Accretive Computation of Global Transformations

A. Fernandez L. Maignan A. Spicher

Univ Paris Est Creteil, LACL, 94000, Creteil, France
firstname.lastname@u-pec.fr

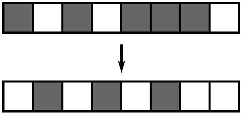
RAMICS 2021

Context

- Global Transformations (GT): general frame of work
  - Describe local, deterministic and synchronous systems

Context

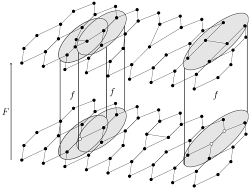
- Global Transformations (GT): general frame of work
  - Describe local, deterministic and synchronous systems
  - Cellular automata



abbab

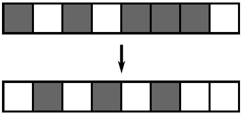
$\Downarrow$

bababbab

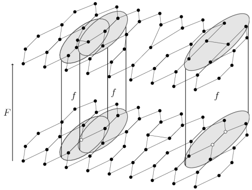


Context

- Global Transformations (GT): general frame of work
  - Describe local, deterministic and synchronous systems
  - Cellular automata, L-systems

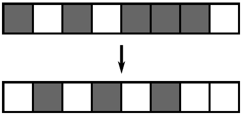


abbab
⇓
bababbab



Context

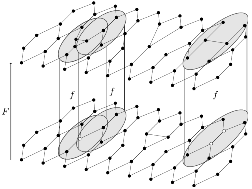
- Global Transformations (GT): general frame of work
  - Describe local, deterministic and synchronous systems
  - Cellular automata, L-systems, causal graph dynamics



abbab

$\Downarrow$

bababbab

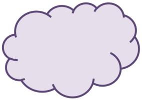


Context

- Global Transformations (GT): general frame of work
  - Describe local, deterministic and synchronous systems
  - Cellular automata, L-systems, causal graph dynamics
- Different structure / space

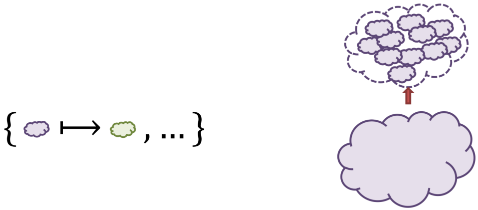
Context

- Global Transformations (GT): general frame of work
  - Describe local, deterministic and synchronous systems
  - Cellular automata, L-systems, causal graph dynamics
- Different structure / space
- Same rewriting procedure



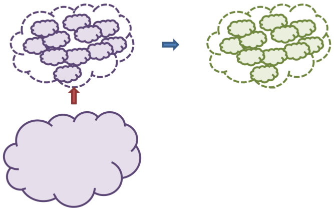
Context

- Global Transformations (GT): general frame of work
  - Describe local, deterministic and synchronous systems
  - Cellular automata, L-systems, causal graph dynamics
- Different structure / space
- Same rewriting procedure
  - pattern matching



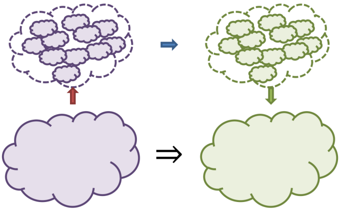
Context

- Global Transformations (GT): general frame of work
  - Describe local, deterministic and synchronous systems
  - Cellular automata, L-systems, causal graph dynamics
- Different structure / space
- Same rewriting procedure
  - pattern matching
  - Local application



Context

- Global Transformations (GT): general frame of work
  - Describe local, deterministic and synchronous systems
  - Cellular automata, L-systems, causal graph dynamics
- Different structure / space
- Same rewriting procedure
  - pattern matching
  - Local application
  - Result reconstruction

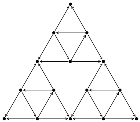


Scope of this work

Define global transformations of presheaves

- Presheaves: generalized graphs
- This presentation: focus only on graphs

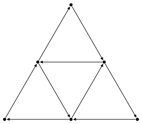
Give an online algorithm for computing GT:

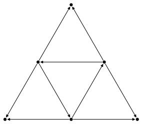


Scope of this work

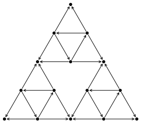
Define global transformations of presheaves

- Presheaves: generalized graphs
- This presentation: focus only on graphs





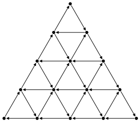
Give an online algorithm for computing GT:

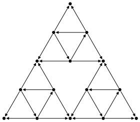


Scope of this work

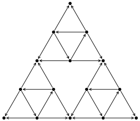
Define global transformations of presheaves

- Presheaves: generalized graphs
- This presentation: focus only on graphs





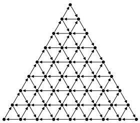
Give an online algorithm for computing GT:

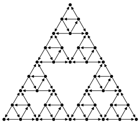


Scope of this work

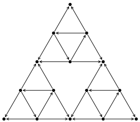
Define global transformations of presheaves

- Presheaves: generalized graphs
- This presentation: focus only on graphs





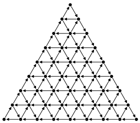
Give an online algorithm for computing GT:

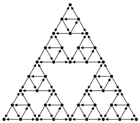


Scope of this work

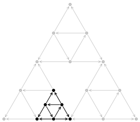
Define global transformations of presheaves

- Presheaves: generalized graphs
- This presentation: focus only on graphs





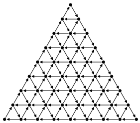
Give an online algorithm for computing GT:

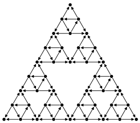


Scope of this work

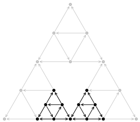
Define global transformations of presheaves

- Presheaves: generalized graphs
- This presentation: focus only on graphs





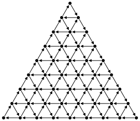
Give an online algorithm for computing GT:

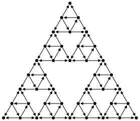


Scope of this work

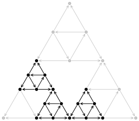
Define global transformations of presheaves

- Presheaves: generalized graphs
- This presentation: focus only on graphs





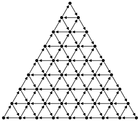
Give an online algorithm for computing GT:

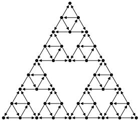


Scope of this work

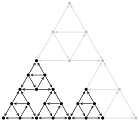
Define global transformations of presheaves

- Presheaves: generalized graphs
- This presentation: focus only on graphs





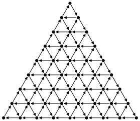
Give an online algorithm for computing GT:

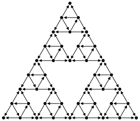


Scope of this work

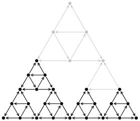
Define global transformations of presheaves

- Presheaves: generalized graphs
- This presentation: focus only on graphs





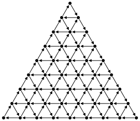
Give an online algorithm for computing GT:

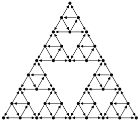


Scope of this work

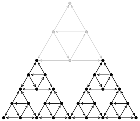
Define global transformations of presheaves

- Presheaves: generalized graphs
- This presentation: focus only on graphs





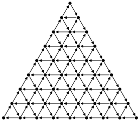
Give an online algorithm for computing GT:

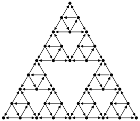


Scope of this work

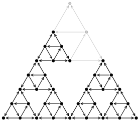
Define global transformations of presheaves

- Presheaves: generalized graphs
- This presentation: focus only on graphs





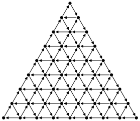
Give an online algorithm for computing GT:

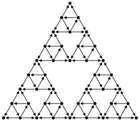


Scope of this work

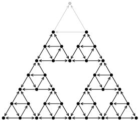
Define global transformations of presheaves

- Presheaves: generalized graphs
- This presentation: focus only on graphs





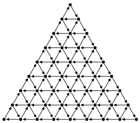
Give an online algorithm for computing GT:

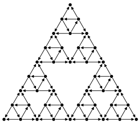


Scope of this work

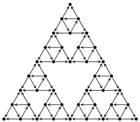
Define global transformations of presheaves

- Presheaves: generalized graphs
- This presentation: focus only on graphs





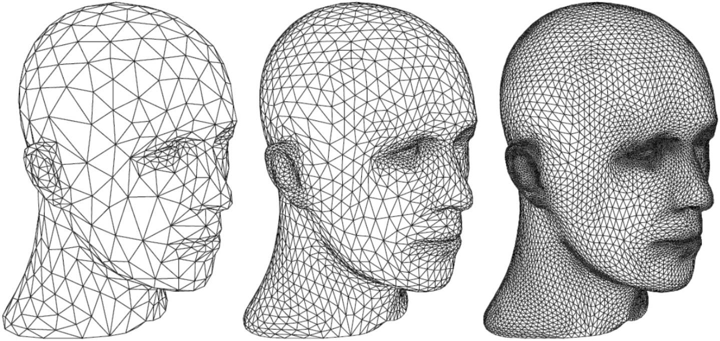
Give an online algorithm for computing GT:



History

History:

- L. M., A. S. **Global** Graph **Transformations**. ICGT 2015
- *Triangular mesh refinement Rule-based Specification*

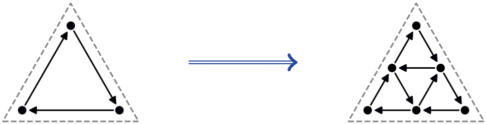


SIGGRAPH 98 Course Notes

History

History:

- L. M., A. S. **Global** Graph **Transformations**. ICGT 2015
- *Triangular mesh refinement Rule-based Specification*

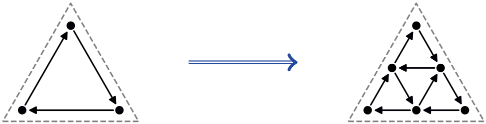


- Simple, but what happens on overlaps ?

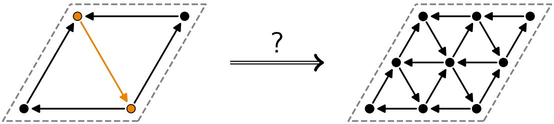
History

History:

- L. M., A. S. **Global** Graph **Transformations**. ICGT 2015
- *Triangular mesh refinement Rule-based Specification*



- Simple, but what happens on overlaps ?

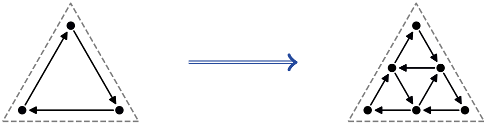


This is what we want !

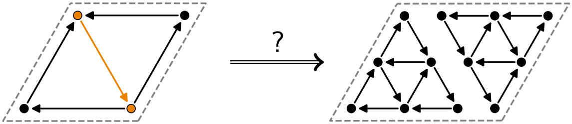
History

History:

- L. M., A. S. **Global** Graph **Transformations**. ICGT 2015
- *Triangular mesh refinement Rule-based Specification*



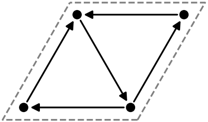
- Simple, but what happens on overlaps ?



Also compliant with the rules...

Solution

How to solve this ambiguity ?

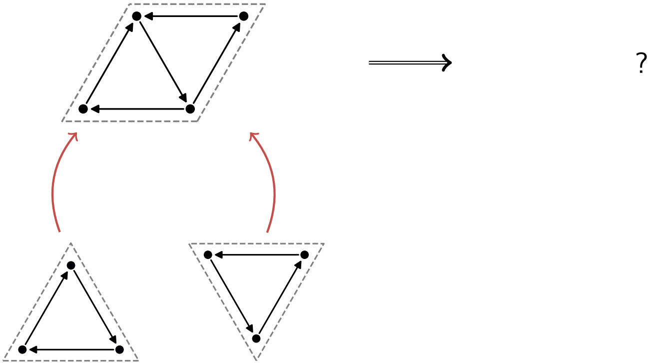


$\Longrightarrow$?

Solution

How to solve this ambiguity ?

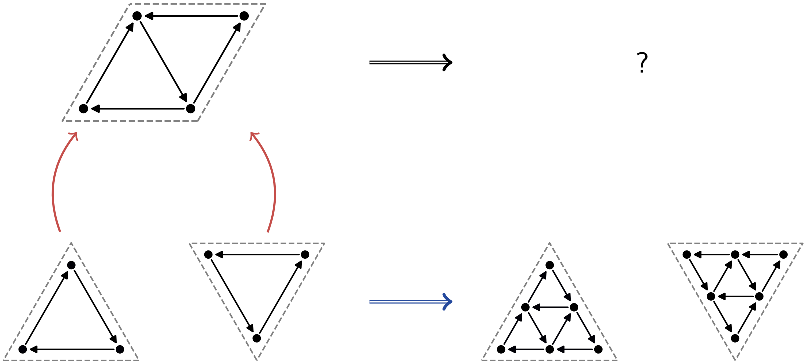
- Pattern-matching



Solution

How to solve this ambiguity ?

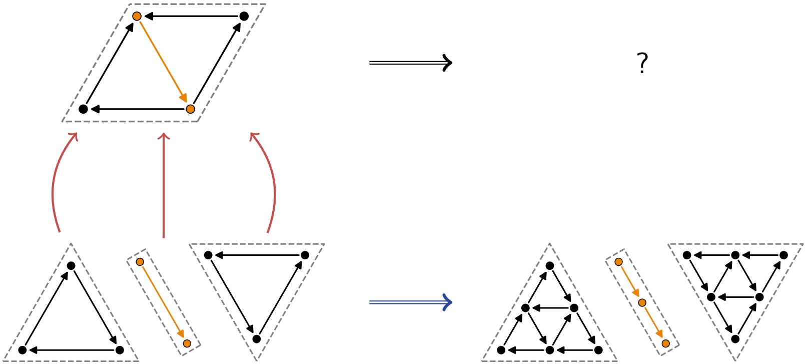
- Pattern-matching ⇒ Local results



Solution

How to solve this ambiguity ?

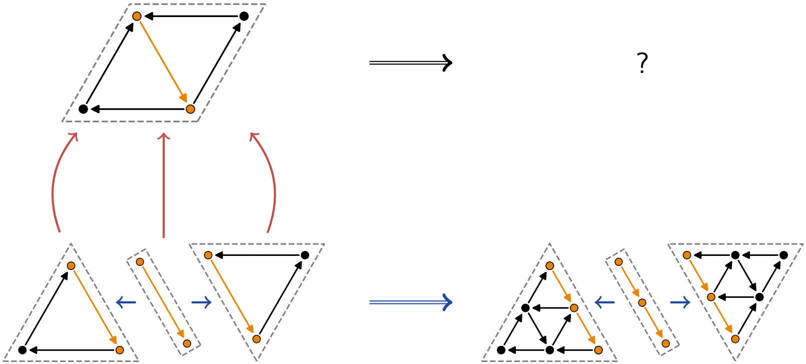
- Pattern-matching ⇒ Local results
- Add a rule



Solution

How to solve this ambiguity ?

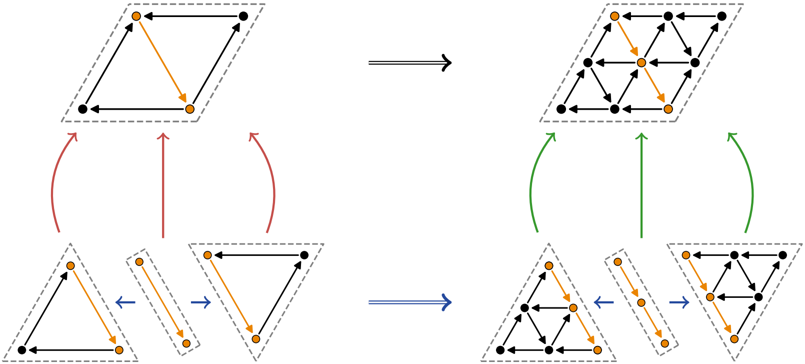
- Pattern-matching ⇒ Local results
- Add a rule + inclusions for overlap



Solution

How to solve this ambiguity ?

- Pattern-matching ⇒ Local results ⇒ Reconstruction
- Add a rule + inclusions for overlap



The category G_M

Structure for "thing being part of other thing " ?

- Preorder

The category G_M

Structure for “thing being part of other thing at some place” ?

- Preorder $\Rightarrow$ Category

The category G_M

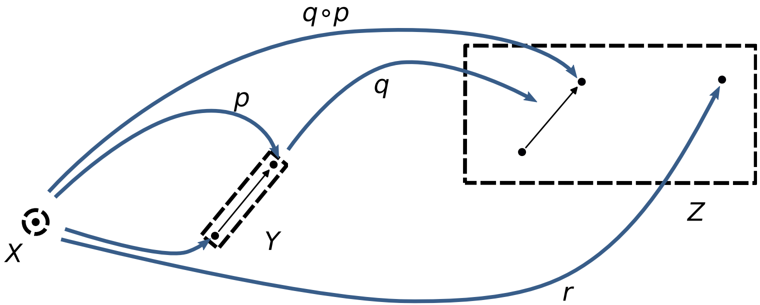
Structure for "thing being part of other thing at some place" ?

- Preorder $\Rightarrow$ Category

Work in G_M: category of graphs with inclusions

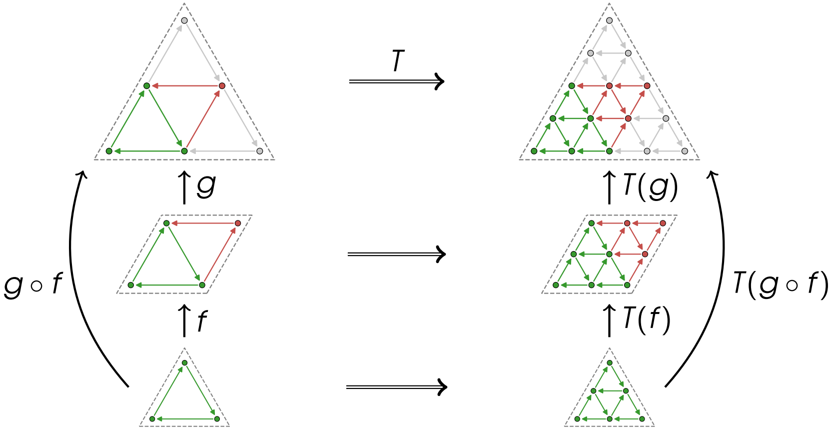
- **Objects**: Graphs (directed multigraphs)
- **Arrows**: Inclusions of graphs (monomorphisms)

$\subset G$: category of graphs with graph homomorphism



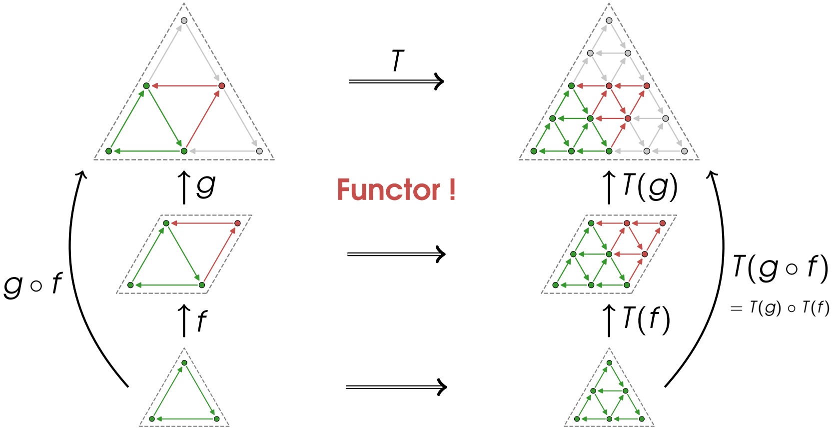
Functor

- Transformation respects arrow ($\simeq$ monotony):
  The way the **local output** occur in the **global output** depends on the way the **local input** occur as a part of the **global input**



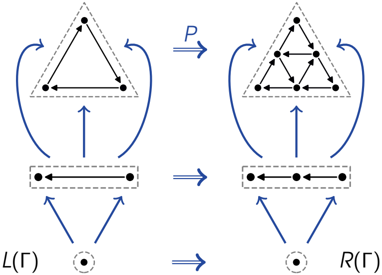
Functor

- Transformation respects arrow ($\simeq$ monotony):
  The way the **local output** occur in the **global output** depends on the way the **local input** occur as a part of the **global input**



Definition: Rule system

A *rule system* T is tuple $\langle \Gamma, L, R \rangle$ where Γ is a category, $L, R : \Gamma \to G_M$ are functors, and L is full and faithfull ($\simeq$ injective).

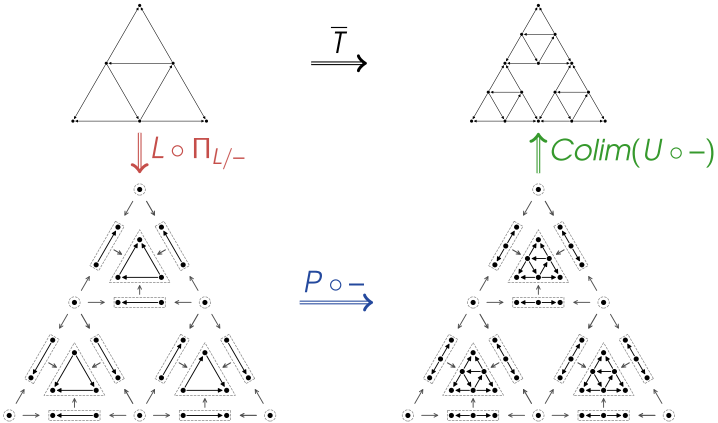


Equivalently: Partial functor $P : G_M \to G_M := R \circ L^{-1}$

Computation

Definition: Global Rewrite Step

The rewrite step is the functor $\overline{T} : G_M \to G$ given by:
$\overline{T}(-) = Colim(U \circ P \circ L \circ \Pi_{L/-})$, with $U : G_M \to G$ forgetful functor.



Global Transformation

No colimits in $G_M \dots$

- Taking colimits in G gives $\overline{T} : G_M \rightarrow G$
- Rule systems do not preserve injectivity !

Definition: Global Transformation

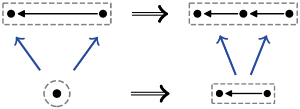
A global transformation T is a rule system such that $\overline{T} : G_M \rightarrow G$ factors through $U : G_M \rightarrow G$. In this case $T : G_M \rightarrow G_M$ is the functor such that $U \circ T = \overline{T}$.

Global transformation: $T : G_M \rightarrow G_M$ is iterable !

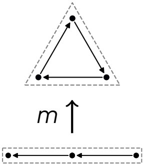
- Triangular mesh refinement, sierpinsky ...

Not global transformations

Consider this rule system:

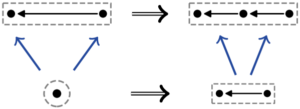


And the inclusion of graphs *m*:

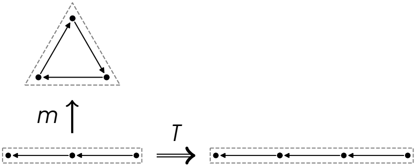


Not global transformations

Consider this rule system:

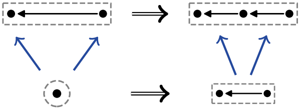


And the inclusion of graphs *m*:

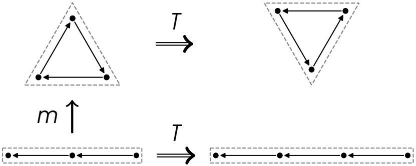


Not global transformations

Consider this rule system:

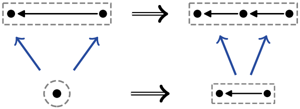


And the inclusion of graphs *m*:

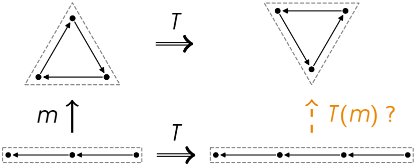


Not global transformations

Consider this rule system:



And the inclusion of graphs m:



No inclusion $T(m)$: T not a global transformation !

Problem: Find a local way to check if rule system is GT ?
(ie. by just looking at the rules)

Computing with GT

Define generic algorithm:

- Big operations $\Rightarrow$ small generic operations
- Breaks down global step $\Rightarrow$ sequence of local steps

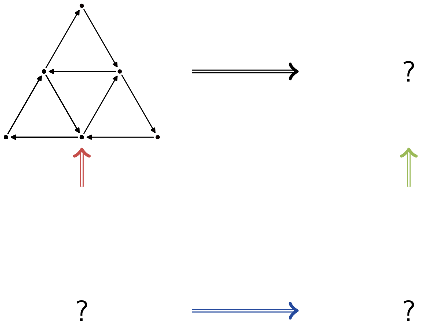
Preserving injectivity for sequence of local steps ?

- Link with global transformation ?

We restrict to connected finite graphs and finite rule systems.

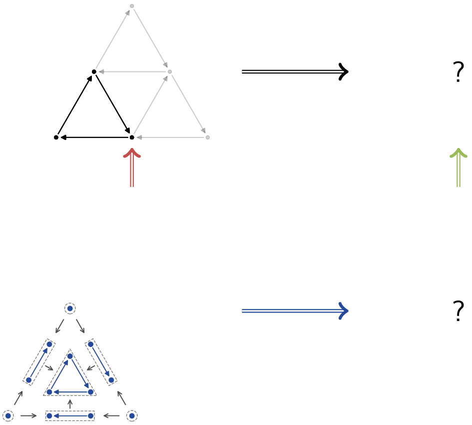
Example run

Online algorithm:



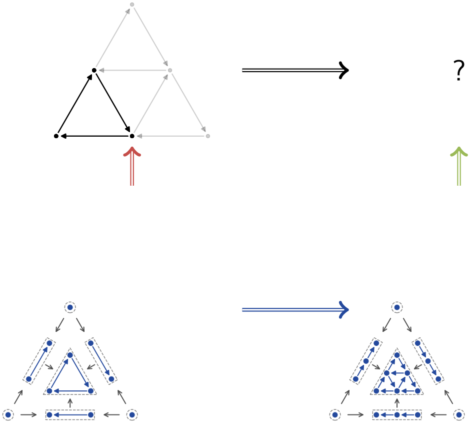
Example run

Online algorithm:



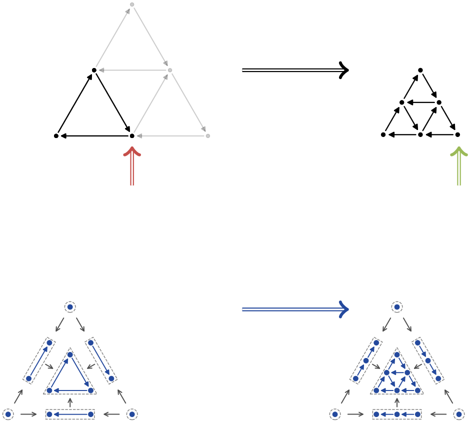
Example run

Online algorithm:

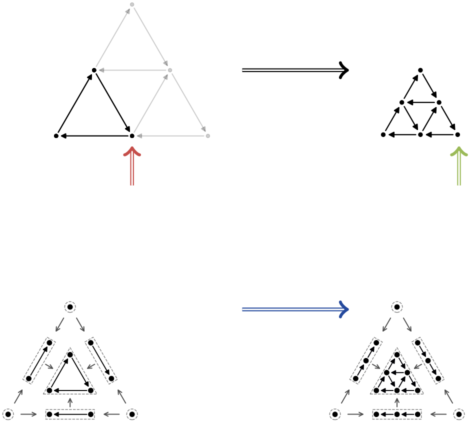


Example run

Online algorithm:

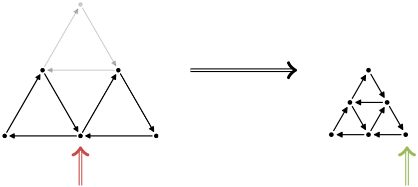


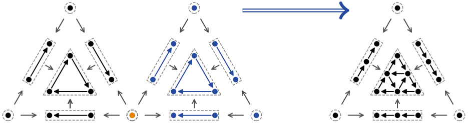
Online algorithm:



Example run

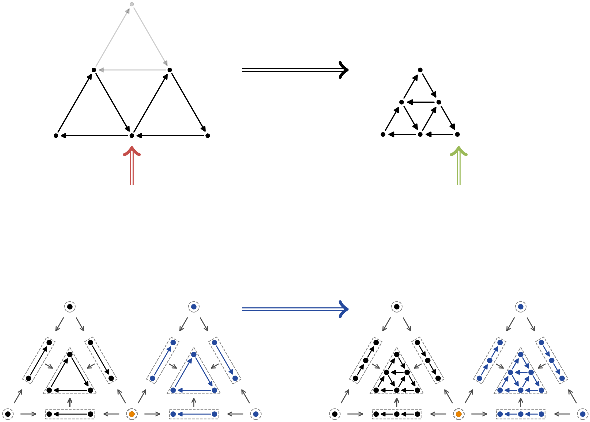
Online algorithm:





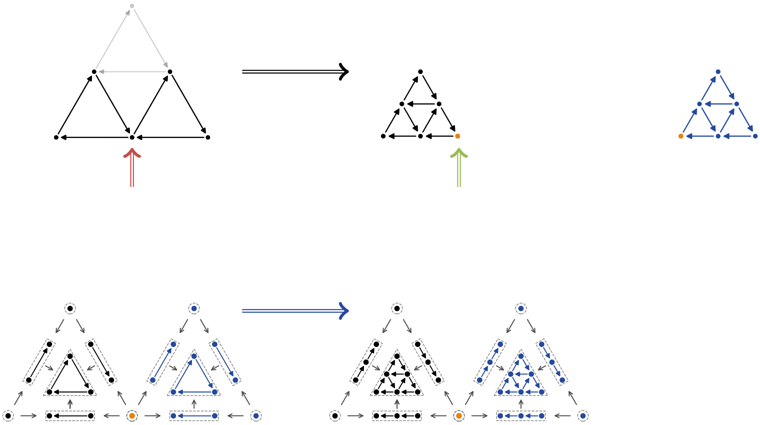
Example run

Online algorithm:

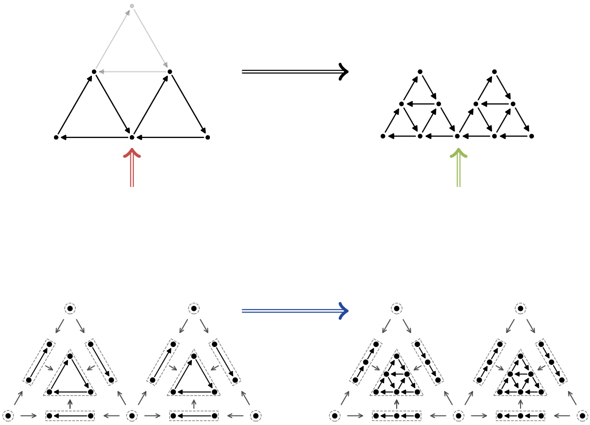


Example run

Online algorithm:

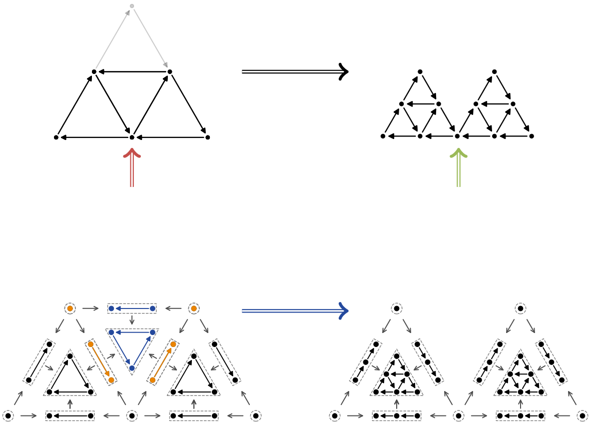


Online algorithm:



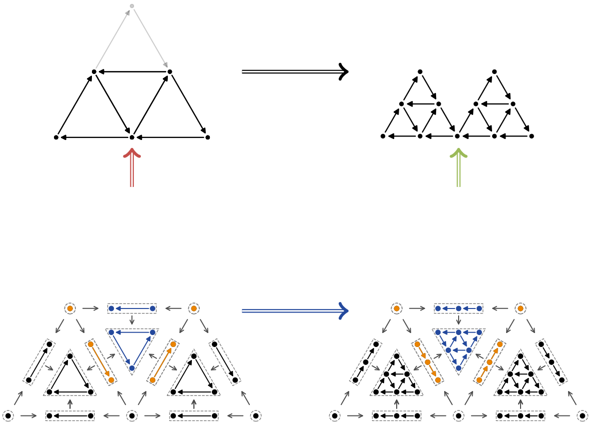
Example run

Online algorithm:

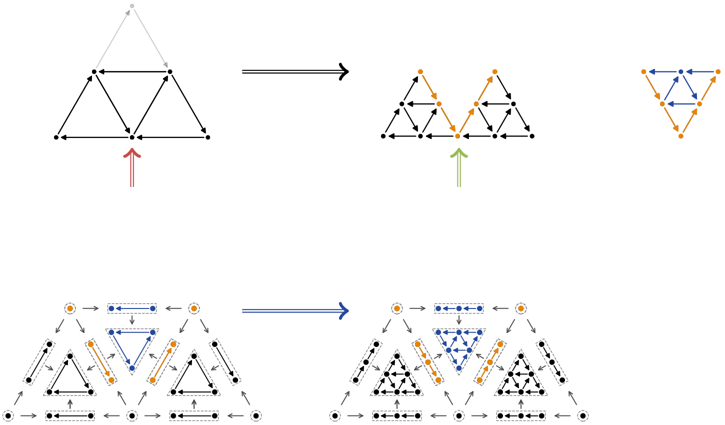


Example run

Online algorithm:

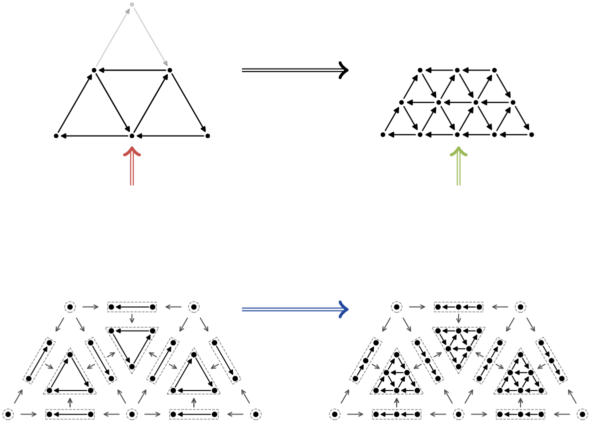


Online algorithm:



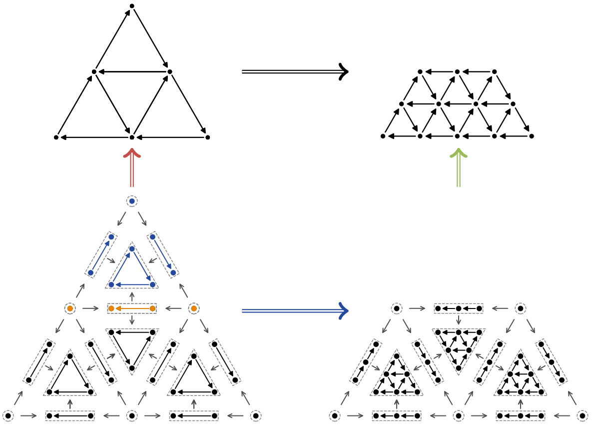
Example run

Online algorithm:

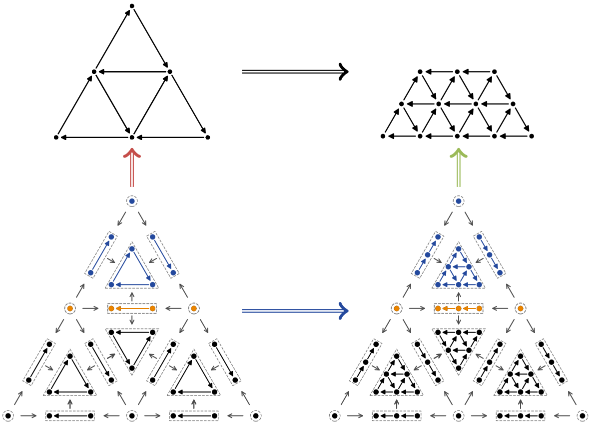


Example run

Online algorithm:

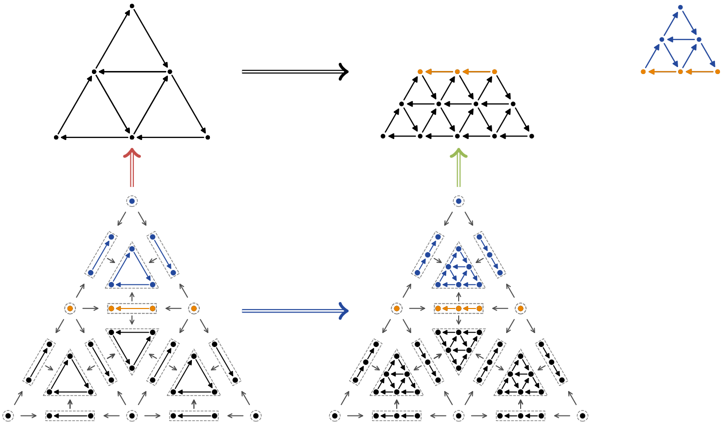


Online algorithm:



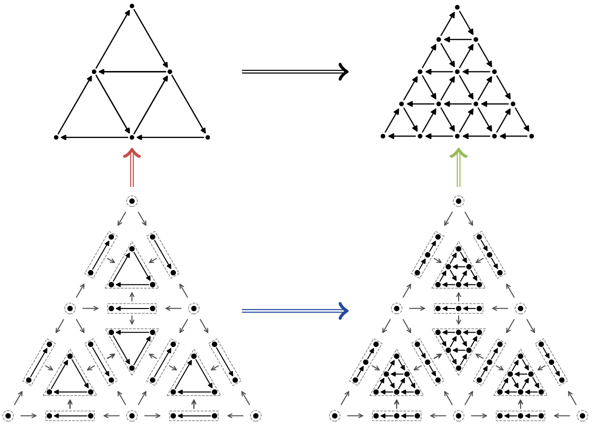
Example run

Online algorithm:



Example run

Online algorithm:



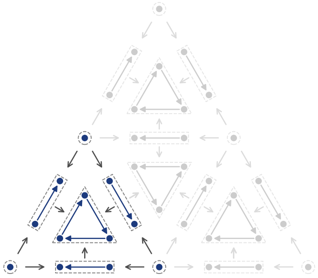
Pattern matching

Proposition

The comma category L/g is isomorphic to a **preorder** (if no automorphism of rule $\Rightarrow$ **poset**).

- Maximal occurences gives the maximal local results
- Other occurences used to glue theses results

Maximal occurences constructed with breadth-first search.



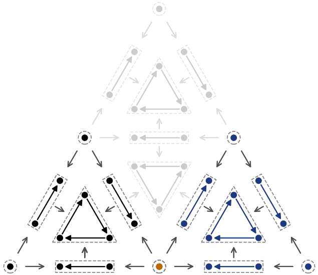
Pattern matching

Proposition

The comma category L/g is isomorphic to a **preorder** (if no automorphism of rule $\Rightarrow$ **poset**).

- Maximal occurences gives the maximal local results
- Other occurences used to glue theses results

Maximal occurences constructed with breadth-first search.



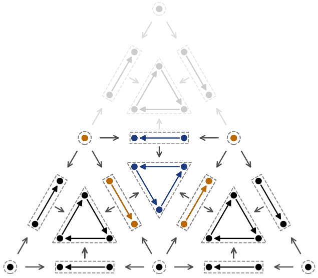
Pattern matching

Proposition

The comma category L/g is isomorphic to a **preorder** (if no automorphism of rule $\Rightarrow$ **poset**).

- Maximal occurences gives the maximal local results
- Other occurences used to glue theses results

Maximal occurences constructed with breadth-first search.



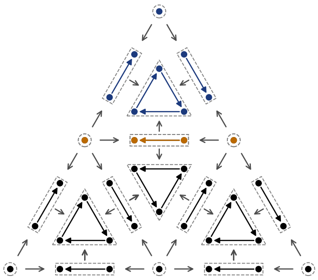
Pattern matching

Proposition

The comma category L/g is isomorphic to a **preorder** (if no automorphism of rule $\Rightarrow$ **poset**).

- Maximal occurences gives the maximal local results
- Other occurences used to glue theses results

Maximal occurences constructed with breadth-first search.



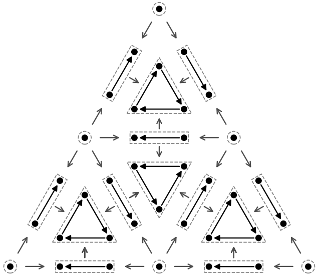
Pattern matching

Proposition

The comma category L/g is isomorphic to a **preorder** (if no automorphism of rule $\Rightarrow$ **poset**).

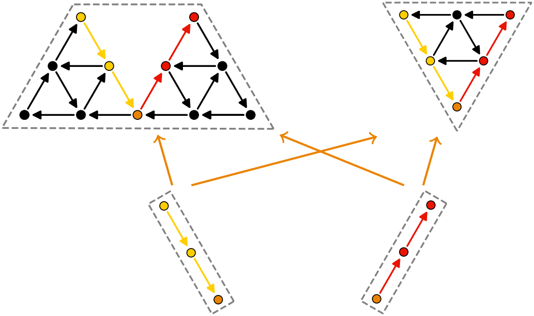
- Maximal occurences gives the maximal local results
- Other occurences used to glue theses results

Maximal occurences constructed with breadth-first search.



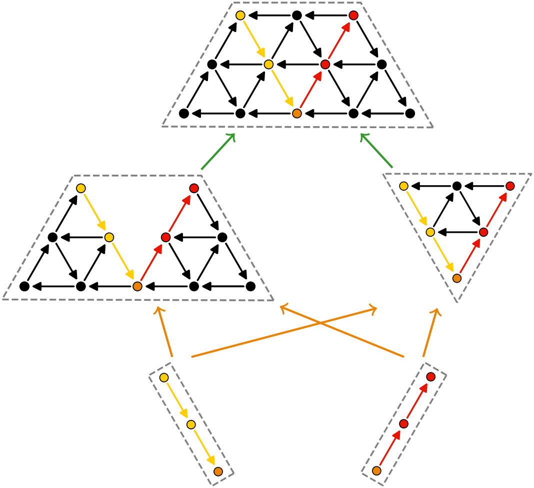
Gluing partial results

Compute gluing along suture:



Gluing partial results

Compute gluing along suture:



Partial computations

Definition: Partial decomposition

Given a graph g, a partial decomposition is a subset M of maximal occurences of L/g such that the restriction $\widetilde{M}$ of L/g to M and morphisms into M is connected.

$\widetilde{L/g}$: category of partial decompositions with set inclusions.

Definition: Partial computation functor

Let $\widetilde{T}_g : \widetilde{L/g} \to G$ given by $T(\widetilde{M}) = Colim(U \circ P \circ L \circ \Pi_{L/g} \restriction \widetilde{M})$.

Definition: Accretive rule system

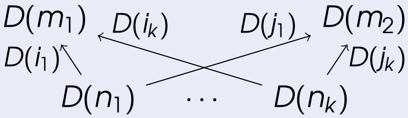
T is *accretive* iff. for any g, $\widetilde{T}_g$ factors through $U : G_M \to G$.

- Accretive $\not\Rightarrow$ Global transformation

Generalized pushout

Definition: suture diagram

A suture diagram $D : I \to G_M$ is a diagram (functor) with the following shape:



The gluing is called a *generalized pushout*:

Definition: generalized pushout

Given a suture diagram $D : I \to G_M$ its generalized pushout is

$$Colim(U \circ D)$$

- Colimit in G: does not preserve injectivity

Accretive global transformations

Pattern matching $\Rightarrow$ sequence $\langle r_1, \ldots, r_k \rangle$ of maximal local results.
Sequence of gluings $\Rightarrow$ *partial results*

$$\begin{array}{ccccccccccccc} r_1 = P_1 & \xrightarrow{i_1} & P_2 & \xrightarrow{i_2} & P_3 & \xrightarrow{i_3} & P_4 & \xrightarrow{i_4} & \cdots & \xrightarrow{\cdots} & P_k \\ & r_2 \overset{j_1}{\nearrow} & & r_3 \overset{j_2}{\nearrow} & & r_4 \overset{j_3}{\nearrow} & & r_5 \overset{j_4}{\nearrow} & \cdots & \nearrow & \end{array}$$

Proposition

Given path $\widetilde{M_1} \subseteq \cdots \subseteq \widetilde{M_k}$ in $\widetilde{L/g}$ where:

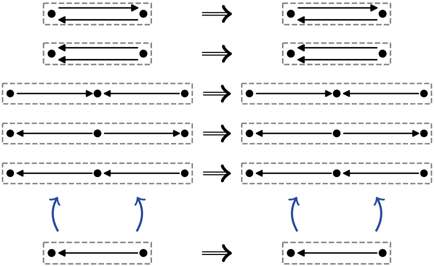
$$\widetilde{M_1} = \{r_1\}, \widetilde{M_{i+1}} = \widetilde{M_i} \cup \{r_{i+1}\}.$$

Then for any $i \in \{1, \ldots, k\}$, $\widetilde{T}_g(\widetilde{M_i}) = P_i$.

- If some gluing is non-injective then T is not acccretive.

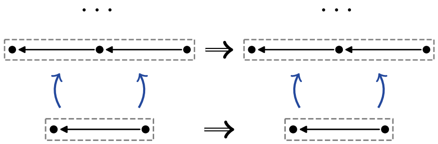
Non-accretive global transformations

Consider the following rule system:



Non-accretive global transformations

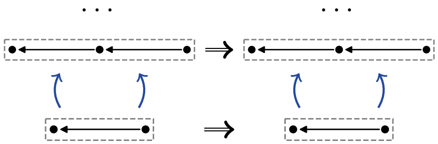
Consider the following rule system:



Removes isolated vertices $\Rightarrow$ Global Transformation

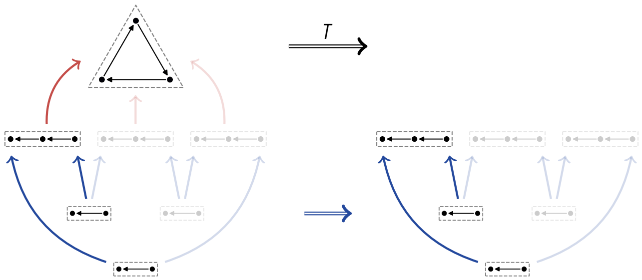
Non-accretive global transformations

Consider the following rule system:



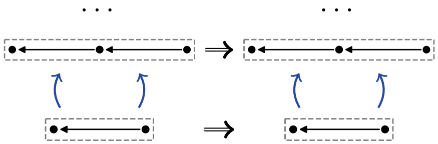
Removes isolated vertices ⇒ Global Transformation

Not accretive :

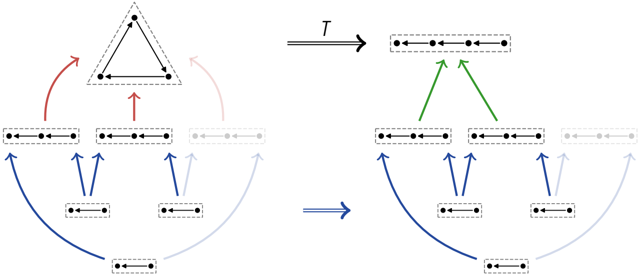


Non-accretive global transformations

Consider the following rule system:

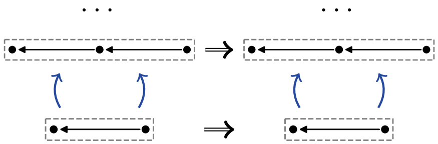


Removes isolated vertices ⇒ Global Transformation
Not accretive :



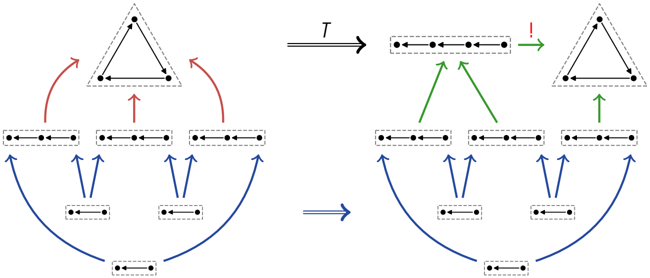
Non-accretive global transformations

Consider the following rule system:



Removes isolated vertices ⇒ Global Transformation

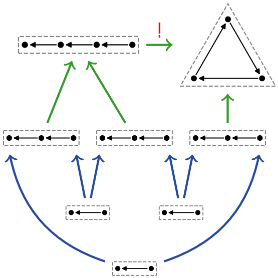
Not accretive :



Issue with last example

Find a criterion on rule systems which implies GT ?

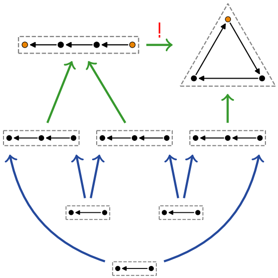
- Criterion on rule systems which implies accretive GT



Issue with last example

Find a criterion on rule systems which implies GT ?

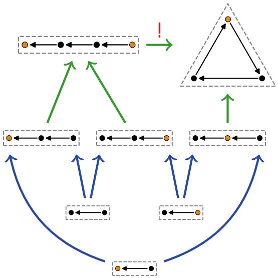
- Criterion on rule systems which implies accretive GT



Issue with last example

Find a criterion on rule systems which implies GT ?

- Criterion on rule systems which implies accretive GT



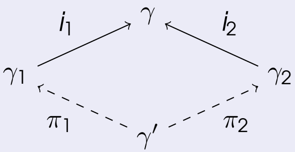
The three local results must be glued at the same time !

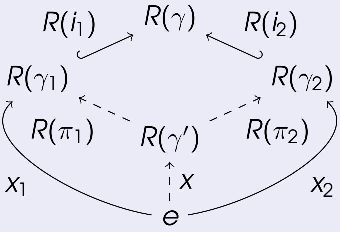
- This gluing is not induced by a suture of two local result !
- Define a criterion which forbids that !

Incrementality

Definition: Incrementality

T is incremental if for any $\gamma_1 \xrightarrow{i_1} \gamma \xleftarrow{i_2} \gamma_2$ in Γ, any graph $e \in \{\, \cdot\, ,\ \cdot \to \cdot \,\}$, and any $R(\gamma_1) \xleftarrow{x_1} e \xrightarrow{x_2} R(\gamma_2)$ such that $R(i_1) \circ x_1 = R(i_2) \circ x_2$, there are $\gamma_1 \xrightarrow{\pi_1} \gamma' \xleftarrow{\pi_2} \gamma_2$ and $x : e \to R(\gamma')$ such that:





Theorem

If a rule-system is incremental, then it is an accretive GT.

Conclusion & Perspectives

Global transformations: Synchronous rewriting

- Works on wide variety of structures

Rewriting algorithm: Sequentialization of GT

- Designed to be generic

Preserving injections:

	non-incr.		incr.	
	non-G.T.	G.T.	non-G.T.	G.T.
non-accretive	ex. Fig. 3a	ex. Fig. 3c	None, Thm. 1/2	None, Thm. 2
accretive	ex. Fig. 3b	ex. Fig. 3d	None, Thm. 1	Sierpenski

Perspectives

Non deterministic computations

- Bicategory of open functors (arxiv)

Algorithm in other categories ?

- Cellular automata, L-Systems, CGD
- ($\mathcal{M}$)-adhesive categories, topos ?

Pattern matching like Knuth-Morris-Pratt

- Break down pattern matching[1]

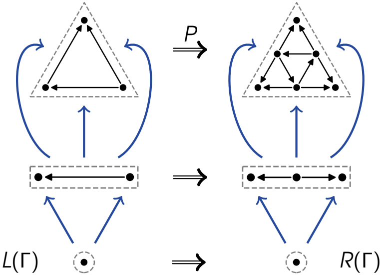
[1] Y. V. Srinivas, A Sheaf-Theoretic approach to pattern matching, 1993

Table of Contents

Rule system

Definition: Rule system

A *rule system* T is tuple $\langle \Gamma, L, R \rangle$ where Γ is a category, $L, R : \Gamma \to G_M$ are functors, and L is full and faithfull ($\simeq$ injective).



Equivalently: Partial functor $P : G_M \to G_M := R \circ L^{-1}$

Non deterministic computations

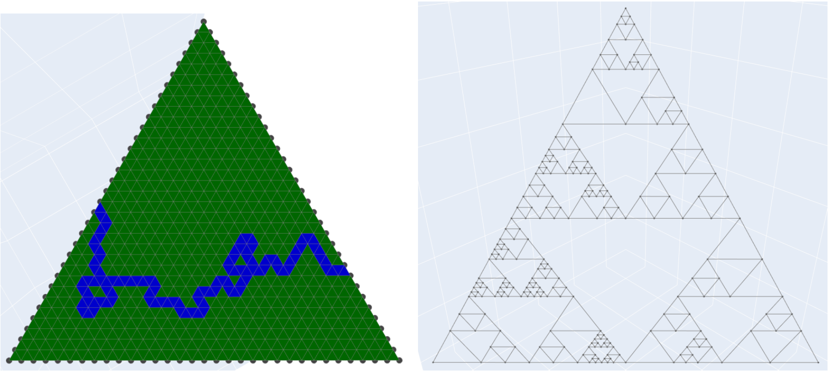
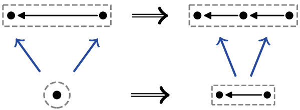


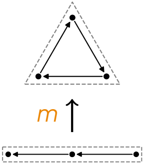
Figure: Left: ND on labels, Right: ND on structure

Not global transformations

Consider this rule system:



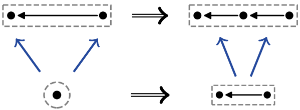
And the inclusion of graphs *m*:



Let's compute:

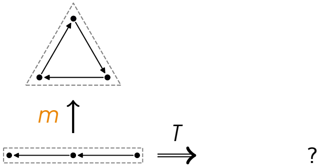
Not global transformations

Consider this rule system:



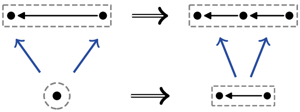
Let's compute:

And the inclusion of graphs *m*:

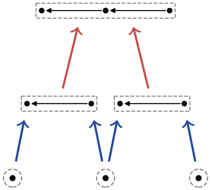


Not global transformations

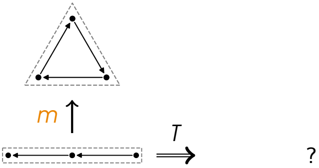
Consider this rule system:



Let's compute:

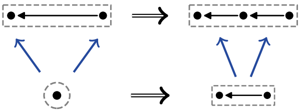


And the inclusion of graphs *m*:

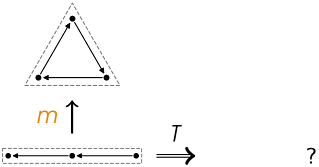


Not global transformations

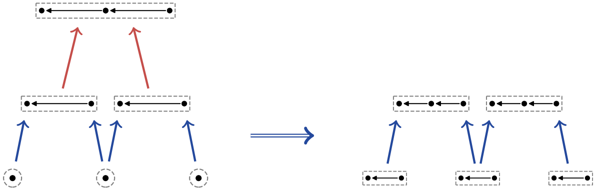
Consider this rule system:



And the inclusion of graphs *m*:

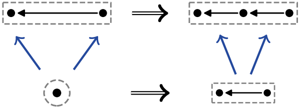


Let's compute:

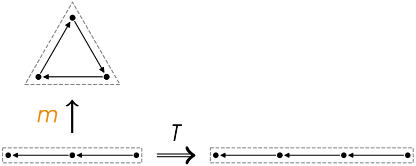


Not global transformations

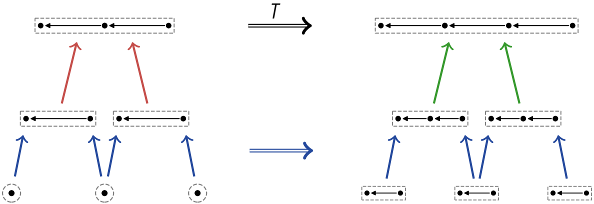
Consider this rule system:



And the inclusion of graphs m:

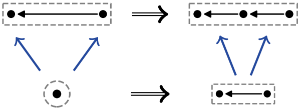


Let's compute:

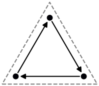


Not global transformations

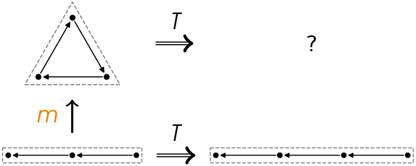
Consider this rule system:



Let's compute:

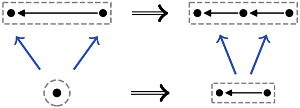


And the inclusion of graphs *m*:

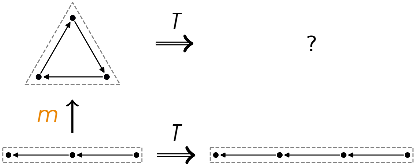


Not global transformations

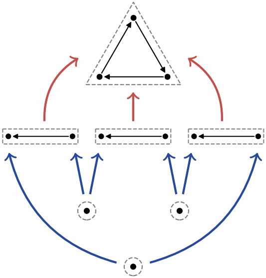
Consider this rule system:



And the inclusion of graphs m:

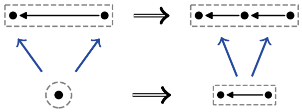


Let's compute:

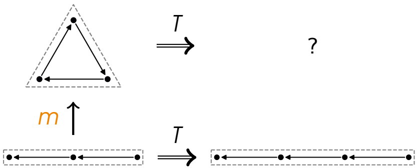


Not global transformations

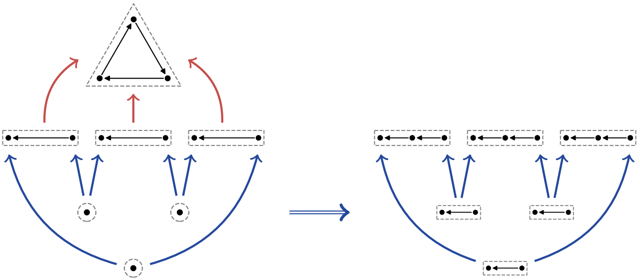
Consider this rule system:



And the inclusion of graphs *m*:

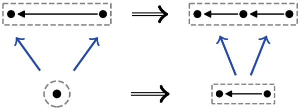


Let's compute:

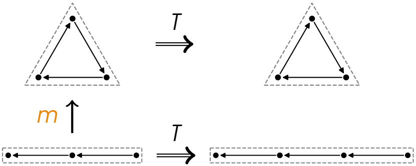


Not global transformations

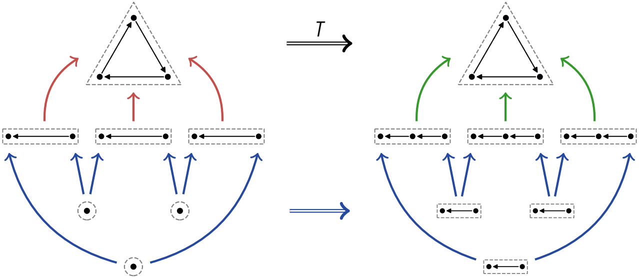
Consider this rule system:



And the inclusion of graphs *m*:

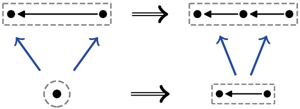


Let´s compute:

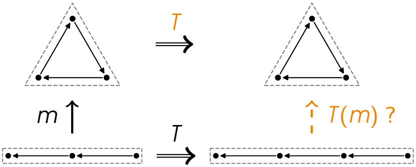


Not global transformations

Consider this rule system:



And the inclusion of graphs m:



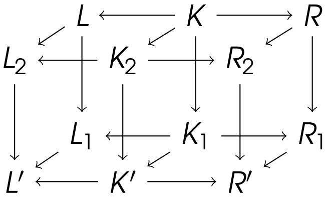
No inclusion $T(m)$: T not a global transformation !

Problem: Find a local way to check if rule system is GT ?
(ie. by just looking at the rules)

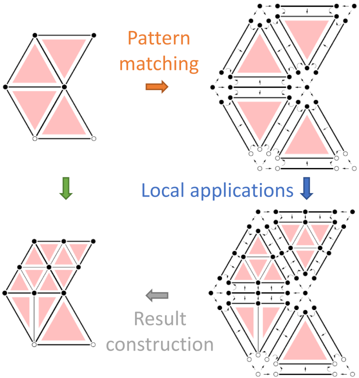
DPO

$$
\begin{array}{ccccc}
L & \longleftarrow & K & \longrightarrow & R \\
\downarrow & & \downarrow & & \downarrow \\
G & \longleftarrow & D & \longrightarrow & H
\end{array}
$$

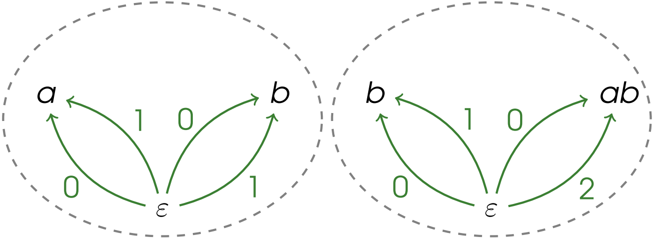
Amalgamation



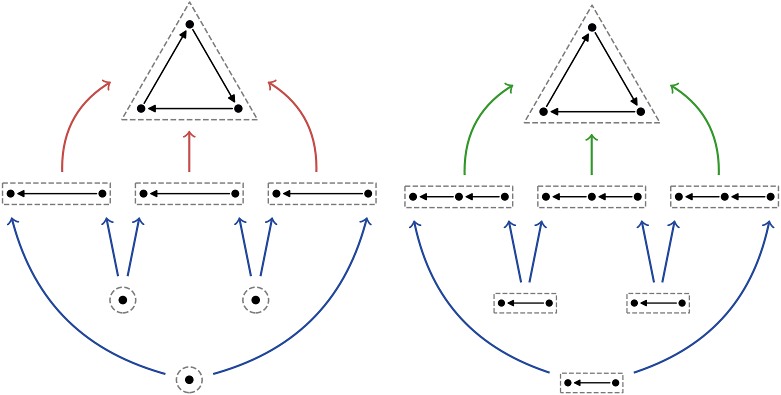
Global transformations



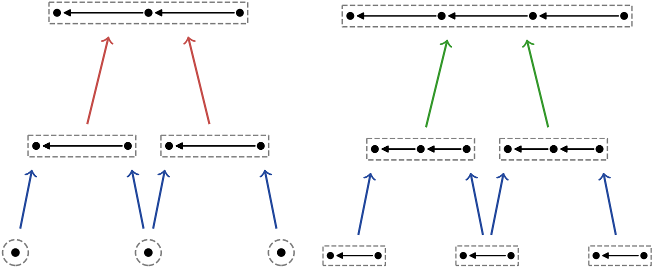
Rules



Not global transformations



Not global transformation 2



Example: L-systems

Definition

A (deterministic context-free) L-system $L : \Sigma^\star \to \Sigma^\star$ is a function given by:

- an alphabet Σ and
- a function $P : \Sigma \to \Sigma^\star$ such that
- $L(a_0 a_1 \ldots a_n) = P(a_0)P(a_1)\ldots P(a_n)$, for any $a_0 a_1 \ldots a_n \in \Sigma^\star$

Example: L-systems

Definition

A (deterministic context-free) L-system $L : \Sigma^\star \to \Sigma^\star$ is a function given by:

- an alphabet Σ and
- a function $P : \Sigma \to \Sigma^\star$ such that
- $L(a_0 a_1 \dots a_n) = P(a_0)P(a_1)\dots P(a_n)$, for any $a_0 a_1 \dots a_n \in \Sigma^\star$

Example:

$$\Sigma = \{a, b\}$$

$$P = \begin{cases} a \mapsto b \\ b \mapsto ab \end{cases}$$

$abbab$

Example: L-systems

Definition

A (deterministic context-free) L-system $L : \Sigma^\star \to \Sigma^\star$ is a function given by:

- an alphabet Σ and
- a function $P : \Sigma \to \Sigma^\star$ such that
- $L(a_0 a_1 \dots a_n) = P(a_0)P(a_1)\dots P(a_n)$, for any $a_0 a_1 \dots a_n \in \Sigma^\star$

Example:

$$\Sigma = \{a, b\}$$

$$P = \begin{cases} a \mapsto b \\ b \mapsto ab \end{cases}$$

$$abbab$$
$$\Downarrow$$
$$bababbab$$

Example: L-systems

Definition

A (deterministic context-free) L-system $L : \Sigma^\star \to \Sigma^\star$ is a function given by:

- an alphabet Σ and
- a function $P : \Sigma \to \Sigma^\star$ such that
- $L(a_0 a_1 \dots a_n) = P(a_0)P(a_1)\dots P(a_n)$, for any $a_0 a_1 \dots a_n \in \Sigma^\star$

Example:

$$\Sigma = \{a, b\}$$

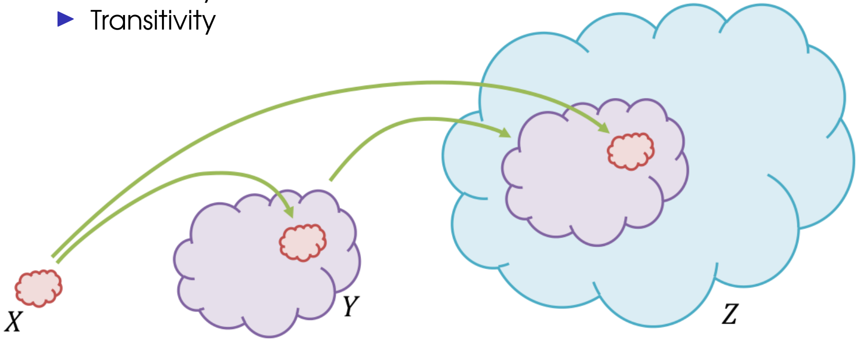
$$P = \begin{cases} a \mapsto b \\ b \mapsto ab \end{cases}$$

$$abbab$$
$$\Downarrow$$
$$bababbab$$
$$\Downarrow$$
$$abbabbababbab$$

Category

Rewriting: Decomposition and recomposition

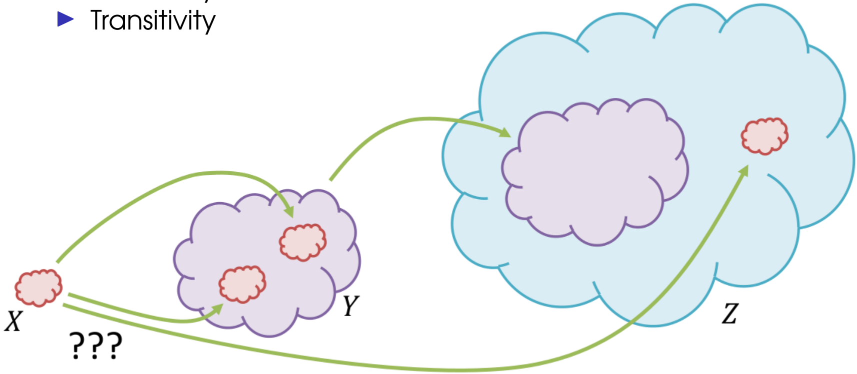
- Notion of "thing" is part of other "thing"
- Preorder structure
  - Reflexivity
  - Transitivity



Category

Rewriting: Decomposition and recomposition

- Notion of "thing" is part of other "thing"
- Preorder structure
  - Reflexivity
  - Transitivity



Category

Rewriting: Decomposition and recomposition

- Notion of "thing" is part of other "thing" at some "place"
- Category
  - Identity
  - Composition