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):**
  - Combinatorial structure $S$
  - Generating function of $S$
  - Choice of patterns $P$
  - Multi-variate generating function
Motivation

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

- Combinatorial structure $S$
- Generating function of $S$
- Choice of patterns $P$
- Multi-variate generating function
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)

\[ T_0 := \{ \bullet \}, T_1 := \{ \begin{array}{c} L \end{array} \}, T_2 := \{ \begin{array}{c} L \end{array} \}, \ldots \]
Motivation

Example: planar rooted binary trees (PRBTs)

\[ G(\lambda) := \sum_{n \geq 0} \frac{\lambda^n}{n!} \]  

(# of structures of size \( n \))
Motivation

Example: planar rooted binary trees (PRBTs)

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

Choose some patterns:

- \( P_1 := \)
- \( P_2 := \)
- \( P_3 := \)
- \( P_4 := \)
Motivation

Example: planar rooted binary trees (PRBTs)

$$\mathcal{T}_0 := \{ \bullet \}$$, $$\mathcal{T}_1 := \{ \hat{Z} \}$$, $$\mathcal{T}_2 := \{ \hat{Y} \}$$, …

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

Choose some patterns:

$$P_1 := \hat{\bullet}$$, $$P_2 := \hat{Y}$$, $$P_3 := \hat{Y}$$, $$P_4 := \hat{Y}$$

$$G(\lambda; \omega_1, \ldots, \omega_k) := \sum_{n \geq 0} \frac{\lambda^n}{n!} \sum_{p_1, \ldots, p_k \geq 0} \frac{\omega_1^{p_1} \cdots \omega_k^{p_k}}{p_1! \cdots p_k!} \left( \text{# of structures of size } n \text{ and with } p_i \text{ occurrences of pattern } P_i \text{ (for } 1 \leq i \leq k) \right)$$
This notation has a precise mathematical meaning within category theory and forms the basis for the well-definedness of the machinery of graph rewriting. It can intuitively be shown in red. The context contains only edges. Each derivation therefore implies a site graph that exhibits the full interface and state of all its agents with an interface of.

Common to rule-based languages are entities with a structure and a collection of start molecules) comprises all molecules that can be generated from the starting molecules by repeated application of the available rules. Each graph and a key organizing principle in chemistry (Sorger, 2011)

Agents have an interface (ii) appropriate for a particular agent type, site names and states (including binding states) mentioned, though in its infancy, is progressing significantly (et al., 2016).

Although not explicitly represented, these processes are not ignored, as they inform what a rule should resemble of disconnected site graphs, each representing one instance (et al.). A concrete chemical system, which conceptualizes a protein as an agent (Nicolas Behr, SOCS 2020, IRIF, December 11, 2020).

Agents stand for proteins and sites for their interaction capabilities, without, however, representing the underlying physical features and processes enabled. An action, denoted by a site, can anchor at most one edge and a site can anchor at most one action, denoted by a site.

The context contains only edges. Each derivation therefore implies a site graph that exhibits the full interface and state of all its agents with an interface of.

Common to rule-based languages are entities with a structure and a collection of start molecules) comprises all molecules that can be generated from the starting molecules by repeated application of the available rules. Each graph and a key organizing principle in chemistry (Sorger, 2011)

Agents have an interface (ii) appropriate for a particular agent type, site names and states (including binding states) mentioned, though in its infancy, is progressing significantly (et al., 2016).

Although not explicitly represented, these processes are not ignored, as they inform what a rule should resemble of disconnected site graphs, each representing one instance (et al.). A concrete chemical system, which conceptualizes a protein as an agent (Nicolas Behr, SOCS 2020, IRIF, December 11, 2020).

Agents stand for proteins and sites for their interaction capabilities, without, however, representing the underlying physical features and processes enabled. An action, denoted by a site, can anchor at most one edge and a site can anchor at most one action, denoted by a site.

The context contains only edges. Each derivation therefore implies a site graph that exhibits the full interface and state of all its agents with an interface of.

Common to rule-based languages are entities with a structure and a collection of start molecules) comprises all molecules that can be generated from the starting molecules by repeated application of the available rules. Each graph and a key organizing principle in chemistry (Sorger, 2011)

Agents have an interface (ii) appropriate for a particular agent type, site names and states (including binding states) mentioned, though in its infancy, is progressing significantly (et al., 2016).

Although not explicitly represented, these processes are not ignored, as they inform what a rule should resemble of disconnected site graphs, each representing one instance (et al.). A concrete chemical system, which conceptualizes a protein as an agent (Nicolas Behr, SOCS 2020, IRIF, December 11, 2020).

Agents stand for proteins and sites for their interaction capabilities, without, however, representing the underlying physical features and processes enabled. An action, denoted by a site, can anchor at most one edge and a site can anchor at most one action, denoted by a site.

The context contains only edges. Each derivation therefore implies a site graph that exhibits the full interface and state of all its agents with an interface of.

Common to rule-based languages are entities with a structure and a collection of start molecules) comprises all molecules that can be generated from the starting molecules by repeated application of the available rules. Each graph and a key organizing principle in chemistry (Sorger, 2011)

Agents have an interface (ii) appropriate for a particular agent type, site names and states (including binding states) mentioned, though in its infancy, is progressing significantly (et al., 2016).

Although not explicitly represented, these processes are not ignored, as they inform what a rule should resemble of disconnected site graphs, each representing one instance (et al.). A concrete chemical system, which conceptualizes a protein as an agent (Nicolas Behr, SOCS 2020, IRIF, December 11, 2020).

Agents stand for proteins and sites for their interaction capabilities, without, however, representing the underlying physical features and processes enabled. An action, denoted by a site, can anchor at most one edge and a site can anchor at most one action, denoted by a site.

The context contains only edges. Each derivation therefore implies a site graph that exhibits the full interface and state of all its agents with an interface of.

Common to rule-based languages are entities with a structure and a collection of start molecules) comprises all molecules that can be generated from the starting molecules by repeated application of the available rules. Each graph and a key organizing principle in chemistry (Sorger, 2011)

Agents have an interface (ii) appropriate for a particular agent type, site names and states (including binding states) mentioned, though in its infancy, is progressing significantly (et al., 2016).

Although not explicitly represented, these processes are not ignored, as they inform what a rule should resemble of disconnected site graphs, each representing one instance (et al.). A concrete chemical system, which conceptualizes a protein as an agent (Nicolas Behr, SOCS 2020, IRIF, December 11, 2020).

Agents stand for proteins and sites for their interaction capabilities, without, however, representing the underlying physical features and processes enabled. An action, denoted by a site, can anchor at most one edge and a site can anchor at most one action, denoted by a site.
Rewriting in the life sciences: **bio-** and **chemo-informatics**

**KAPPA**

Kappa Language
A rule-based language for modeling interaction networks

**MØD**

Algorithmic Cheminformatics Group

MedØlDatschgerl

`Axin binds a region in the armadillo repeat of β-catenin, if β-catenin is unphosphorylated at 741 and 529`
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
A fundamental challenge: **causal pathway dynamics**
A fundamental challenge: **causal pathway dynamics**

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

**Panel iii**

Axin

**CTNNB1**

**Panel iv**

\[
\text{Axin}(\text{CBD}[.]), \text{ctnnb1}(\text{arm1}[.], T41\{u\}[.], S29\{u\}[.]) \rightarrow \\
\text{Axin}(\text{CBD}[1]), \text{ctnnb1}(\text{arm1}[1], T41\{u\}[.], S29\{u\}[.])
\]
A fundamental challenge: **causal pathway dynamics**

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

We invite the readers to consult [6] or [7] for compact accounts of the relevant technical definitions of
partially effective rewrite systems via pattern-counting observables.

A Unifying Theory of CTMC Semantics
Nicolas Behr and Jean Krivine
Université de Paris, CNRS, IRIF, F-75006, Paris, France

Acknowledgments. The research leading to these results has received funding from the European Community’s
Seventh Framework Programme under grant agreement no. 610030 (Magnus). The authors would like to thank the
participants of the Magna workshop for valuable comments on drafts of this paper. They also thank the anonymous reviews
for their insightful comments and suggestions.

Rewriting Theory for the Life Sciences: A Unifying Theory of CTMC Semantics
Nicolas Behr
Université de Paris, CNRS, IRIF, F-75006, Paris, France

Keywords. Double-Pushout rewriting, bio-physical rewriting, rule-algebraic methods, bio-chemical complex-dynamics.

1 Introduction

Graph rewriting has emerged as a powerful formalism to represent complex systems whose dy-
namics can be captured by a finite set of rules. The most prevalent approaches employed in this field
are based on the notion of a graph rewriting system, sometimes also called a graph transformation
system. The central concept of a graph rewriting system is a graph rewrite rule, which is a pair of
graphs that differ only by a single bond (the bond to be deleted or added in the rewrite process).

Compositionality of Rewriting Rules with Conditions
Nicolas Behr and Jean Krivine
Université de Paris, CNRS, IRIF, F-75006, Paris, France

In the present work, we provide the necessary technical constructions in order to consider
interaction networks using graph rewriting models, in which proteins are the vertices of a graph
structure. This is connected outside the domain of the match (which would yield side effects)
when rules are applied (which makes
possible to consider the situation of a graph rewriting rule that has an effect on a graph structure but
that does not change it).

Institution de Recherche et d’Histoire des Sciences (IHES)
Université de Paris, CNRS UMR 8243
8 Place du Maréchal de Lattre de Tassigny 91405 Palaiseau Cedex France
jean.krivine@irif.fr

In this paper, we contribute to the theoretical foundations of these types of rewriting theory a number of
important developments. First, we develop a unifying theory of continuous-time Markov chains (CTMCs) for stochastic
rewriting systems. Our work is based on the notion of a graph rewriting system, where the
graph rewriting system is a triple consisting of a set of rules, a start graph, and a final graph.

1I n t r o d u c t i o n a n d r e l a t i o n t o p r e v i o u s w o r k

Graph rewriting has emerged as a powerful formalism to represent complex systems whose dy-
namics can be captured by a finite set of rules. The most prevalent approaches employed in this field
are based on the notion of a graph rewriting system, sometimes also called a graph transformation
system. The central concept of a graph rewriting system is a graph rewrite rule, which is a pair of
graphs that differ only by a single bond (the bond to be deleted or added in the rewrite process).

In the present work, we provide the necessary technical constructions in order to consider
interaction networks using graph rewriting models, in which proteins are the vertices of a graph
structure. This is connected outside the domain of the match (which would yield side effects)
when rules are applied (which makes
possible to consider the situation of a graph rewriting rule that has an effect on a graph structure but
that does not change it).

Institution de Recherche et d’Histoire des Sciences (IHES)
Université de Paris, CNRS UMR 8243
8 Place du Maréchal de Lattre de Tassigny 91405 Palaiseau Cedex France
jean.krivine@irif.fr

In this paper, we contribute to the theoretical foundations of these types of rewriting theory a number of
important developments. First, we develop a unifying theory of continuous-time Markov chains (CTMCs) for stochastic
rewriting systems. Our work is based on the notion of a graph rewriting system, where the
graph rewriting system is a triple consisting of a set of rules, a start graph, and a final graph.

Institution de Recherche et d’Histoire des Sciences (IHES)
Université de Paris, CNRS UMR 8243
8 Place du Maréchal de Lattre de Tassigny 91405 Palaiseau Cedex France
jean.krivine@irif.fr

In this paper, we contribute to the theoretical foundations of these types of rewriting theory a number of
important developments. First, we develop a unifying theory of continuous-time Markov chains (CTMCs) for stochastic
rewriting systems. Our work is based on the notion of a graph rewriting system, where the
graph rewriting system is a triple consisting of a set of rules, a start graph, and a final graph.

Institution de Recherche et d’Histoire des Sciences (IHES)
Université de Paris, CNRS UMR 8243
8 Place du Maréchal de Lattre de Tassigny 91405 Palaiseau Cedex France
jean.krivine@irif.fr

In this paper, we contribute to the theoretical foundations of these types of rewriting theory a number of
important developments. First, we develop a unifying theory of continuous-time Markov chains (CTMCs) for stochastic
rewriting systems. Our work is based on the notion of a graph rewriting system, where the
graph rewriting system is a triple consisting of a set of rules, a start graph, and a final graph.
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.

* Supported by the Center for Research and Interdisciplinarity (CRI) and Marie Skłodowska-Curie Individual fellowship (Grant Agreement No. 753750 – RaSiR).
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.
Rewriting Theory for the Life Sciences: A Unifying Theory of CTMC Semantics

Nicolas Behr\textsuperscript{1}\textsuperscript{[0000–0002–8738–5040]} and Jean Krivine\textsuperscript{2}[0000–0001–7261–7462]

\textsuperscript{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

\textsuperscript{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

1 Introduction and relation to previous work

On Stochastic Rewriting and Combinatorics
Nicolas Behrand Jean Krivine
Université de Paris, CNRS, IRIF, F-75006, Paris, France

The author would like to thank Paul-André Melliès and Noam Zeilberger for fruitful discussions and valuable feedback.

For more details, see the main references and further reading section.

On Stochastic Rewriting and Combinatorics
Nicolas Behrand Jean Krivine
Université de Paris, CNRS, IRIF, F-75006, Paris, France

The author would like to thank Paul-André Melliès and Noam Zeilberger for fruitful discussions and valuable feedback.

For more details, see the main references and further reading section.

On Stochastic Rewriting and Combinatorics
Nicolas Behrand Jean Krivine
Université de Paris, CNRS, IRIF, F-75006, Paris, France

The author would like to thank Paul-André Melliès and Noam Zeilberger for fruitful discussions and valuable feedback.

For more details, see the main references and further reading section.

On Stochastic Rewriting and Combinatorics
Nicolas Behrand Jean Krivine
Université de Paris, CNRS, IRIF, F-75006, Paris, France

The author would like to thank Paul-André Melliès and Noam Zeilberger for fruitful discussions and valuable feedback.

For more details, see the main references and further reading section.
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 \textit{(relative) complement}
Set (relative) complement
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.
**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 \cite{LackSobocinski2003}

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**.

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.


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

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**.


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


A category $C$ possesses a **strict initial object** $\emptyset \in \vert C \vert$ (the “empty object”) if

1. $\forall X \in \text{ob}(C) : \exists ! (i_X : \emptyset \rightarrow X) \in \text{mono}(C)$
2. $\forall X \in \text{ob}(C) : \exists (X \rightarrow \emptyset) \Rightarrow X \cong \emptyset$
Example: presheaves

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

- **objects** of $\mathcal{S}$ are functors $F : \mathcal{S}^{op} \to \mathsf{SET}$
- **morphisms** of $\mathcal{S}$ are natural transformations $\phi : F \Rightarrow G$
Example: **presheaves**

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

- **objects** of $\hat{\mathcal{S}}$ are functors $F : \mathcal{S}^{op} \to \text{Set}$
- **morphisms** of $\hat{\mathcal{S}}$ are natural transformations $\phi : F \Rightarrow G$

**Special case:** (directed) **multigraphs** as $\hat{\mathcal{G}}$, where $\mathcal{G} : V \rightrightarrows E$
Example: presheaves

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

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

Special case: (directed) multigraphs as $\hat{G}$, where $G : V \xrightarrow{s} E$

$\Rightarrow$ 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 $\mathcal{S}$ a (small) category, the category $\hat{\mathcal{S}}$ of **presheaves** over $\mathcal{S}$ has

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

**Special case:** (directed) **multigraphs** as $\hat{\mathcal{G}}$, where $\mathcal{G} : V \xrightarrow{s} E$

$\Rightarrow$ 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.

\[
\begin{align*}
G_1(E) & \xrightarrow{G_1(s)} G_1(V) \\
G_2(E) & \xrightarrow{G_2(s)} G_2(V)
\end{align*}
\]

\[
\begin{align*}
G_1(E) & \xrightarrow{G_1(t)} G_1(V) \\
G_2(E) & \xrightarrow{G_2(t)} G_2(V)
\end{align*}
\]

commute
Pushouts implement “gluing” along partial overlaps

### Interpretation:
- $A$ — intersection of $B$ and $C$ in $D$
- $D$ — union of $B$ and $C$ along $A$
Brief comments on abstract category-theoretical operations:

• pushout (PO) along monomorphisms in the category $\text{Set}$:
  
  \[
  \begin{array}{ccc}
  A & \rightarrow & B \\
  \downarrow & & \downarrow \\
  C & \rightarrow & D
  \end{array}
  \]

  Interpretation:
  
  $D$ – union of $B$ and $C$ along $A$

• pushout complement (POC) of $D - \beta B - \beta A$:
  
  A set $C$ and monomorphisms $D - \beta C - \beta A$ such that the square $\square ABCD$ is a pushout

• pullback (PB) along monomorphisms in the category $\text{Set}$:
  
  \[
  \begin{array}{ccc}
  A & \rightarrow & B \\
  \downarrow & & \downarrow \\
  D & \rightarrow & C
  \end{array}
  \]

  Interpretation:
  
  $A$ – intersection of $B$ and $C$ in $D$

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

- **Pushout (PO)** along monomorphisms in the category $\text{Set}$:
  - **Interpretation:** $A$ — intersection of $B$ and $C$ in $D$  
  - $D$ — union of $B$ and $C$ along $A$

- **Pushout complement (POC)** of $D - B - A$:
  - A set $C$ and monomorphisms $D - C - A$ such that the square $\text{pushout}$ is a pushout

- **Pullback (PB)** along monomorphisms in the category $\text{Set}$:
  - **Interpretation:** $A$ — intersection of $B$ and $C$ in $D$
  - $D$ — union of $B$ and $C$ along $A$
Pushout complements implement partial deletions
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 Grothendieke) is an adhesive category.

Sources of constructions of adhesive categories

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


- **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 Grothendieck) is an adhesive category.


• **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.

1 Introduction

An important aspect of the standard theory of continuous-time Markov chains [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 constructable 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

We invite the readers to consult [6] or [7] for compact accounts of the relevant technical definitions of $M\text{-adhesive}$ categories, pullbacks, pushouts, pushout complements, final pullback complements and their respective properties.
Running example: **planar rooted binary trees (PRBTs)**
Running example: **planar rooted binary trees (PRBTs)**

\[
\mathcal{T}_0 := \{ i \}, \quad \mathcal{T}_1 := \{ \hat{1}, \hat{L}, \hat{R} \}, \quad \mathcal{T}_2 := \{ \hat{L}, \hat{R} \}, \quad \ldots
\]
Running example: planar rooted binary trees (PRBTs)

\[ \mathcal{I}_0 := \{ \bullet \}, \quad \mathcal{I}_1 := \{ \mathcal{I} \}, \quad \mathcal{I}_2 := \{ \mathcal{I} \}, \quad \ldots \]

\[ \text{prePRBF} := \text{FinGraph}/T_{PRBF}, \quad T_{PRBF} := \]

Nicolas Behr, SOCS 2020, IRIF, December 11, 2020
Running example: **planar rooted binary trees (PRBTs)**

\[
\mathcal{I}_0 := \{ \bullet \}, \quad \mathcal{I}_1 := \{ \begin{array}{c} L \\ \bullet \\ R \end{array} \}, \quad \mathcal{I}_2 := \{ \begin{array}{c} L \\ \bullet \\ R \\ \circ \end{array} \}, \quad \ldots
\]

**prePRBF** := FinGraph/\(T_{PRBF}\), \(T_{PRBF} := \) 

**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 $p : P \to P'$ be a monomorphism.
Constraints formalism for adhesive categories

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

- $p \models true$ — in words: “$p$ satisfies the condition true”
DefPUPPVU.

Constraints formalism for adhesive categories

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

- $p \vdash \text{true}$ — in words: “$p$ satisfies the condition true”

- for every mono $a : P \leftrightarrow Q$ and every condition $c_Q$ (over $Q$), $p \vdash \exists (a, c_Q)$ iff there exists a mono $q : Q \leftrightarrow P'$ such that $p = q \circ a$ and $q \vdash c_Q$

\[
\exists (a, c_Q) \Rightarrow P \xrightarrow{a} Q \xleftarrow{c_Q} P'
\]
Constraints formalism for adhesive categories

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

- $p \models \text{true} \quad \text{— in words: “} p \text{ satisfies the condition } \text{true”}$
- for every mono $a : P \hookrightarrow Q$ and every condition $c_Q$ (over $Q$), $p \models \exists (a, c_Q)$ iff there exists a mono $q : Q \hookrightarrow P'$ such that $p = q \circ a$ and $q \models c_Q$
- for $c, c', c''$ constraints,
  - $p \models \neg c \iff \neg(p \models c)$
  - $p \models (c' \land c'') \iff (p \models c') \land (p \models c'')$
**Constraints formalism for adhesive categories**

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

- $p \models \text{true}$ — in words: “$p$ satisfies the condition true”
- for every mono $a : P \rightarrow Q$ and every condition $c_Q$ (over $Q$), $p \models \exists (a, c_Q)$ iff there exists a mono $q : Q \rightarrow P'$ such that $p = q \circ a$ and $q \models c_Q$
- for $c, c', c''$ constraints,
  - $p \models \neg c :\iff \neg (p \models c)$
  - $p \models (c' \land c'') :\iff (p \models c') \land (p \models c'')$

**Definition.** An **object** $X \in \text{obj}(\mathbf{C})$ is defined to satisfy a **constraint** $c_{\emptyset}$ (i.e. a condition formulated over the initial object $\emptyset \in \text{obj}(\mathbf{C})$) iff $\iota_X : \emptyset \rightarrow X \models c_{\emptyset}$.
Running example: **planar rooted binary trees (PBRTs)**

\[
\mathcal{T}_0 := \{ \bullet \}, \; \mathcal{T}_1 := \{ \mathcal{L}_1 \mathcal{Y}_1 \}, \; \mathcal{T}_2 := \{ \mathcal{Y}_2, \; \mathcal{Y}_3 \}, \; \ldots
\]
Running example: **planar rooted binary trees (PBRTs)**

\[ \text{prePRBF} := \text{FinGraph}/T_{PRBF}, \quad T_{PRBF} := \]

\[ \mathcal{T}_0 := \{ \cdot \}, \quad \mathcal{T}_1 := \{ \mathcal{T}_{\cdot \mathcal{L}} \}, \quad \mathcal{T}_2 := \{ \mathcal{T}_{\mathcal{L}\mathcal{R}} \}, \ldots \]
Running example: **planar rooted binary trees (PBRTs)**

\[ \text{prePRBF} := \text{FinGraph}/T_{PRBF}, \quad T_{PRBF} := \begin{array}{c} \text{prePRBF} \\ \text{PRBF} \end{array} \]

\[ c_{PRBF} := c_{PRBF}^{(-)} \land c_{PRBF}^{(+)} \]

\[ \mathcal{F}_0 := \{1\}, \quad \mathcal{F}_1 := \{L,R\}, \quad \mathcal{F}_2 := \{L,L,R,R\}, \ldots \]
Running example: planar rooted binary trees (PBRTs)

\[ \text{prePRBF} := \text{FinGraph}/T_{\text{PRBF}}, \quad T_{\text{PRBF}} := \begin{array}{c}
\begin{array}{c}
\text{L} \\
\text{R}
\end{array}
\end{array} \]

\[ c_{\text{PRBF}} := c_{\text{PRBF}} \wedge c_{\text{PRBF}}^{\text{(+)}}, \quad c_{\text{PRBF}}^{\text{(-)}} := \bigwedge_{N \in \mathcal{N}_{\text{PRBF}}} \overline{\mathcal{A}}(\emptyset \rightarrow N), \quad \mathcal{N}_{\text{PRBF}} := \left\{ \begin{array}{c}
\text{L}, \quad \text{R}, \\
\text{L}^L, \quad \text{R}^R,
\end{array} \right\} \cup \bigcup_{T,T' \in \{\text{I},\text{L},\text{R}\}} \left\{ \begin{array}{c}
\text{L}^L, \quad \text{R}^R, \\
\text{L}^R, \quad \text{R}^L
\end{array} \right\} \]

\[ \mathcal{T}_0 := \{ \{ \} \}, \quad \mathcal{T}_1 := \{ \begin{array}{c}
\text{L}
\end{array} \}, \quad \mathcal{T}_2 := \{ \begin{array}{c}
\text{L}, \quad \text{R}, \quad \text{L}^L, \quad \text{R}^R
\end{array} \}, \ldots \]
Running example: **planar rooted binary trees (PBRTs)**

$$\text{prePRBF} := \text{FinGraph}/T_{PRBF}, \quad T_{PRBF} := \begin{tikzpicture} [scale=0.5] \node (n1) at (0,0) [circle,draw] {}; \node (n2) at (-1,-1) [circle,draw] {}; \node (n3) at (1,-1) [circle,draw] {}; \draw (n1) edge [left] (n2) \draw (n1) edge [right] (n3); \end{tikzpicture}$$

$$c_{PRBF} := c_{PRBF}^(-) \land c_{PRBF}^(+), \quad c_{PRBF}^(-) := \bigwedge_{N \in \mathcal{N}_{PRBF}} \bar{A}(\emptyset \leftrightarrow N), \quad \mathcal{N}_{PRBF} := \left\{ \begin{array}{l} \{ 1, 1, 1, 1, 1, 1, 1, 1 \} \cup \bigcup_{T,T' \in \{I,L,R\}} \{ T \wedge T', T \wedge T', T \wedge T' \} \\
\{ T \wedge T', T \wedge T', T \wedge T' \} \\
\{ T \wedge T', T \wedge T', T \wedge T' \} \end{array} \right\}$$

$$c_{PRBF}^(-) := \bigwedge_{N \in \mathcal{N}_{PRBF}} \bar{A}(\emptyset \leftrightarrow N), \quad c_{PRBF}^(+):= \forall (\emptyset \leftrightarrow \begin{array}{l} 1 \leftrightarrow 1, 1 \leftrightarrow 1, 1 \leftrightarrow 1 \end{array}) \land \forall (\emptyset \leftrightarrow \begin{array}{l} 1 \leftrightarrow 1, 1 \leftrightarrow 1, 1 \leftrightarrow 1 \end{array})$$

$$\bigwedge_{T \in \{L,R\}} \forall \left( \emptyset \leftrightarrow T, \bigvee_{T' \in \{I,L,R\}} \exists \left( \begin{array}{l} T \leftrightarrow T' \end{array} \right) \right)$$

$$\mathcal{I}_0 := \{ \bullet \}, \quad \mathcal{I}_1 := \{ \begin{array}{l} L \end{array}, \begin{array}{l} R \end{array} \}, \quad \mathcal{I}_2 := \{ \begin{array}{l} L \begin{array}{l} L \end{array}, \begin{array}{l} R \end{array} \end{array}, \begin{array}{l} R \end{array}, \begin{array}{l} L \end{array} \}, \ldots$$
Running example: planar rooted binary trees (PBRTs)

prePRBF := FinGraph/TPRBF,  TPRBF :=

\[\begin{array}{c}
\text{c}_{\text{PRBF}} := c_{\text{PRBF}}^{(-)} \land c_{\text{PRBF}}^{(+)} \\
\text{c}_{\text{PRBF}}^{(-)} := \bigwedge_{N \in \mathcal{N}_{\text{PRBF}}} \bar{A}(\emptyset \leftrightarrow N), \quad \mathcal{N}_{\text{PRBF}} := \left\{ \text{trees} \right\} \cup \bigcup_{T,T' \in \{I,L,R\}} \left\{ t^{T}, t'T', tT', t'T' \right\}
\end{array}\]

\[\begin{array}{c}
\text{c}_{\text{PRBF}}^{(+)} := \forall \left( \emptyset \leftrightarrow t, \exists \left( t \leftrightarrow t'R \right) \right) \land \forall \left( \emptyset \leftrightarrow t, \exists \left( t \leftrightarrow t'R \right) \right) \\
\land \bigwedge_{T \in \{L,R\}} \bigwedge_{T' \in \{I,L,R\}} \forall \left( \emptyset \leftrightarrow t, \forall \left( t \leftrightarrow t' \right) \right)
\end{array}\]

We then define the set $P_{\text{PRBF}}$ of PRBF patterns and the set $S_{\text{PRBF}}$ of states (with the latter coinciding of course with the set of PRBFs):

$P_{\text{PRBF}} := \left\{ X \in \text{obj}(\text{prePRBF}) \mid X \models c_{\text{PRBF}}^{(-)} \right\}, \quad S_{\text{PRBF}} := \left\{ X \in P_{\text{PRBF}} \mid X \models c_{\text{PRBF}}^{(+)} \right\}$
The central “workflow” of **categorical rewriting theory**

Fix an **adhesive finitary extensive category** $\mathcal{C}$
The central “workflow” of categorical rewriting theory

Fix an **adhesive finitary extensive category** $C$

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

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 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**:

\[ r = (O \xleftarrow{r} I) \equiv (O \xleftrightarrow{o} K \xleftrightarrow{i} I) \]

**Note:** here, “isomorphism” refers to entry-wise isomorphisms of the spans that encode rules, i.e. not the standard notion of isomorphism of spans.
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 7), 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 is obtained by non-determinate sequential concurrent application of rules in Double-Pushout (DPO) and Squared-Pushout (SqPO) rewriting with application conditions based upon Mac Lane’s categories. We uncover an intrinsic interplay between the category-theoretical concepts of conditions on rules and morphisms, the non-determinate sequential concurrent application of rules with application conditions, and finally the property of associativity of the composition of rules.
The suitable notion of rules with conditions

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

$$\overline{\text{Lin}}(\mathcal{C}) := \{(O \leftarrow K^i \rightarrow I; c_l) \mid o, i \in \mathcal{M}, c_l \in \text{cond}(\mathcal{C})\}. \quad (1)$$

We define two rules with conditions $R_j = (r_j, c_{l_j})$ ($j = 1, 2$) equivalent, denoted $R_2 \sim R_1$, iff $c_{l_1} \equiv c_{l_2}$ and if there exist isomorphisms $\omega, \kappa, \iota \in \text{iso}(\mathcal{C})$ such that the diagram on the right commutes. We denote by $\overline{\text{Lin}}(\mathcal{C})_\sim$ the set of equivalence classes under $\sim$ of rules with conditions.

$$O_1 \longleftrightarrow K_1 \longleftrightarrow I_1$$

$$O_2 \longleftrightarrow K_2 \longleftrightarrow I_2 \quad (2)$$
The suitable notion of rule applications

Definition 2. Let \( r = (O \leftrightarrow K \leftrightarrow I) \in \text{Lin}(C) \) and \( c_I \in \text{cond}(C) \) be concrete representatives of some equivalence class \( R \in \text{Lin}(C) \), and let \( X, Y \in \text{obj}(C) \) be objects. Then a type \( T \) direct derivation is defined as a commutative diagram such as below right, where all morphisms are in \( M \) (and with the left representation a shorthand notation)

\[
\begin{align*}
O & \xrightarrow{r} I \\
Y & \xleftarrow{m} X \\
m^*\downarrow & \quad \mathbb{T} \quad \downarrow m \\
\end{align*}
\]

\[
\begin{align*}
O & \longleftrightarrow K \longleftrightarrow I \\
Y & \longleftrightarrow \overline{K} \longleftrightarrow X \\
m^*\downarrow & \quad (B) \quad \uparrow k \quad (A) \quad \downarrow m .
\end{align*}
\]

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

1. \( T = \text{DPO} \): given \((m : I \leftrightarrow X) \in M\), \( m \) is a DPO-admissible match of \( R \) into \( X \), denoted \( m \in M^{\text{DPO}}_R(X) \), if \( m \vdash c_I \) 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 \( r = (O \leftrightarrow K \leftrightarrow I) \in \text{Lin}(C) \) and \( c_I \in \text{cond}(C) \) be concrete representatives of some equivalence class \( R \in \overline{\text{Lin}(C)} \), and let \( X, Y \in \text{obj}(C) \) be objects. Then a type \( T \) \textbf{direct derivation} is defined as a commutative diagram such as below right, where all morphism are in \( M \) (and with the left representation a shorthand notation)

\[
\begin{array}{c}
O & \xrightarrow{r} & I \\
\downarrow m^\ast & & \downarrow m \\
Y & \xleftarrow{\mathbb{T}} & X
\end{array}
\]

\[
\begin{array}{c}
O \xleftarrow{\mathbb{K}} \xrightarrow{K} I \\
\downarrow m^\ast & & \downarrow m \\
Y \xleftarrow{(B)} & \xrightarrow{\mathbb{A}} & K \xleftarrow{r} X
\end{array}
\]

(3)

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

1. \( T = \text{SqPO} \): given \( (m : I \leftrightarrow X) \in M \), \( m \) is a \textbf{SqPO-admissible match of} \( R \) into \( X \), denoted \( m \in M_{R}^{\text{SqPO}}(X) \), if \( m \vDash c_I \), in which case \( (A) \) is constructed as a \textbf{final pullback complement} and \( (B) \) as a \textbf{pushout}.
The suitable notion of rule applications

Definition 2. Let \( r = (O \leftrightarrow K \leftrightarrow I) \in \text{Lin}(\mathbf{C}) \) and \( c_I \in \text{cond}(\mathbf{C}) \) be concrete representatives of some equivalence class \( R \in \overline{\text{Lin}(\mathbf{C})} \), and let \( X, Y \in \text{obj}(\mathbf{C}) \) be objects. Then a type \( T \) 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)

\[
\begin{array}{ccc}
O & \xleftarrow{r} & I \\
m^* \downarrow & \mathcal{T} & \downarrow m \\
Y & \xleftarrow{} & X
\end{array}
\quad := 
\begin{array}{ccc}
O & \xleftarrow{} & K & \xrightarrow{} & I \\
m^* \downarrow & (B) & \xrightarrow{k} & (A) & \xrightarrow{} & m \\
Y & \xleftarrow{} & \overline{K} & \xrightarrow{} & X
\end{array}
\] (3)

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

2. \( T = \text{SqPO} \): given \((m : I \leftrightarrow X) \in \mathcal{M}, m \) is a \textbf{SqPO-admissible match of} \( R \) into \( X \), denoted \( m \in M^\text{SqPO}_R(X) \), if \( m \vdash c_I \), in which case \((A)\) is constructed as a \textbf{final pullback complement} and \((B)\) as a \textbf{pushout}.

For types \( T \in \{\text{DPO}, \text{SqPO}\} \), we will sometimes employ the notation \( R_m(X) \) for the object \( Y \).
The suitable notion of rule applications

Definition 2. Let \( r = (O \leftrightarrow K \leftrightarrow I) \in \text{Lin}(\mathcal{C}) \) and \( c_i \in \text{cond}(\mathcal{C}) \) be concrete representatives of some equivalence class \( R \in \overline{\text{Lin}(\mathcal{C})} \), and let \( X, Y \in \text{obj}(\mathcal{C}) \) be objects. Then a type \( \mathcal{T} \) 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)

\[
\begin{array}{c}
O \xleftarrow{r} I \\
\downarrow m^* \quad \Downarrow m \\
Y \xleftarrow{} X
\end{array}
\quad :=
\begin{array}{c}
O \xleftarrow{} K \xrightarrow{} I \\
\downarrow m^* \quad \Downarrow k \quad \Downarrow m \\
Y \xleftarrow{} K \xleftarrow{} X
\end{array}
\quad (3)
\]

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

3. \( \mathcal{T} = \text{DPO}^\dagger \): given just the “plain rule” \( r \) and \((m^*: O \rightarrow Y) \in \mathcal{M}\), \( m^* \) is a \( \text{DPO}^\dagger \)-admissible match of \( r \) into \( X \), denoted \( m \in M_{r, \text{DPO}^\dagger}(Y) \), if \((B)\) is constructable as a pushout complement, in which case \((B)\) is constructed as a pushout.
categorical rewriting theory
(since the mid-70’s)
categorical rewriting theory
(since the mid-70’s)
a TRACELET
(of length 5)
The suitable notion of rule compositions

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

\[
\begin{array}{c}
O_2 \xleftarrow{r_2} I_2 & \longleftarrow & M_{21} & \longrightarrow & O_1 \xleftarrow{r_1} I_1 \\
\end{array}
\]

(4)
The suitable notion of rule compositions

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

$$
\begin{array}{c}
O_2 \xleftarrow{r_2} I_2 \xrightarrow{\text{PO}} M_{21} \xrightarrow{\text{PO}} O_1 \xleftarrow{r_1} I_1 \\
\end{array}
$$

(4)
The suitable notion of **rule compositions**

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

\[
\begin{array}{cccc}
O_2 & \xleftarrow{r_2} & I_2 & \xleftrightarrow{T} & M_{21} & \xrightarrow{r_1} & O_1 \\
\downarrow & & \vphantom{\leftrightarrow} & & \downarrow & & \vphantom{\leftrightarrow} \\
O_{21} & & N_{21} & & & \\
\end{array}
\]

(4)
The suitable notion of rule compositions

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

\[
\begin{array}{c}
O_2 \xleftarrow{r_2} I_2 & \xleftrightarrow{PO} & M_{21} & \xrightarrow{r_1} O_1 \\
\downarrow T & & \downarrow & \\
O_{21} & \xleftrightarrow{DPO^\dagger} & N_{21} & \xleftrightarrow{I_{21}}
\end{array}
\]  

(4)
The suitable notion of rule compositions

**Definition 3.** Let \( R_1, R_2 \in \overline{\text{Lin}(C)} \) be two equivalence classes of rules with conditions, and let \( r_j \in \text{Lin}(C) \) and \( c_{l_j} \) be concrete representatives of \( R_j \) (for \( j = 1, 2 \)). For \( T \in \{ \text{DPO, SqPO} \} \), an \( M \)-span \( \mu = (l_2 \leftrightarrow M_{21} \leftrightarrow O_1) \) (i.e. with \( (M_{21} \leftrightarrow O_1), (M_{21} \leftrightarrow l_2) \in M \)) is a\( T \)-admissible match of \( R_2 \) into \( R_1 \) if the diagram below is constructable (with \( N_{21} \) constructed by taking pushout)

\[
\begin{array}{ccc}
O_2 & \xleftarrow{r_2} & I_2 \\
\downarrow & & \downarrow \text{PO} \\
O_{21} & \xleftarrow{r_1} & I_1 \\
\end{array}
\]

\[
\begin{array}{ccc}
M_{21} & \xleftarrow{r_2} & O_1 \\
\downarrow & & \downarrow \text{DPO}^\dagger \\
N_{21} & \xleftarrow{r_1} & I_2 \\
\end{array}
\]

and if \( c_{l_{21}} \neq \text{false} \). Here, the condition \( c_{l_{21}} \) is computed as

\[
c_{l_{21}} := \text{Shift}(l_1 \leftrightarrow l_{21}, c_{l_1}) \land \text{Trans}(N_{21} \leftrightarrow l_{21}, \text{Shift}(l_2 \leftrightarrow N_{21}, c_{l_2})).
\]
The suitable notion of **rule compositions**

**Definition 3.** Let $R_1, R_2 \in \overline{\text{Lin}(C)}_{\sim}$ be two equivalence classes of rules with conditions, and let $r_j \in \text{Lin}(C)$ and $c_{ij}$ be concrete representatives of $R_j$ (for $j = 1, 2$). For $\mathbb{T} \in \{\text{DPO}, \text{SqPO}\}$, an $\mathcal{M}$-span $\mu = (I_2 \leftarrow M_{21} \leftarrow O_1)$ (i.e. with $(M_{21} \leftarrow O_1), (M_{21} \leftarrow I_2) \in \mathcal{M}$) 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)

$$
\begin{array}{c}
O_2 \xleftarrow{r_2} I_2 & \xleftarrow{} & M_{21} & \xrightarrow{r_1} O_1 \\
\downarrow \mathbb{T} & & \downarrow \text{PO} & & \downarrow \text{DPO}^\dagger \\
O_{21} & \xleftarrow{} & N_{21} & \xrightarrow{} & I_{21}
\end{array}
$$

(4)

and if $c_{I_{21}} \neq \text{false}$. Here, the condition $c_{I_{21}}$ is computed as

$$c_{I_{21}} := \text{Shift}(l_1 \leftarrow l_{21}, c_{l_1}) \land \text{Trans}(N_{21} \leftarrow l_{21}, \text{Shift}(l_2 \leftarrow N_{21}, c_{l_2})).$$

(5)

In this case, we define the **type $\mathbb{T}$ composition of $R_2$ with $R_1$ along $\mu$**, denoted $R_2^{\mu\downarrow \mathbb{T}}R_1$, as

$$R_2^{\mu\downarrow \mathbb{T}}R_1 := [(O_{21} \leftarrow l_{21}; c_{I_{21}})]_{\sim},$$

(6)

where $(O_{21} \leftarrow l_{21}) := (O_{21} \leftarrow N_{21}) \circ (N_{21} \leftarrow l_{21})$ (with $\circ$ the **span composition** operation).
I. Categorical Rewriting Theory
II. Rule Algebra Framework
III. Rule-Algebraic Combinatorics
Physics insight: the rule algebra formalism

\[
\begin{align*}
(O \xrightarrow{r} I) & \quad \leadsto \quad \delta (O \xrightarrow{r} I) \\
\text{a rule} & \quad \text{a basis vector} \\
\text{of a vector space } \mathcal{R} & 
\end{align*}
\]
Physics insight: the **rule algebra** formalism

\[
\begin{array}{c}
(O \xleftarrow{r} I) \\
\Downarrow \\
\delta (O \xleftarrow{r} I)
\end{array}
\]

a **rule**  

a **basis vector**  

of a **vector space** \( \mathcal{R} \)

**Definition:** the **rule algebra product** \( \ast_{\mathcal{R}} : \mathcal{R} \times \mathcal{R} \to \mathcal{R} \) is defined via

\[
\delta(r_2) \ast_{\mathcal{R}} \delta(r_1) := \sum_{\mu \in M_{r_2}(r_1)} \delta \left( r_2 \overset{\mu}{\leftarrow} r_1 \right)
\]

“sum over ways to compose the rules”
Physics insight: the **rule algebra** formalism

\[
\begin{align*}
(O \xrightarrow{r} I) & \quad \xrightarrow{\delta} \quad \delta (O \xrightarrow{r} I) \\
\text{a rule} & \quad \text{a basis vector} \\
\text{of a vector space } \mathcal{R} & 
\end{align*}
\]

**Definition:** the **rule algebra product** \( \star_{\mathcal{R}} : \mathcal{R} \times \mathcal{R} \to \mathcal{R} \) is defined via

\[
\delta (r_2) \star_{\mathcal{R}} \delta (r_1) := \sum_{\mu \in M_{r_2}(r_1)} \delta \left( r_2 \xleftarrow{\mu} r_1 \right) \quad \text{“sum over ways to compose the rules”}
\]

\[
\begin{align*}
O_2 & \xrightarrow{r_2} I_2 \xleftarrow{\mu} O_1 \xrightarrow{r_1} I_1 \\
O_{21} & \xrightarrow{r_2'} N_{21} \xleftarrow{r_1'} I_{21}
\end{align*}
\]
Physics insight: the **rule algebra** formalism

**Definition:** the rule algebra product \( \star_{\mathcal{R}} : \mathcal{R} \times \mathcal{R} \to \mathcal{R} \) is defined via

\[
\delta(r_2) \star_{\mathcal{R}} \delta(r_1) := \sum_{\mu \in M_{r_2}(r_1)} \delta(r_2 \ benefacto \ r_1)
\]

"sum over ways to compose the rules"

**Theorem**

\( \text{LiCS 2016, CSL 2018, GCM 2019, LMCS 2020, ICGT 2020} \)

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

\( \Rightarrow \) a new fundamental tool in **rewriting theory**, **combinatorics** and **concurrency theory**
Mathematics of chemical reactions

Example: \[ 2X \xrightleftharpoons[\alpha]{\text{X}} X \quad (\alpha \in \mathbb{R}_{>0}) \]
Mathematics of chemical reactions

Example: \( 2X \xrightleftharpoons[\alpha]{\alpha} X \quad (\alpha \in \mathbb{R}_{>0}) \)
Mathematics of **chemical reactions**

**Example:** \[ 2X \xrightleftharpoons[\alpha]{\text{X}} X \quad (\alpha \in \mathbb{R}_{>0}) \]
Mathematics of chemical reactions

Example: \( 2X \xrightarrow{\alpha} X \quad (\alpha \in \mathbb{R}_{>0}) \)

\[ p_n(t) := \Pr(#X = n \text{ at time } t) = ? \]
Mathematics of chemical reactions

**Example:** \[ 2X \xrightarrow{\alpha} X \quad (\alpha \in \mathbb{R}_{>0}) \]

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

**Delbrück (1940):** \[ P(t; x) := \sum_{n \geq 0} p_n(t) x^n \]

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

*Max Delbrück (1906-1981)*

1969 Nobel Prize laureate (medicine and physiology)
Mathematics of chemical reactions

Example: \[ 2X \xrightarrow{\alpha} X \quad (\alpha \in \mathbb{R}_{>0}) \]

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

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

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

a linear operator...

\[ \hat{x}(x^n) := x^{n+1}, \quad \partial_x(x^n) := \begin{cases} 0 & \text{if } n = 0 \\ n \cdot x^{n-1} & \text{if } n > 0 \end{cases} \]
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 $\hat{C}$, 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 $\hat{C}$, e.g. graphs, trees, molecules, ….)

Example: $|n\rangle := |\bullet \ldots \bullet\rangle$ (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 $\hat{C}$, e.g. graphs, trees, molecules, ….)

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

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

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

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

Nicolas Behr, SOCS 2020, IRIF, December 11, 2020
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

\[ \rho : \mathcal{R} \rightarrow \text{End}(\mathcal{C}) \]

is a **representation** of the rule algebra \((\mathcal{R}, *_{\mathcal{R}})\), i.e.

\[ \rho(\delta(r_2)) \rho(\delta(r_1)) |X\rangle = \rho(\delta(r_2) *_{\mathcal{R}} \delta(r_1)) |X\rangle \]
Rule algebra framework (Part II)  

Theorem \( \rho : \mathcal{R} \rightarrow \text{End}(\mathcal{C}) \) is a representation of the rule algebra \((\mathcal{R}, \ast_{\mathcal{R}})\), i.e.

\[
\rho(\delta(r_2)) \rho(\delta(r_1)) |X\rangle = \rho(\delta(r_2) \ast_{\mathcal{R}} \delta(r_1)) |X\rangle
\]

\( |n\rangle := |\bullet \ldots \bullet \rangle \)  
Example:  

\[
\begin{align*}
|n\rangle \quad &:= \quad |\bullet \ldots \bullet \rangle \quad &\text{n vertices} \\
\rho(\delta(\bullet \leftarrow \emptyset)) |n\rangle \quad &:= \quad |n+1\rangle \\
\rho(\delta(\emptyset \leftarrow \bullet)) |n\rangle \quad &:= \quad \begin{cases} 
0 & \text{if } n = 0 \\
 n \cdot |n-1\rangle & \text{if } n > 0
\end{cases}
\end{align*}
\]

\[
\hat{x}(x^n) = x^{n+1}
\]

\[
\partial_x(x^n) = \begin{cases} 
0 & \text{if } n = 0 \\
 n \cdot x^{n-1} & \text{if } n > 0
\end{cases}
\]
Rule algebra framework (Part II)

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

**Theorem**

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

\[ \rho : \mathcal{R} \to \text{End}(\mathcal{C}) \] is a representation of the rule algebra \((\mathcal{R}, *_{\mathcal{R}})\), i.e.

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

\[
\begin{align*}
|n\rangle & := |\bullet \ldots \bullet\rangle \\
\rho(\delta(\bullet \leftarrow \emptyset)) |n\rangle &= |n+1\rangle \\
\rho(\delta(\emptyset \leftarrow \bullet)) |n\rangle &= \begin{cases} 0 & \text{if } n = 0 \\ n \cdot |n-1\rangle & \text{if } n > 0 \end{cases} \\
\end{align*}
\]

Example: \[ \leftrightarrow x^n \]

\[
\begin{align*}
\hat{x}(x^n) &= x^{n+1} \\
\partial_x(x^n) &= \begin{cases} 0 & \text{if } n = 0 \\ n \cdot x^{n-1} & \text{if } n > 0 \end{cases} \\
\end{align*}
\]

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

\[ \alpha \left( \hat{x}^2 \partial_x - \hat{x} \partial_x \right) \]
Rule algebra framework (Part II)

\[ \rho(\delta(r)) \ket{X} := \sum_{m \in M_r(X)} \ket{r_m(X)} \]

**Theorem**

\[ \rho : \mathcal{R} \to \text{End}(\hat{\mathcal{C}}) \] is a **representation** of the rule algebra \((\mathcal{R}, \ast_{\mathcal{R}})\), i.e.

\[ \rho(\delta(r_2)) \rho(\delta(r_1)) \ket{X} = \rho(\delta(r_2) \ast_{\mathcal{R}} \delta(r_1)) \ket{X} \]

\[ |n\rangle := |\bullet \ldots \bullet\rangle \quad \text{n vertices} \]

Example:

\[ \rho(\delta(\bullet \leftarrow \emptyset)) |n\rangle = |n + 1\rangle \quad \iff \quad \hat{x}(x^n) = x^{n+1} \]

\[ \rho(\delta(\emptyset \leftarrow \bullet)) |n\rangle = \begin{cases} 0 & \text{if } n = 0 \\ n \cdot |n - 1\rangle & \text{if } n > 0 \end{cases} \quad \iff \quad \partial_x(x^n) = \begin{cases} 0 & \text{if } n = 0 \\ n \cdot x^{n-1} & \text{if } n > 0 \end{cases} \]

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

\[ \alpha \left( \rho(\delta(\bullet \bullet \leftarrow \bullet)) - \rho(\delta(\bullet \leftarrow \bullet)) \right) \quad \iff \quad \alpha \left( \hat{x}^2 \partial_x - \hat{x} \partial_x \right) \]

\( \Rightarrow \) 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 $R_1, \ldots, R_n \in \text{Lin}(\mathbf{C})$. For some choice of parameters $\gamma_1, \ldots, \gamma_n \in \mathbb{R}$, define a linear operator

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

(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 \( R_1, \ldots, R_n \in \text{Lin}(\mathbf{C}) \). For some choice of parameters \( \gamma_1, \ldots, \gamma_n \in \mathbb{R} \), define a linear operator

\[
\hat{G} := \sum_{j=1}^{n} \gamma_j \rho(\delta(R_j)).
\]  

(7)

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

\[
S_{\hat{G}} := \bigcup_{n>0} S_{\hat{G}}^{(n)}, \quad S_{\hat{G}}^{(n)} := \begin{cases} \{X_0\} & \text{if } n = 0 \\ \{X \in \text{obj}(\mathbf{C}) \equiv | X_0 \Rightarrow_{(n)} X\} & \text{if } n > 0 \end{cases}
\]

(8)

Here, \( \Rightarrow_{(i)} \) \( i>0 \) denotes the reachability relation on \( \text{obj}(\mathbf{C}) \equiv \times^2 \) with respect to the initial configuration \( X_0 \in \text{obj}(\mathbf{C}) \equiv \) and the rule-set \( \{R_j\}_{n=1}^{n} \) used to define \( \hat{G} \).
Running example: **planar rooted binary trees (PRBTs)**

\[
\begin{align*}
\mathcal{I}_0 &:= \{ \bullet \}, & \mathcal{I}_1 &:= \{ \begin{array}{c} L \ \bullet \ R \end{array} \}, & \mathcal{I}_2 &:= \{ \begin{array}{c} L \ \bullet \ R \end{array}, \begin{array}{c} L \ \bullet \ R \end{array} \}, & \ldots \\
\end{align*}
\]
Running example: planar rooted binary trees (PRBTs)

Notational convention: \[ \equiv \begin{array}{ccc} I \, & \equiv L \, & \equiv R \end{array} \]
Running example: **planar rooted binary trees (PRBTs)**

Notational convention:

\[ \equiv \quad \equiv \quad \equiv \]

### The Rémy uniform generator in the rule-algebra formalism

\[
\hat{G} := \hat{G}_L + \hat{G}_R, \quad \hat{G}_L := \sum_{T \in \{I,L,R\}} \rho \left( \delta \left( \begin{array}{c}
\hat{L} \\
\hat{R} \\
\hat{T}
\end{array} \right) \quad \begin{array}{c}
\leftarrow \\
\leftarrow \\
\rightarrow 
\end{array} \quad \begin{array}{c}
\hat{T} \\
\hat{T}
\end{array}; \quad \text{Shift} \left( \emptyset \rightarrow \begin{array}{c}
\hat{T}
\end{array}, c_{PBRT} \right) \right) \right)
\]

\[
\hat{G}_R := \sum_{T \in \{I,L,R\}} \rho \left( \delta \left( \begin{array}{c}
\hat{L} \\
\hat{R} \\
\hat{T}
\end{array} \right) \quad \begin{array}{c}
\rightarrow \\
\rightarrow \\
\leftarrow 
\end{array} \quad \begin{array}{c}
\hat{T} \\
\hat{T}
\end{array}; \quad \text{Shift} \left( \emptyset \rightarrow \begin{array}{c}
\hat{T}
\end{array}, c_{PBRT} \right) \right) \right)
\]
The importance of this distinction between patterns and states is that in general one may define observables, namely those of the form $\text{distribution over all trees in "generation"}$ that the resulting conditions are subsumed by the ones explicitly mentioned in (45).

We may finally define $G_0$, $|t|$, while $G$, $\hat{G}$, $\hat{G}_L$, $\hat{G}_R$, $\hat{T}$, $\hat{P}$, $\hat{C}$, $\hat{R}$, $\hat{L}$, $\hat{I}$, $\hat{D}$, $\hat{M}$, and $\hat{G}$ by the rules (as black vertices in the compressed notation, and in graphs and analyze PRBTs via the so-called constraint-preserving graphs).

From hereon, we will simplify our graphical notations via omitting the vertices when drawing PRBTs, as elements of connected planar rooted binary trees (PRBTs) with $\hat{G}$, $\hat{G}_L$, $\hat{G}_R$, $\hat{T}$, $\hat{P}$, $\hat{C}$, $\hat{R}$, $\hat{L}$, $\hat{I}$, $\hat{D}$, $\hat{M}$, and $\hat{G}$ by the rules (as black vertices in the compressed notation, and in graphs and analyze PRBTs via the so-called constraint-preserving graphs).

The Rémy uniform generator in the rule-algebra formalism:

\[ \hat{G} := \hat{G}_L + \hat{G}_R, \hat{G}_L := \sum_{T \in \{L, R\}} \rho \left( \delta \left( \begin{array}{c} \text{Shift} \left( \emptyset \rightarrow ^T, c_{\text{PRBT}} \right) \end{array} \right) \right) \]

\[ \hat{G}_R := \sum_{T \in \{L, R\}} \rho \left( \delta \left( \begin{array}{c} \text{Shift} \left( \emptyset \rightarrow ^T, c_{\text{PRBT}} \right) \end{array} \right) \right) \]
Running example: planar rooted binary trees (PRBTs)

Notational convention: \( \equiv \begin{align*} & I, \quad \setminus \equiv L, \quad \slash \equiv R \end{align*} \)

The Rémy uniform generator in the rule-algebra formalism

\[
\hat{G} := \hat{G}_L + \hat{G}_R, \quad \hat{G}_L := \sum_{T \in \{L,R\}} \rho \left( \delta \left( \begin{array}{c} \bullet \\ \hat{L} \\ \hat{T} \end{array} \right) \leftrightarrow \begin{array}{c} \bullet \\ \hat{R} \\ \hat{T} \end{array} \right) ; \text{Shift} \left( \emptyset \mapsto \begin{array}{c} \bullet \\ \hat{T} \end{array} , {c_{PBRT}} \right) \right) \\
\hat{G}_R := \sum_{T \in \{L,R\}} \rho \left( \delta \left( \begin{array}{c} \bullet \\ \hat{L} \\ \hat{T} \end{array} \right) \leftrightarrow \begin{array}{c} \bullet \\ \hat{R} \\ \hat{T} \end{array} \right) ; \text{Shift} \left( \emptyset \mapsto \begin{array}{c} \bullet \\ \hat{T} \end{array} , {c_{PBRT}} \right) \right) 
\]

\[
\hat{G} |t\rangle = \sum_{t \in \mathcal{T}_1} 2! |t\rangle, \quad \forall t \in \mathcal{T}_1 : \hat{G} |t\rangle = \sum_{t' \in \mathcal{T}_2} 3! |t'\rangle, \ldots, \forall t \in \mathcal{T}_n : \hat{G} |t\rangle = \sum_{t' \in \mathcal{T}_{n+1}} (n+2)! |t'\rangle
\]
Counting **patterns** in combinatorial structures

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

$$\hat{O}_{P,q,c_P} := \rho^\mathcal{T}_{c} \left( \delta \left( P \xleftrightarrow{q} Q \xrightarrow{q} P; c_P \right) \right), \quad \hat{O}_{P,c_p} := \rho^\mathcal{T}_{c} \left( \delta \left( P \xleftrightarrow{id_P} P \xrightarrow{id_P} P; c_P \right) \right)$$

(9)
Counting **patterns** in combinatorial structures

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

$$
\hat{O}_{P,q;c_P} := \rho^{\text{DPO}}_C \left( \delta \left( P \xleftrightarrow{q} Q \xrightarrow{a} P; c_P \right) \right), \quad \hat{O}_{P;c_P} := \rho^{\text{SqPO}}_C \left( \delta \left( P \xleftrightarrow{id_P} P \xrightarrow{id_P} P; c_P \right) \right)
$$

(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:

$$
\forall R = \left( O \xleftrightarrow{\alpha} K \xrightarrow{i} I, c_I \right) \in \text{Lin}(C) : \quad \langle | \rho^\mathcal{T}_C(\delta(R)) = \langle | \hat{O}_\mathcal{T}(\delta(R)) \\
\hat{O}_{\text{DPO}}(\delta(R)) := \hat{O}_{l,k;c_l}, \quad \hat{O}_{\text{SqPO}}(\delta(R)) := \hat{O}_{l;c_l}, \quad \langle | : C \rightarrow \mathbb{R} : |X| \mapsto \langle | X \rangle := 1_{\mathbb{R}}.
$$

(10)
Counting **patterns** in combinatorial structures

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

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

(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:

$$
\forall R = \left( O \xleftarrow{o} K \xrightarrow{i} l, c_l \right) \in \text{Lin}(C) : \quad \langle | \rho^{\mathcal{T}}_{C}(\delta(R)) \rangle = \langle | \hat{O}_{\mathcal{T}}(\delta(R)) \rangle \\
\hat{O}_{\text{DPO}}(\delta(R)) := \hat{O}_{1,k;c_l} , \quad \hat{O}_{\text{SqPO}}(\delta(R)) := \hat{O}_{l;l,c_l} , \quad \langle | : \hat{C} \rightarrow l; \ 2 \rangle : \ |X\rangle \mapsto \langle |X\rangle := 1_{l}.
$$

(10)

In other words, $\hat{O}_{1,k;c_l}$ and $\hat{O}_{l;l,c_l}$ permit to **count** the number of matches of the rewriting rule $R = (O \xleftarrow{o} K \xrightarrow{i} l, c_l)$ in DPO- and SqPO-semantics, respectively. More explicitly, we find that

$$
\langle | \hat{O}_{P,q;c_l} |X\rangle = | M_{P \xleftarrow{Q} P; c_l}^{\text{DPO}}(X) | , \quad \langle | \hat{O}_{P;c_l} |X\rangle = | M_{P \xleftarrow{id_P} P \xrightarrow{id_P} P; c_l}^{\text{SqPO}}(X) | .
$$

(11)
Counting patterns in combinatorial structures

Example 1. The simplest type of observables encountered in practice are the “plain” pattern-counting observables $\hat{O}_P = \hat{O}_{P,\text{id}_P;\text{true}} = \hat{O}_{P;\text{true}}$, 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 \( \hat{O}_P = \hat{O}_{p,\text{id} \text{;} \text{true}} = \hat{O}_{p,\text{true}} \), with typical examples including

- \( \hat{O}_P \) (counting **vertices**),
Counting *patterns* in combinatorial structures

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

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

- $\hat{O}_{**}$ (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 $\hat{O}_P = \hat{O}_{P, \text{id}_P; \text{true}} = \hat{O}_{P; \text{true}}$, with typical examples including

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

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

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

$$\hat{O}_{..\leftrightarrow..;\emptyset(\bullet\bullet\bullet\bullet\bullet\bullet)} = \hat{O}_{..;\emptyset(\bullet\bullet\bullet\bullet\bullet\bullet)}$$

counts **pairs of vertices not linked by an edge**.
A rule-algebraic \textit{generating-functionology}

Definition 6.
Definition 6.

let $|X_0\rangle \in \hat{C}$ denote the initial state

\[
|X_0\rangle
\]  

(12)
A rule-algebraic \textit{generating-functionology}

\textbf{Definition 6.} Let $\hat{G}$ be a linear operator (the \textit{generator}),
let $|X_0\rangle \in \hat{C}$ denote the \textit{initial state}

$$e^{\lambda \hat{G}} |X_0\rangle$$ (12)
A rule-algebraic generating-functionology

Definition 6. Let \( \hat{G} \) be a linear operator (the generator), let \( \hat{O}_1, \ldots, \hat{O}_m \) be a choice of (finitely many) pattern observables, and let \( |X_0\rangle \in \hat{C} \) denote the initial state

\[
e^{\omega \cdot \hat{O}} e^{\lambda \hat{G}} |X_0\rangle
\]  

(12)
A rule-algebraic generating-functionology

**Definition 6.** Let \( \hat{G} \) be a linear operator (the generator), let \( \hat{O}_1, \ldots, \hat{O}_m \) be a choice of (finitely many) pattern observables, and let \( |X_0\rangle \in \mathcal{X} \) denote the initial state. Then the exponential moment-generating function (EMGF) \( G(\lambda; \omega) \) is defined as

\[
G(\lambda; \omega) := \langle \omega \cdot \hat{O} e^{\lambda \hat{G}} | X_0 \rangle
\]

Here, we employed the shorthand notation \( \omega \cdot \hat{O} := \sum_{j=1}^{m} \omega_j \hat{O}_j \), and \( \lambda \) as well as \( \omega_1, \ldots, \omega_m \) are formal variables.
A rule-algebraic generating-functionology

**Definition 6.** Let \( \hat{\mathcal{G}} \) be a linear operator (the generator), let \( \hat{O}_1, \ldots, \hat{O}_m \) be a choice of (finitely many) pattern observables, and let \( |X_0\rangle \in \hat{\mathcal{C}} \) denote the initial state. Then the exponential moment-generating function (EMGF) \( \mathcal{G}(\lambda; \omega) \) is defined as

\[
\mathcal{G}(\lambda; \omega) := \langle | e^{\omega \cdot \hat{O}} e^{\lambda \hat{\mathcal{G}}} |X_0\rangle \quad (12)
\]

Here, we employed the shorthand notation \( \omega \cdot \hat{O} := \sum_{j=1}^{m} \omega_j \hat{O}_j \), and \( \lambda \) as well as \( \omega_1, \ldots, \omega_m \) are formal variables.
A rule-algebraic generating-functionology

Definition 6. Let $\hat{\mathcal{G}}$ be a linear operator (the generator), let $\hat{O}_1, \ldots, \hat{O}_m$ be a choice of (finitely many) pattern observables, and let $|X_0\rangle \in \hat{\mathcal{G}}$ denote the initial state. Then the exponential moment-generating function (EMGF) $G(\lambda; \omega)$ is defined as

\[
G(\lambda; \omega) := \langle | e^{\omega \cdot \hat{\mathcal{O}}} e^{\lambda \hat{\mathcal{G}}} | X_0 \rangle
\]

(12)

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

- Since we assume $X_0$ to be a finite object, clearly each of the sets $S^{(n)}_G$ is of finite cardinality.

- The coefficients $g_n = \langle | \hat{\mathcal{G}}^n | X_0 \rangle$ are evidently of finite value as well, which in summary permits the following repartition of the formal power series $G(\lambda; 0)$:

\[
G(\lambda; 0) = \sum_{n \geq 0} \frac{\lambda^n}{n!} \sum_{X \in S^{(n)}_G} g_n(X), \quad g_n(X) := \langle X | \hat{\mathcal{G}}^n | X_0 \rangle
\]

(13)

Consequently, the configurations $X \in S^{(n)}_G$ 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.

Nicolas Behr, SOCS 2020, IRIF, December 11, 2020
A rule-algebraic generating-functionology

Definition 6. Let \( \hat{G} \) be a linear operator (the generator), let \( \hat{O}_1, \ldots, \hat{O}_m \) be a choice of (finitely many) pattern observables, and let \( |X_0\rangle \in \hat{C} \) denote the initial state. Then the exponential moment-generating function (EMGF) \( G(\lambda; \omega) \) is defined as

\[
G(\lambda; \omega) := \langle | e^{\omega \cdot \hat{O}} e^{\lambda \cdot \hat{G}} | X_0 \rangle
\]

(12)

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

- For generic values of \( \omega \), \( G(\lambda; \omega) \) evaluates as follows:

\[
G(\lambda; \omega) = \sum_{n \geq 0} \frac{\lambda^n}{n!} \langle | e^{\omega \cdot \hat{O}} \hat{G}^n | X_0 \rangle = \sum_{n \geq 0} \frac{\lambda^n}{n!} \sum_{X \in S_c^{(n)}} g_n(X) e^{\omega \cdot N(X)}, \quad N_i(X) := \langle | \hat{O}_i | X \rangle.
\]

(14)
Combinatorial evolution equations

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

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

(15)
Combinatorial evolution equations

The formal EMGF evolution equation for $G(\lambda; \omega)$ reads as follows:

$$\frac{\partial}{\partial \lambda} G(\lambda; \omega) = \langle | (e^{ad_{\omega} \cdot \hat{G}}) e^{\omega \cdot \hat{G}} \hat{G} | X_0 \rangle \quad (ad_A(B) := AB - BA)$$

(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:

$$(PJC') \quad \forall q \in \mathbb{Z}_{\geq 0} : \exists N(n) \in \mathbb{Z}_{\geq 0}^m, \gamma_q(\omega, k) \in \mathbb{R} : \langle | ad_{\omega} \cdot \hat{G} \rangle = \sum_{k=0}^{N(q)} \gamma_k(\omega, k) \langle | \hat{O}^k \rangle \quad (16)$$
Combinatorial evolution equations

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

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

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:

$$(PJC') \quad \forall q \in \mathbb{Z}_{\geq 0} : \exists \underbrace{N(n)}_{\in \mathbb{Z}_{\geq 0}}, \gamma_q(\omega, k) \in \mathbb{R} : \langle \big| ad_{\omega}^q \hat{G} \rangle = \sum_{k=0}^{N(q)} \gamma_k(\omega, k) \langle \big| \hat{O}^k \rangle$$

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

$$\frac{\partial}{\partial \lambda} G(\lambda; \omega) = G(\omega, \partial \omega) G(\lambda; \omega), \quad G(\omega, \partial \omega) = \left( \langle | e^{ad_{\omega} \hat{G}} \rangle \right)_{\hat{O} \rightarrow \partial \omega}.$$
Running example: **planar rooted binary trees (PRBTs)**

Notational convention: \[ \equiv \begin{array}{c} I \ \ \ \ \ \ \ \ L \ \ \ \ \ \ \ \ R \end{array} \]

### The Rémy uniform generator in the rule-algebra formalism

\[
\hat{G} := \hat{G}_L + \hat{G}_R, \quad \hat{G}_L := \sum_{T \in \{I,L,R\}} \rho \left( \delta \left( \begin{array}{c} L \\ T \\ R \end{array} \right) \mapsto \begin{array}{c} I \\ L \\ R \end{array} \ ; \ Shift \left( \emptyset \mapsto \begin{array}{c} T \\ c_{PBRT} \end{array} \right) \right) \]

\[
\hat{G}_R := \sum_{T \in \{I,L,R\}} \rho \left( \delta \left( \begin{array}{c} L \\ T \\ R \end{array} \right) \mapsto \begin{array}{c} I \\ L \end{array} \ ; \ Shift \left( \emptyset \mapsto \begin{array}{c} T \\ c_{PBRT} \end{array} \right) \right) \]

\[
\hat{G} \left| \frac{}{} \right\rangle = \sum_{t \in \mathcal{T}_1} 2! \left| t \right\rangle, \quad \forall t \in \mathcal{T}_1 : \hat{G} \left| t \right\rangle = \sum_{t' \in \mathcal{T}_2} 3! \left| t' \right\rangle, \ldots, \quad \forall t \in \mathcal{T}_n : \hat{G} \left| t \right\rangle = \sum_{t' \in \mathcal{T}_{n+1}} (n+2)! \left| t' \right\rangle
\]
Running example: planar rooted binary trees (PRBTs)

\[
\hat{O}_E := \sum_{T \in \{I,L,R\}} \rho \left( \delta \left( \bullet_T \leftrightarrow \bullet_T \leftrightarrow \bullet_T ; \text{true} \right) \right)
\]
Running example: **planar rooted binary trees (PRBTs)**

\[
\hat{O}_E := | := \sum_{T \in \{I,L,R\}} \rho \left( \delta \left( \begin{array}{c} \vdots \\ \vdots \\ \vdots \end{array} \right) \right)
\]

\[
\left[ \hat{O}_E, \hat{G} \right] = \left[ | + \setminus \div \plus, \begin{array}{c} *Y + *Y \\ *Y + *Y \end{array} \right] = *Y + *Y + *Y + *Y + \ldots \ldots = 2\hat{G}
\]
Running example: planar rooted binary trees (PRBTs)

\[
\hat{O}_E := \sum_{T \in \{I,L,R\}} \rho \left( \delta \left( \begin{array}{c}
  \vdots \\
  T \\
  \vdots 
\end{array} \right) \leftarrow \begin{array}{c}
  \vdots \\
  T \\
  \vdots 
\end{array} \leftarrow \begin{array}{c}
  \vdots \\
  T \\
  \vdots 
\end{array} ; \text{true} \right) \right)
\]

\[
[\hat{O}_E, \hat{G}] = \left[ \begin{array}{c}
  + \backslash \backslash / , \ Y + Y \ Y \end{array} \right] = \begin{array}{c}
  Y \ Y + Y \ Y + Y \ Y + Y \ Y + \ldots \ldots \end{array} = 2\hat{G}
\]

\[\langle | \hat{G} = 2 \langle | \hat{O}_E \]

Nicolas Behr, SOCS 2020, IRIF, December 11, 2020
Running example: planar rooted binary trees (PRBTs)

\[ \hat{O}_E := \sum_{T \in \{I,L,R\}} \rho \left( \delta \left( \begin{array}{c} \begin{array}{c} \uparrow \downarrow \end{array} \end{array} \right) \right) \]

\[ [\hat{O}_E, \hat{G}] = \left[ \begin{array}{c} \begin{array}{c} + \setminus + / \end{array}, \begin{array}{c} Y + Y \end{array} \end{array} \right] = Y + Y + Y + Y + Y + \ldots - \ldots = 2\hat{G} \]

\[ \langle \hat{G} \rangle = 2 \langle \hat{O}_E \rangle \]

\[
\left\{
\begin{array}{l}
\frac{\partial}{\partial \lambda} \mathcal{G}(\lambda; \varepsilon) = 2e^{2\varepsilon} \frac{\partial}{\partial \varepsilon} \mathcal{G}(\lambda; \varepsilon) \\
\mathcal{G}(0; \varepsilon) = \langle e^{\varepsilon \hat{O}_E} \rangle = e^{\varepsilon}
\end{array}
\right.

\Rightarrow \quad \mathcal{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)

\[\hat{O}_{P1} := \sum_{T \in \{I.L,R\}} T, \quad \hat{O}_{P2} := \sum_{T \in \{I.L,R\}} T, \quad \hat{O}_{P3} := \sum_{T \in \{I.L,R\}} T\]
Running example: planar rooted binary trees (PRBTs)

\[
\hat{O}_1 := \sum_{T \in \{I.L.R\}} \hat{O}_1, \quad \hat{O}_2 := \sum_{T \in \{I.L.R\}} \hat{O}_2, \quad \hat{O}_3 := \sum_{T \in \{I.L.R\}} \hat{O}_3
\]

\[
[\hat{O}_2, \hat{G}] = \hat{O}_2 + \hat{O}_2 - \hat{O}_2 - \hat{O}_2
\]

\[
[\hat{O}_3, \hat{G}] = \hat{O}_3 + \hat{O}_3 + \hat{O}_3 + \hat{O}_3 - \hat{O}_3 - \hat{O}_3 - \hat{O}_3 - \hat{O}_3
\]

\[
[\hat{O}_2, [\hat{O}_2, \hat{G}]] = [\hat{O}_2, \hat{G}], \quad [\hat{O}_2, [\hat{O}_3, \hat{G}]] = [\hat{O}_3, \hat{G}] + \hat{R}_3
\]

\[
[\hat{O}_3, [\hat{O}_3, \hat{G}]] = [\hat{O}_3, \hat{G}] + 2\hat{R}_3 \quad [\hat{O}_2, \hat{R}_3] = 0, \quad [\hat{O}_3, \hat{R}_3] = -\hat{R}_3
\]

\[
\langle [\hat{O}_2, \hat{G}] \rangle = \langle (3\hat{O}_1 - 2\hat{O}_2) \rangle, \quad \langle [\hat{O}_3, \hat{G}] \rangle = \langle (4\hat{O}_2 - 3\hat{O}_3) \rangle, \quad \langle \hat{R}_3 \rangle = \langle \hat{O}_3 \rangle
\]
Running example: planar rooted binary trees (PRBTs)

\[ \hat{O}_{P1} := \sum_{T \in \{I,L,R\}} \hat{\varepsilon}_T, \quad \hat{O}_{P2} := \sum_{T \in \{I,L,R\}} \hat{\mu}_T, \quad \hat{O}_{P3} := \sum_{T \in \{I,L,R\}} \hat{\nu}_T \]

\[ \mathcal{G}(\lambda; \omega) := \langle e^{\omega \hat{\phi}} e^{\lambda \hat{G}} \rangle, \quad \omega \cdot \hat{\phi} := \varepsilon \hat{O}_E + \gamma \hat{O}_{P1} + \mu \hat{O}_{P2} + \nu \hat{O}_{P3} \]

\[ \frac{\partial}{\partial \lambda} \mathcal{G}(\lambda; \omega) = \langle (e^{ad_{\omega \hat{\phi}}} \hat{G}) e^{\omega \hat{\phi}} e^{\lambda \hat{G}} \rangle = \langle \left( e^{ad_{\varepsilon \hat{O}_E}} e^{ad_{\gamma \hat{O}_{P1}}} \right) e^{ad_{\mu \hat{O}_{P2}}} e^{ad_{\nu \hat{O}_{P3}}} e^{\omega \hat{\phi}} e^{\lambda \hat{G}} \rangle \]

\[ = e^{2\varepsilon + \gamma} \langle \left( e^{ad_{\varepsilon \hat{O}_E}} \hat{G} + (\varepsilon - 1)[\hat{O}_{P2}, \hat{G}] + e^{\mu} (e^{\nu} - 1)[\hat{O}_{P3}, \hat{G}] + (e^{\mu} - e^{\nu}) \hat{R}_{P3} \right) e^{\omega \hat{\phi}} e^{\lambda \hat{G}} \rangle \]

\[ = e^{2\varepsilon + \gamma} \langle (2 \hat{O}_E + 3(\nu + 1) \hat{O}_{P1} + (4\varepsilon + (4\varepsilon + 1) \hat{O}_{P2} + (3\varepsilon + 3\varepsilon + 1) \hat{O}_{P3} e^{\omega \hat{\phi}} e^{\lambda \hat{G}} \rangle \]

\[ + e^{\mu} (e^{\nu} - 1)[\hat{O}_{P3}, \hat{G}] + (e^{\mu} - e^{\nu}) \hat{R}_{P3} \rangle e^{\omega \hat{\phi}} e^{\lambda \hat{G}} \rangle \]

\[ = e^{2\varepsilon + \gamma} \langle (2 \frac{\partial}{\partial \varepsilon} + 3(\mu - 1) \frac{\partial}{\partial \lambda} + (4\mu + 1) \hat{O}_{P2} + (3\mu + 3\mu + 1) \hat{O}_{P3} e^{\omega \hat{\phi}} e^{\lambda \hat{G}} \rangle \]

\[ + (3\mu + 3\mu + 1) \hat{O}_{P3} e^{\omega \hat{\phi}} e^{\lambda \hat{G}} \rangle \]

Nicolas Behr, SOCS 2020, IRIF, December 11, 2020
\[ P_{2l} := \begin{array}{c}
  * \\
\end{array} \]
\[ P_{3l} := \begin{array}{c}
  * \\
\end{array} \]

\[ \mathcal{T}_0 = \{ \ | \ \} \]
$P_{2I} := \begin{array}{c}
\star \\
\end{array}$

$P_{3I} := \begin{array}{c}
\star \\
\end{array}$

$\mathcal{F}_1 = \{Y\}$
$P_{2l} := \begin{array}{c}
\begin{array}{c}
\quad \\
\quad \\
\quad
\end{array}
\end{array}
\quad *
$

$P_{3l} := \begin{array}{c}
\begin{array}{c}
\quad \\
\quad \\
\quad
\end{array}
\end{array}
\quad *
$

$\mathcal{T}_2 = \left\{ \begin{array}{c}
\quad \\
\quad \\
\quad
\end{array}, \begin{array}{c}
\quad \\
\quad \\
\quad
\end{array}, \begin{array}{c}
\quad \\
\quad \\
\quad
\end{array} \right\}$
\[ P_{2l} := \text{\includegraphics[width=0.2\textwidth]{tree1}} \]

\[ P_{3l} := \text{\includegraphics[width=0.2\textwidth]{tree2}} \]

\[ \mathcal{T}_3 = \{ \text{\includegraphics[width=0.2\textwidth]{tree3}}, \text{\includegraphics[width=0.2\textwidth]{tree4}}, \text{\includegraphics[width=0.2\textwidth]{tree5}}, \text{\includegraphics[width=0.2\textwidth]{tree6}} \} \]
$P_{2l} := \begin{array}{c}
\begin{tikzpicture}
\node (l1) at (0,0) {*};
\node (l2) at (1,0) {\star};
\node (l3) at (2,0) {\star};
\end{tikzpicture}
\end{array}$

$P_{3l} := \begin{array}{c}
\begin{tikzpicture}
\node (l1) at (0,0) {*};
\node (l2) at (1,0) {*};
\node (l3) at (2,0) {*};
\end{tikzpicture}
\end{array}$

$\mathcal{T}_3 = \{ \begin{array}{c}
\begin{tikzpicture}
\node (l1) at (0,0) {\star};
\node (l2) at (1,0) {\star};
\node (l3) at (2,0) {\star};
\end{tikzpicture}
, \begin{tikzpicture}
\node (l1) at (0,0) {\star};
\node (l2) at (1,0) {\star};
\node (l3) at (2,0) {\star};
\end{tikzpicture}
, \begin{tikzpicture}
\node (l1) at (0,0) {\star};
\node (l2) at (1,0) {\star};
\node (l3) at (2,0) {\star};
\end{tikzpicture}
, \begin{tikzpicture}
\node (l1) at (0,0) {\star};
\node (l2) at (1,0) {\star};
\node (l3) at (2,0) {\star};
\end{tikzpicture}
, \begin{tikzpicture}
\node (l1) at (0,0) {\star};
\node (l2) at (1,0) {\star};
\node (l3) at (2,0) {\star};
\end{tikzpicture}
, \begin{tikzpicture}
\node (l1) at (0,0) {\star};
\node (l2) at (1,0) {\star};
\node (l3) at (2,0) {\star};
\end{tikzpicture}
, \begin{tikzpicture}
\node (l1) at (0,0) {\star};
\node (l2) at (1,0) {\star};
\node (l3) at (2,0) {\star};
\end{tikzpicture}
\end{array} \}$
\[ P_{2l} := \]
\[ P_{3l} := \]

\[ \mathcal{T}_3 = \{ \} \]
$P_{2l} := \begin{tikzpicture}
\end{tikzpicture}$

$P_{3l} := \begin{tikzpicture}
\end{tikzpicture}$

$\mathcal{T}_3 = \{ \begin{tikzpicture}
\end{tikzpicture}, \begin{tikzpicture}
\end{tikzpicture}, \begin{tikzpicture}
\end{tikzpicture}, \begin{tikzpicture}
\end{tikzpicture}, \begin{tikzpicture}
\end{tikzpicture} \}$
$P_{2l} := \begin{array}{c}
\end{array}$

$P_{3l} := \begin{array}{c}
\end{array}$

$\mathcal{T}_3 = \{ \begin{array}{c}
\end{array} \}$
$P_{2l} := \begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\ast
\end{array}
\end{array}
\end{array}$

$P_{3l} := \begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\ast
\end{array}
\end{array}
\end{array}$

$\mathcal{T}_3 = \{\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\ast
\end{array}
\end{array}
\end{array},
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\ast
\end{array}
\end{array}
\end{array},
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\ast
\end{array}
\end{array}
\end{array},
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\ast
\end{array}
\end{array}
\end{array},
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\ast
\end{array}
\end{array}
\end{array}\}
\[ P_{2I} := \]

\[ P_{3I} := \]
\[ P_{2I} := \]

\[ P_{3I} := \]
$P_{2l} := \begin{array}{c}
\star \\
\star \\
\star 
\end{array}$

$P_{3l} := \begin{array}{c}
\star \\
\star \\
\star 
\end{array}$

$|\mathcal{T}_{100}| = \frac{200!}{100!} \approx 10^{217}$
$P_{2l} := \begin{array}{c}
\star \\
\end{array}$

$P_{3l} := \begin{array}{c}
\star \\
\end{array}$
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 automati[ed 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 !