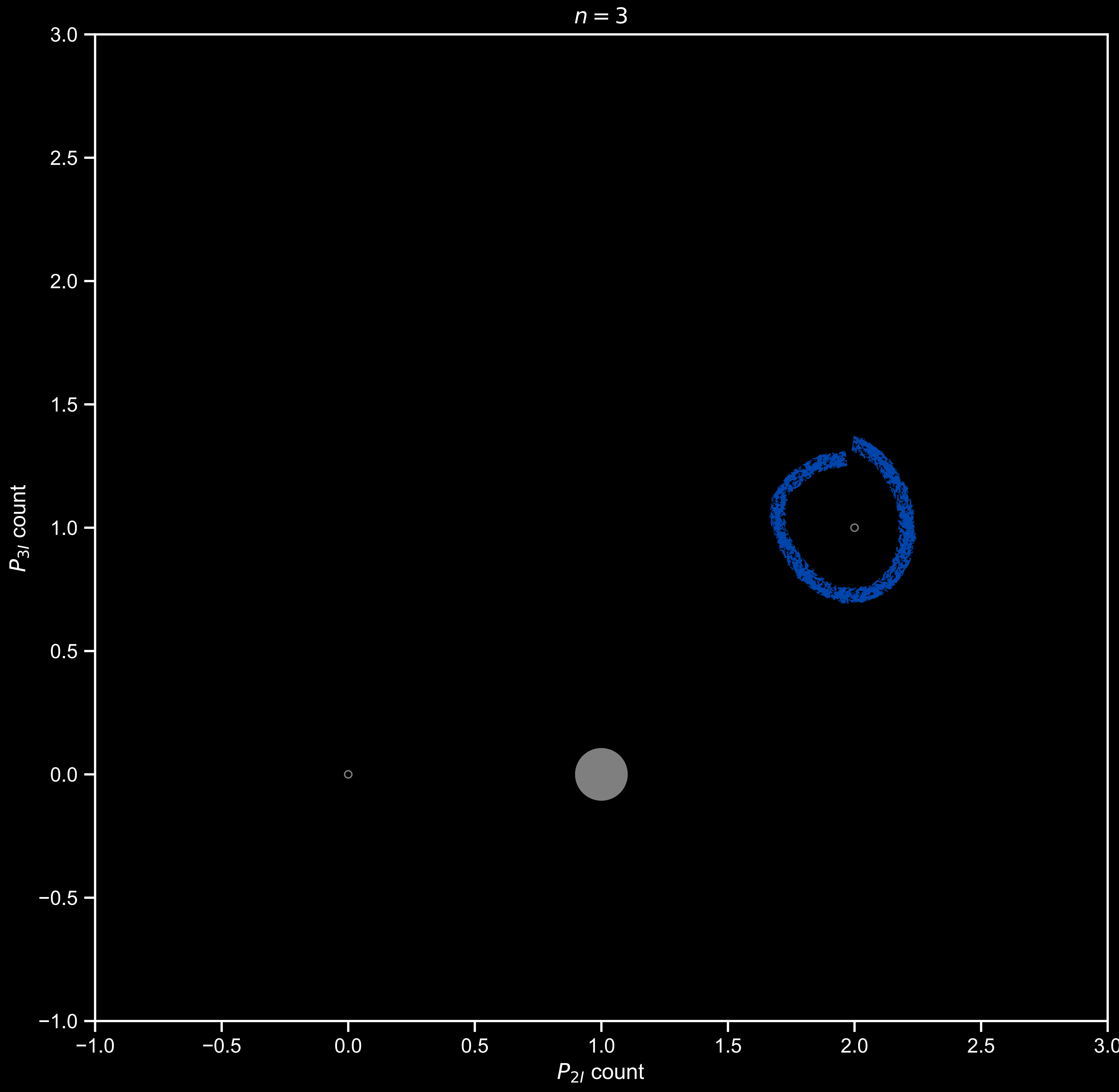
$P_{3I} :=$ 



$\mathscr{T}_3 = \{$  $\}$

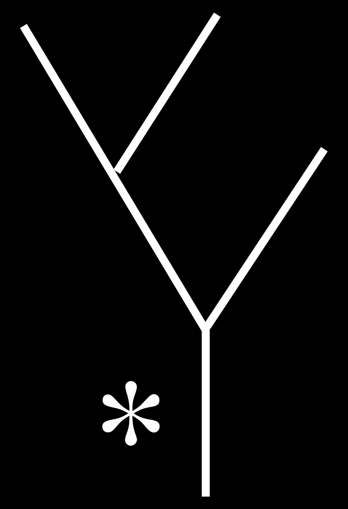
$P_{2I} :=$ 

$P_{3I} :=$ 

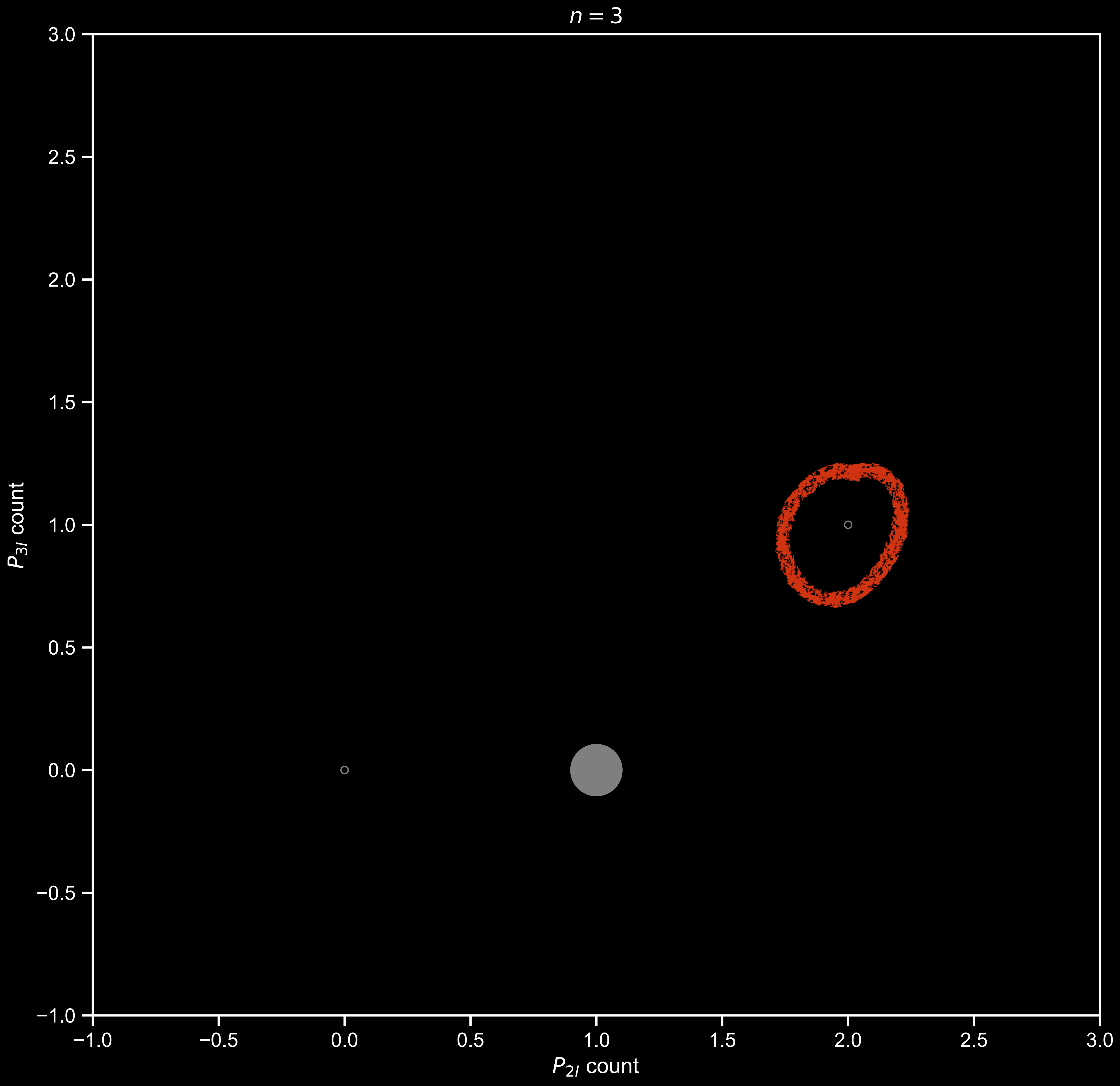


$\mathscr{T}_3 =$ 

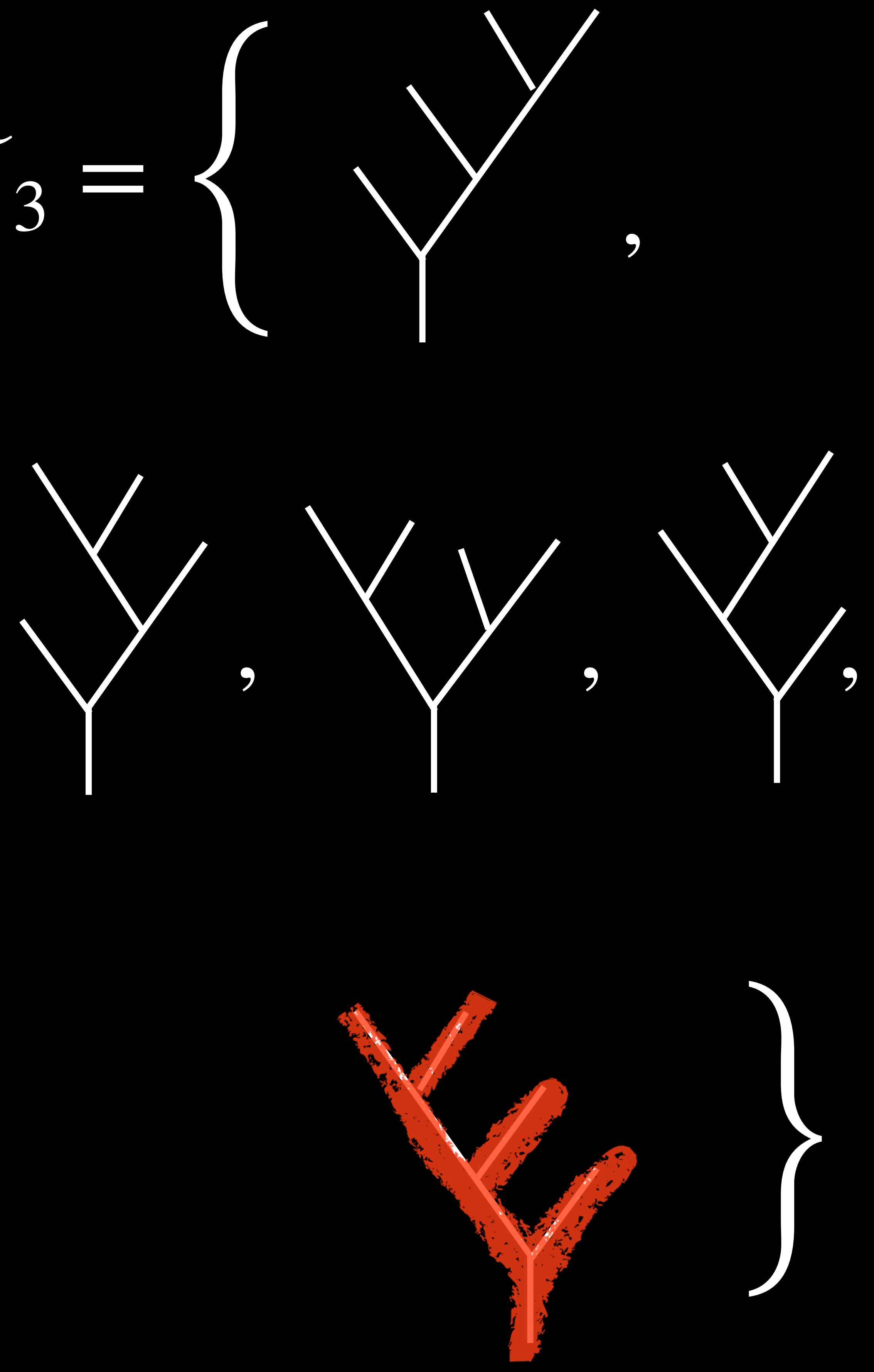
$P_{2I} :=$



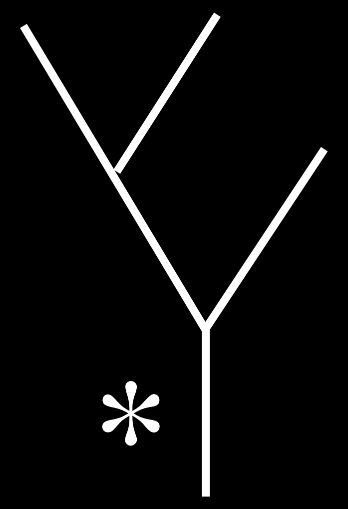
$P_{3I} :=$

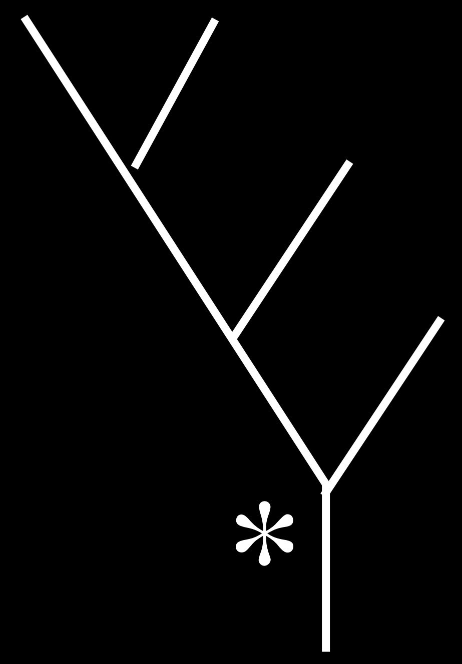


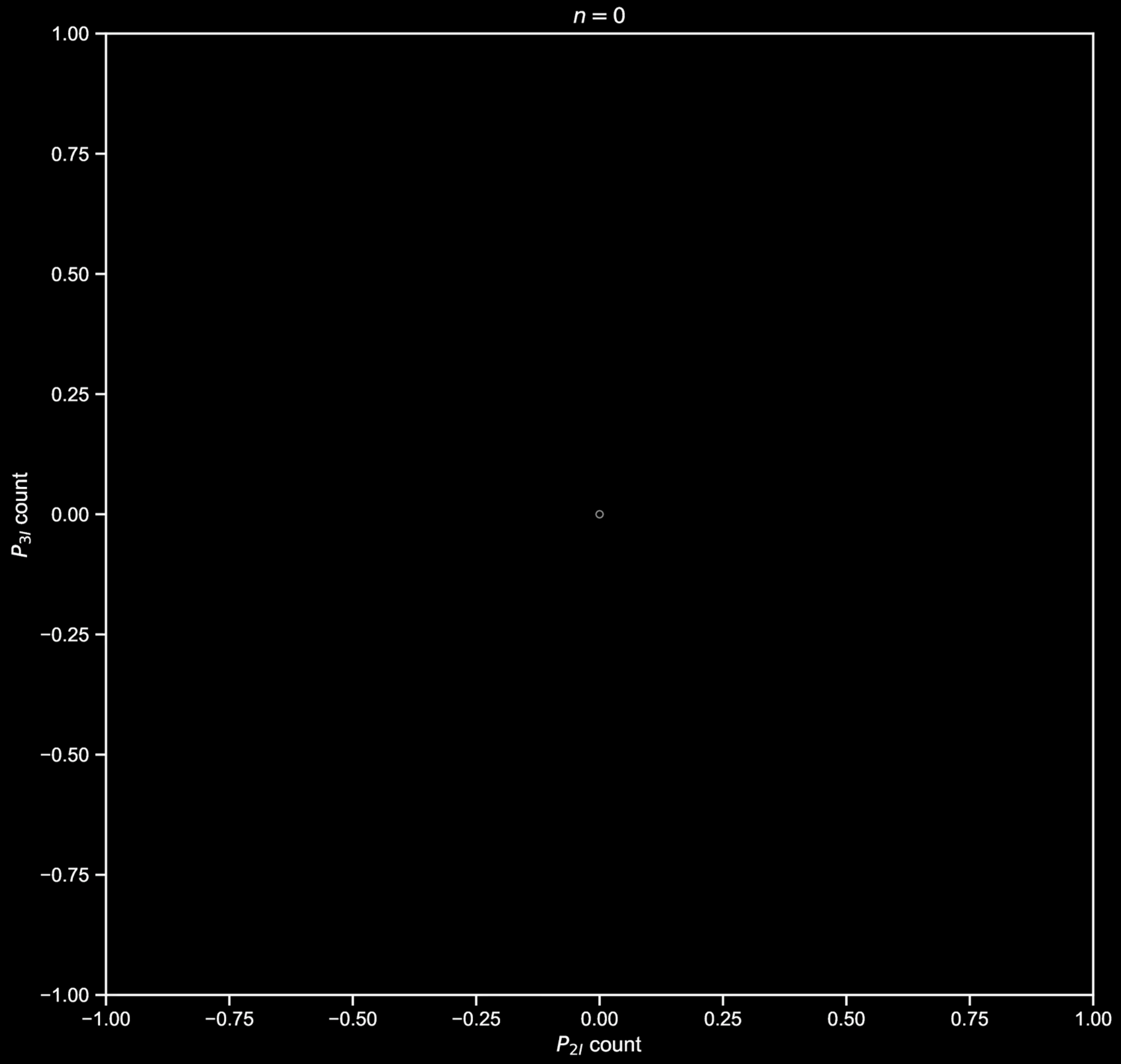
$\mathscr{T}_3 = \{$



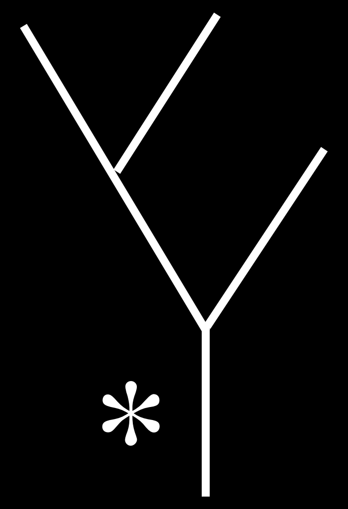
$\}$

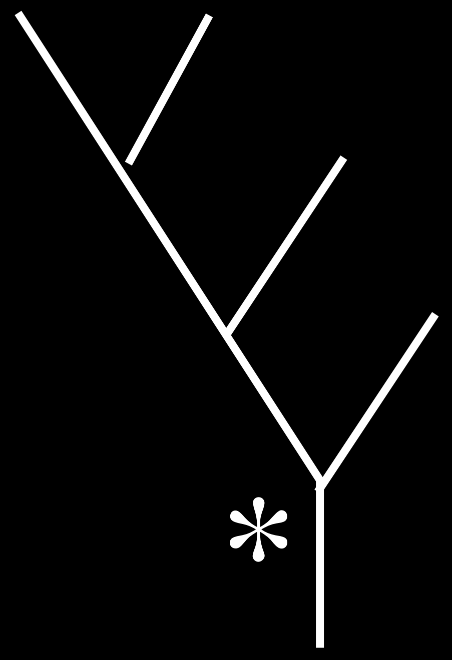
$P_{2I} :=$ 

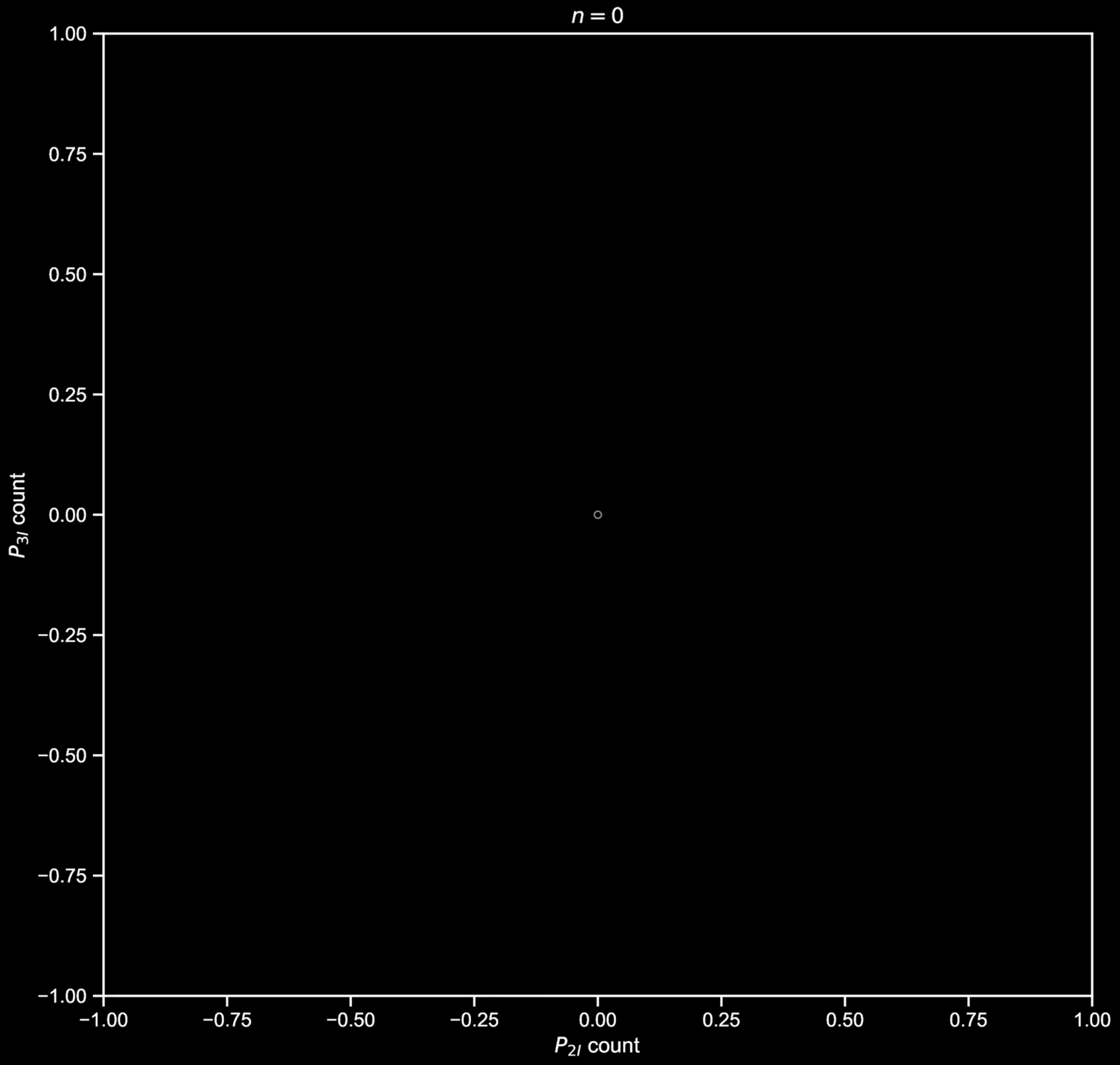
$P_{3I} :=$ 

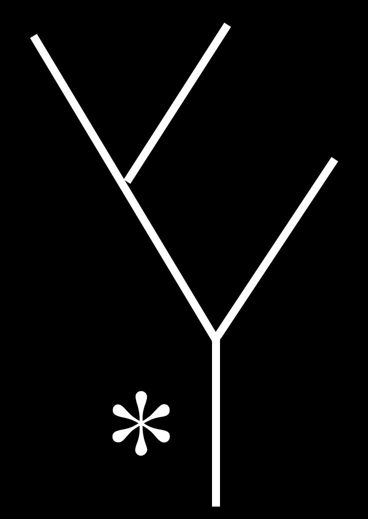


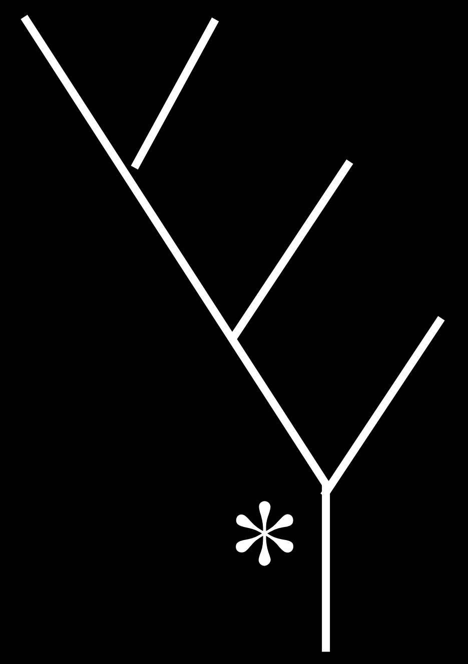
$n = 0$
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
P_{3I} count
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
P_{2I} count

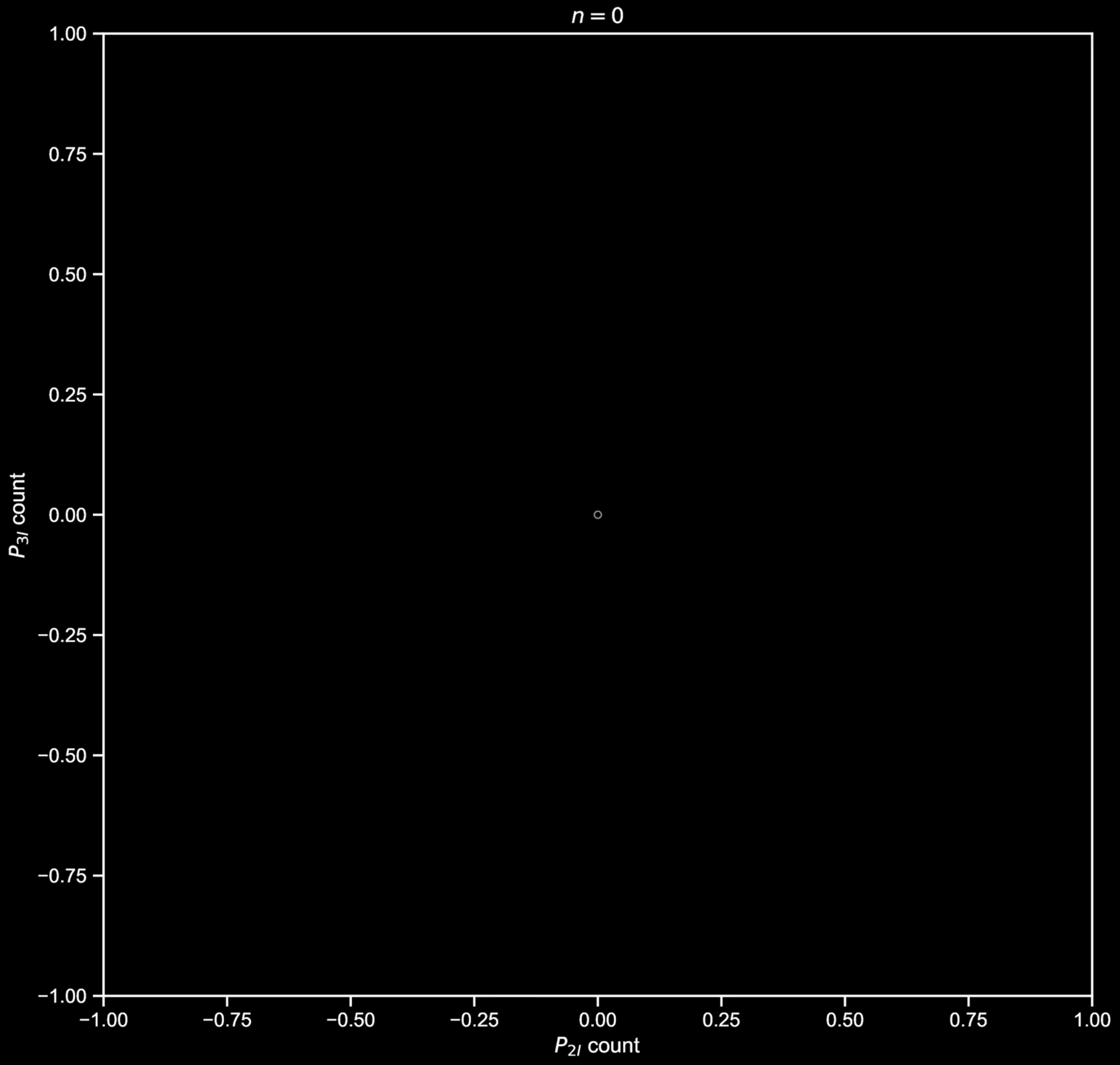
$P_{2I} :=$ 

$P_{3I} :=$ 

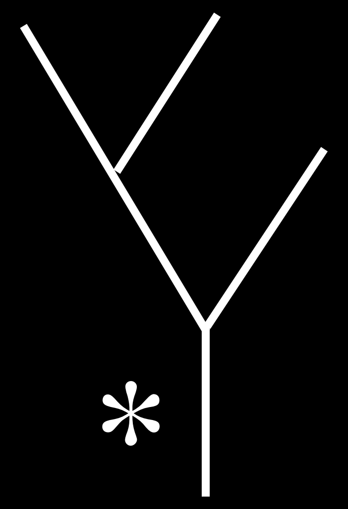

$n = 0$
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
P_{3I} count
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
P_{2I} count

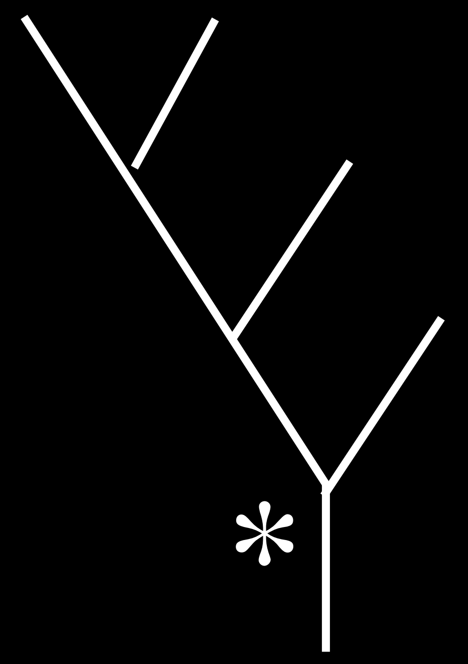
$P_{2I} :=$ 

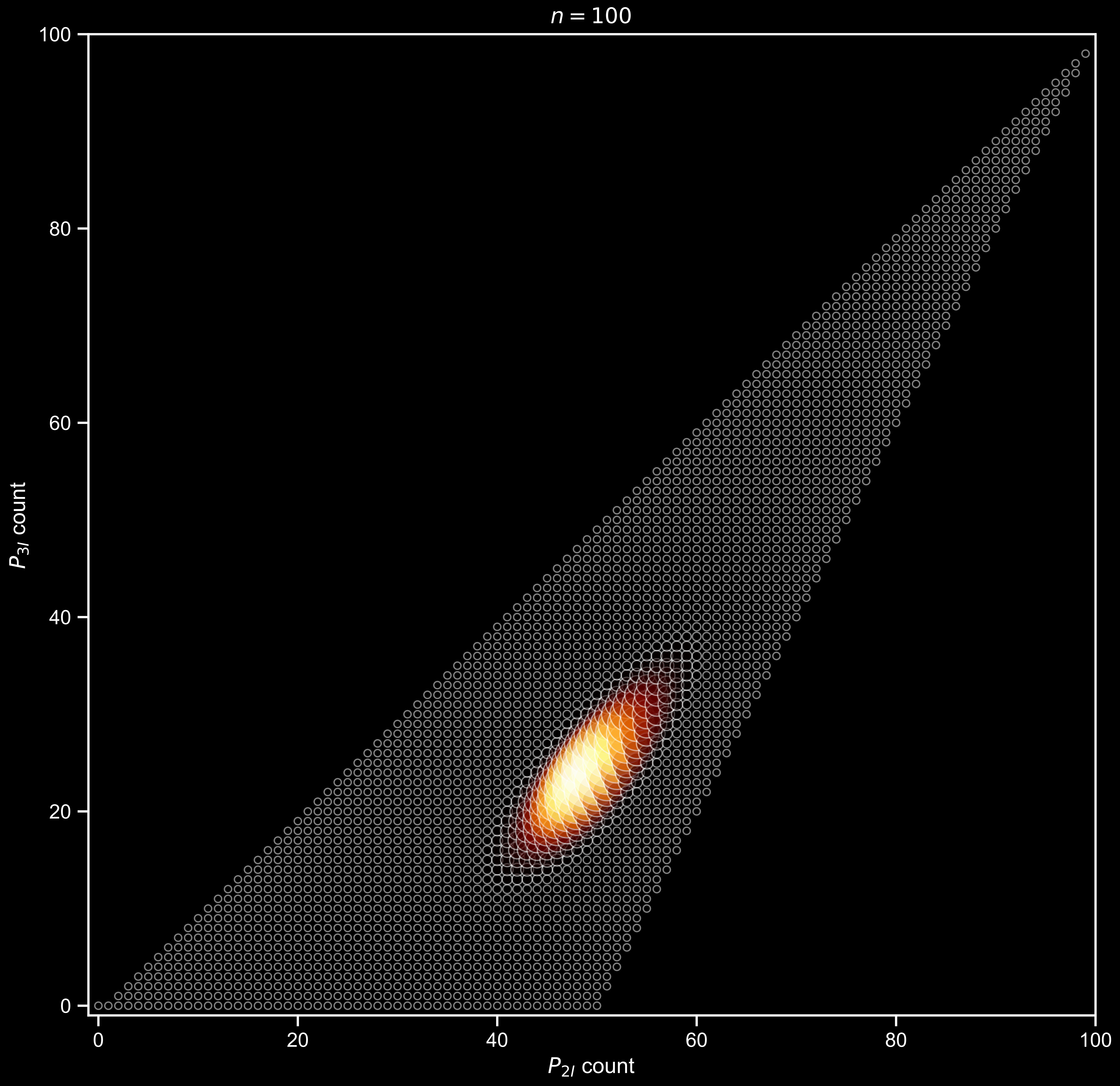
$P_{3I} :=$ 



$$|\mathscr{T}_{100}| = \frac{200!}{100!} \approx 10^{217}$$

$P_{2I} :=$ 

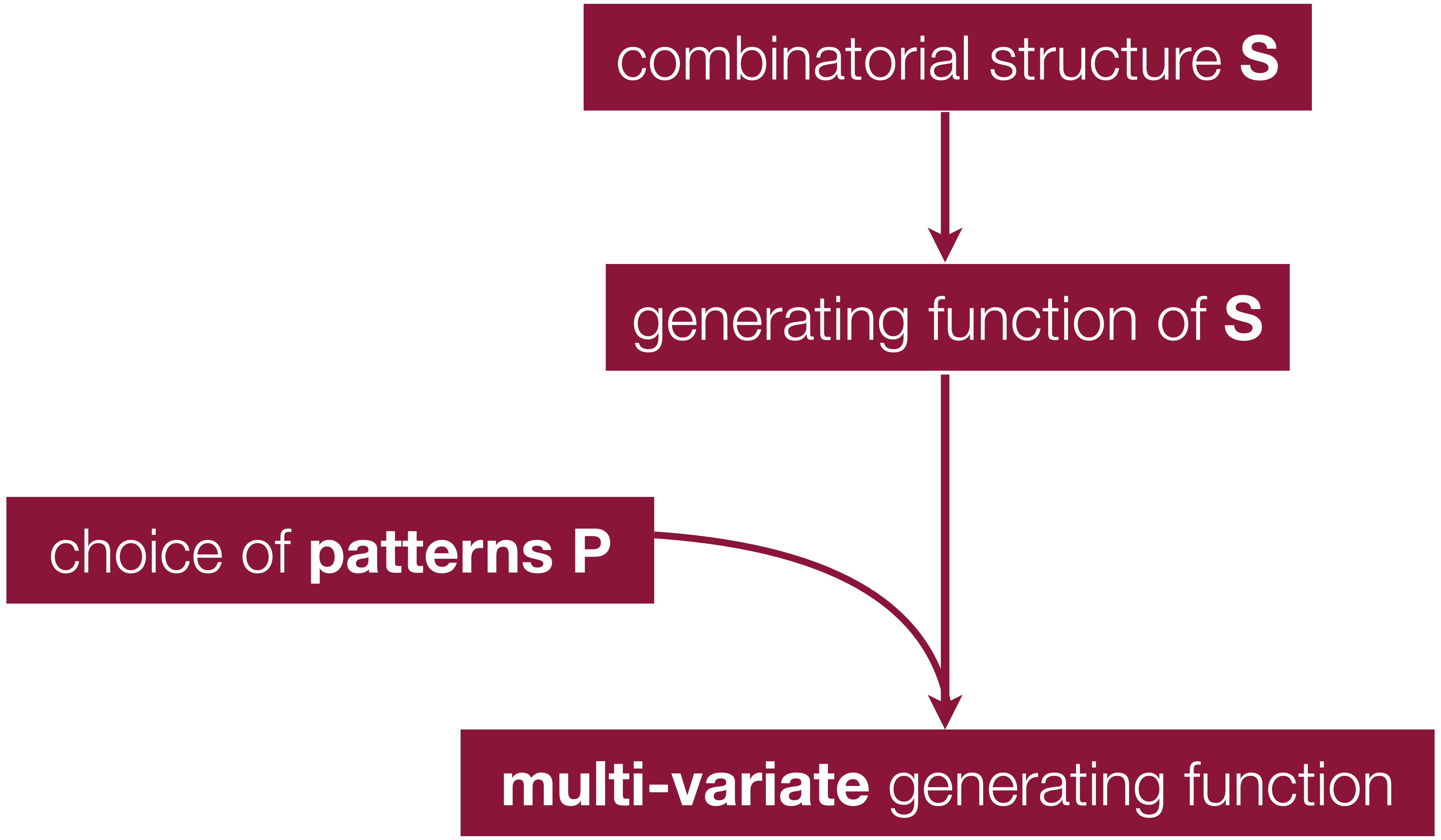
$P_{3I} :=$ 




20
0
0
20
40
60

Summary

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

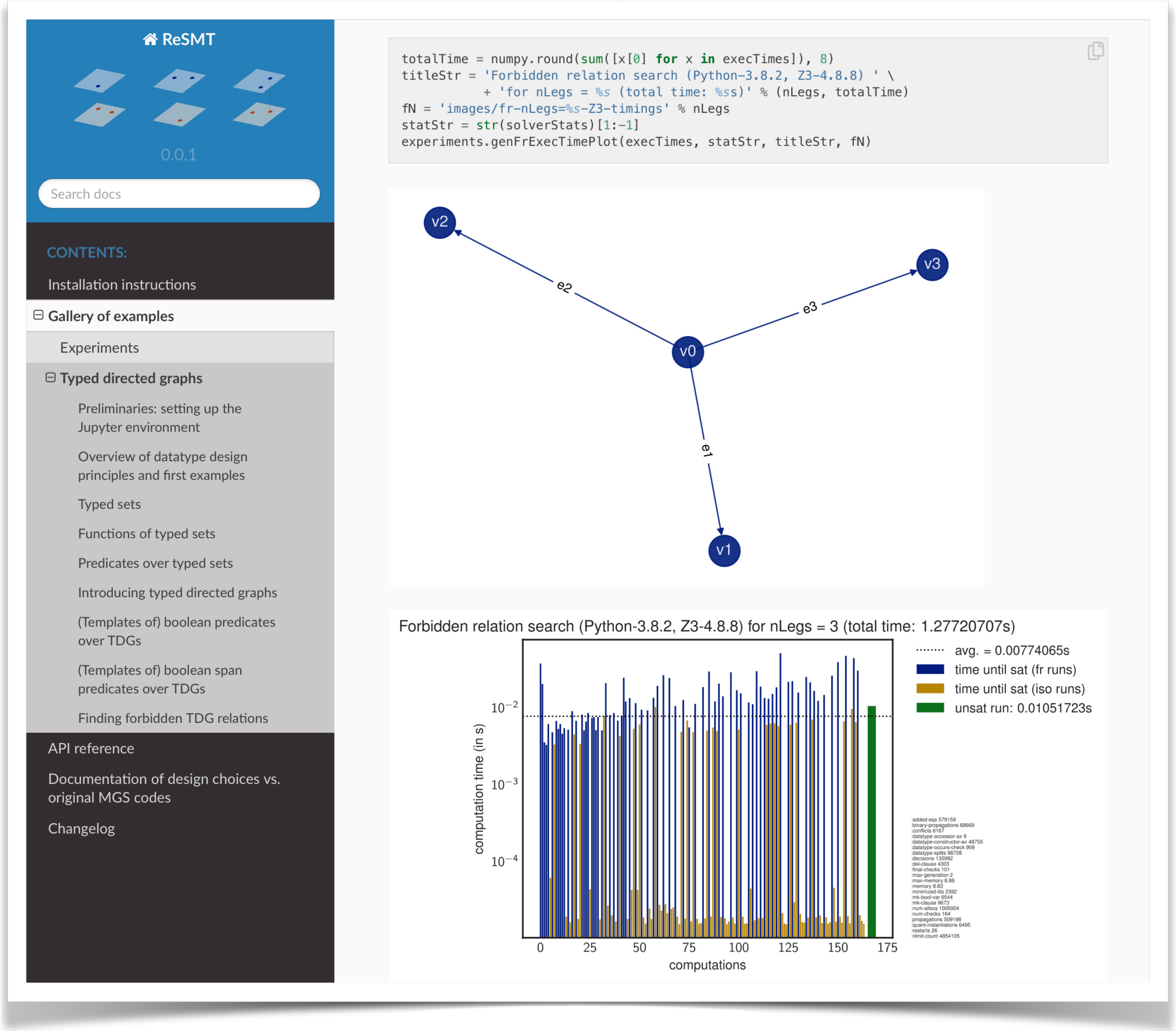


- **generate** structure **S** via applying **rewriting rules** to some **initial configuration** "in all possible ways"
- **count patterns** via applying special types of rewriting rules
- formulate **generating functions** via **linear operators** associated to rewriting rules

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

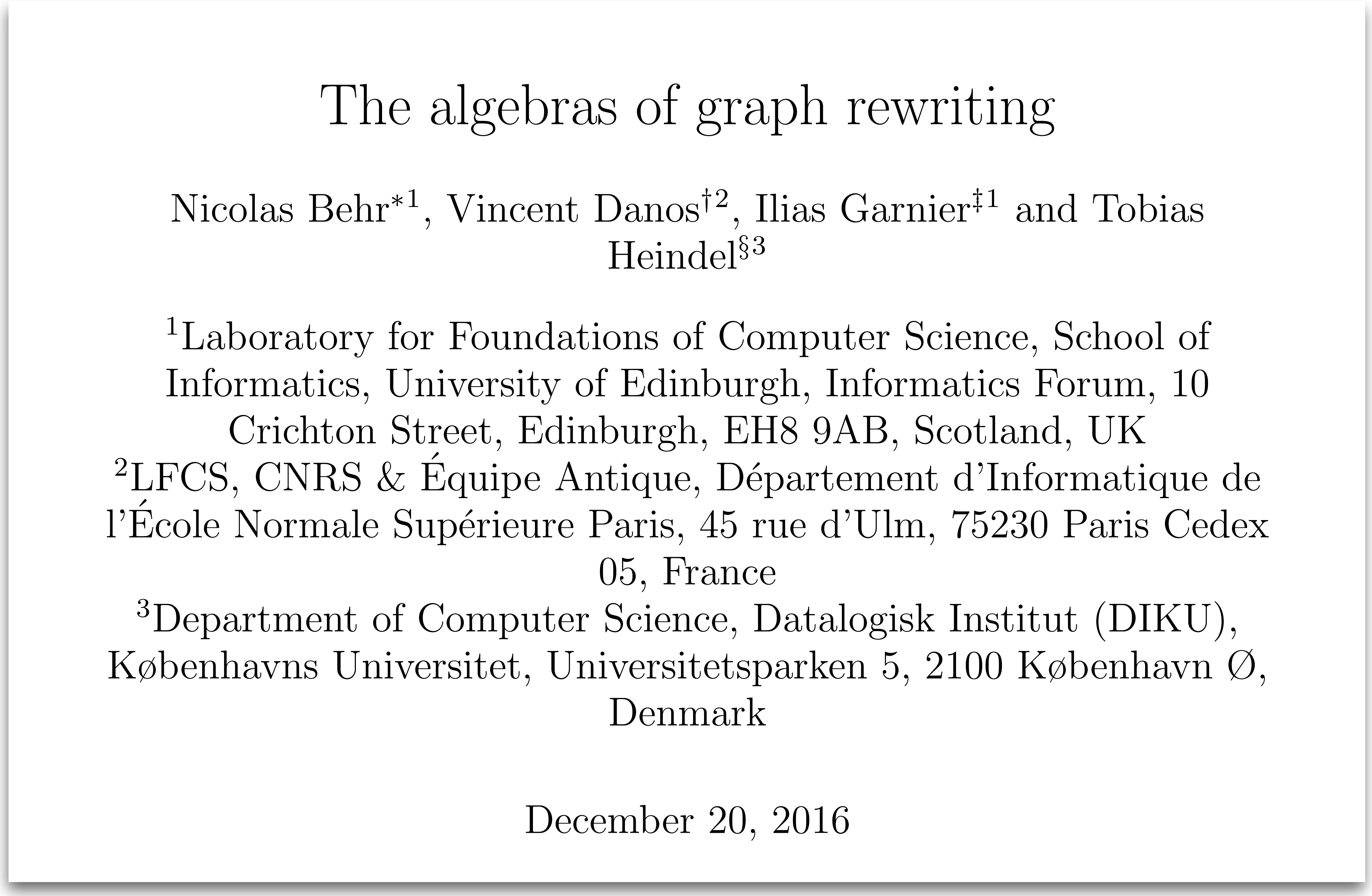
Outlook

- towards **automated computations** via **ReSMT**
- "Flajolet-style" analytic combinatorics via the rule-algebraic approach?
- **asymptotics** of pattern-count distributions?
- **Hopf algebra(s)** of **tracelets**…



https://gitlab.com/nicolasbehr/ReSMT

Outlook

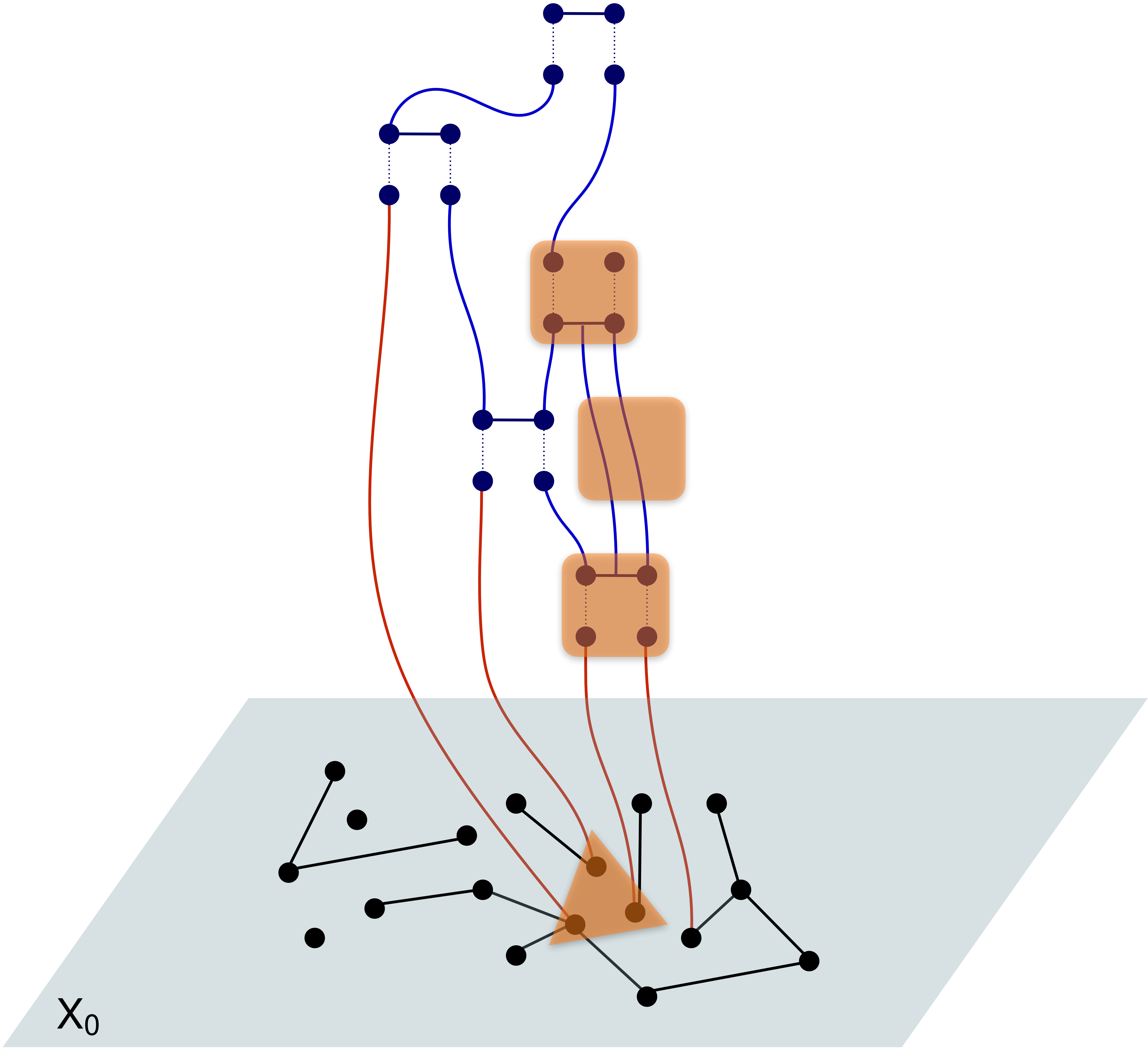


The algebras of graph rewriting

Nicolas Behr*[1], Vincent Danos†[2], Ilias Garnier‡[1] and Tobias Heindel§[3]

[1]Laboratory for Foundations of Computer Science, School of Informatics, University of Edinburgh, Informatics Forum, 10 Crichton Street, Edinburgh, EH8 9AB, Scotland, UK
[2]LFCS, CNRS & Équipe Antique, Département d'Informatique de l'École Normale Supérieure Paris, 45 rue d'Ulm, 75230 Paris Cedex 05, France
[3]Department of Computer Science, Datalogisk Institut (DIKU), Københavns Universitet, Universitetsparken 5, 2100 København Ø, Denmark

December 20, 2016



Merci beaucoup !

SOCS 2020