

Automated Termination Proving: Contributions to Graph Rewriting via Extended Weighted Type Graphs and Morphism Counting

Qi QIU

LIRIS, UMR 5205 CNRS
Université Claude Bernard Lyon 1, France
Supervisor: Xavier URBAIN

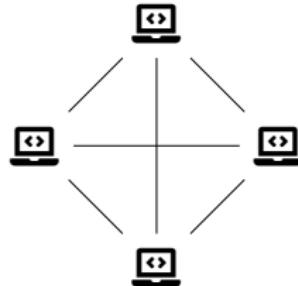


Université Claude Bernard



Motivation & Goal

Distributed systems:



Failures can be catastrophic: A row of four small icons representing different critical systems: a hospital bed, a train, an airplane, and a rocket ship.

Ensuring correctness is difficult.

- ▶ Needham-Schroeder protocol shown to be vulnerable 17 years after publication

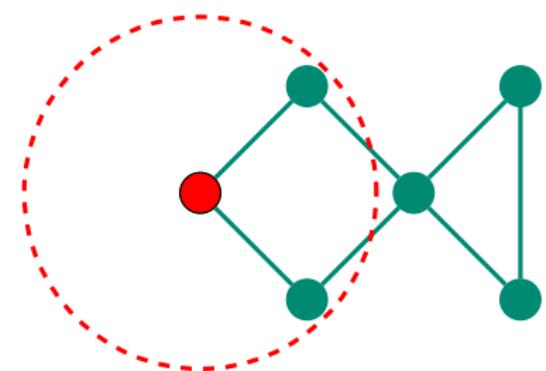
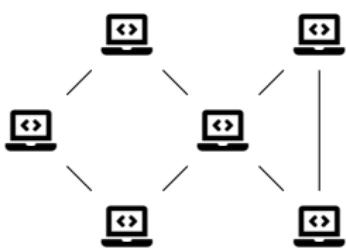
This thesis: automated verification

- ▶ Minimal user effort
- ▶ No expertise required
- ▶ Mathematically rigorous

Graph Transformation: Intuition

Modeling of distributed systems

System configurations: graphs

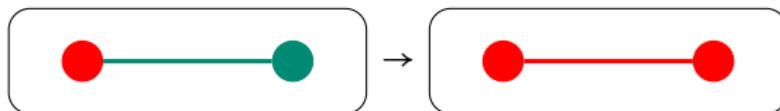


Algorithm behavior:

graph transformation based on **local** knowledge

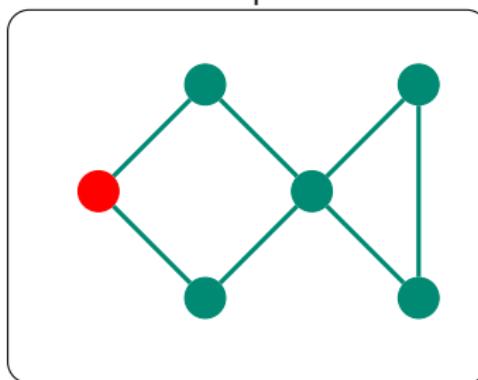
Graph Transformation: Spanning-tree Construction

Graph transformation rule:



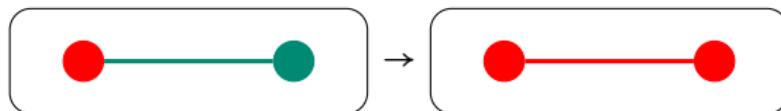
Replace the left-hand side with the right-hand side

Application of the rule while possible:



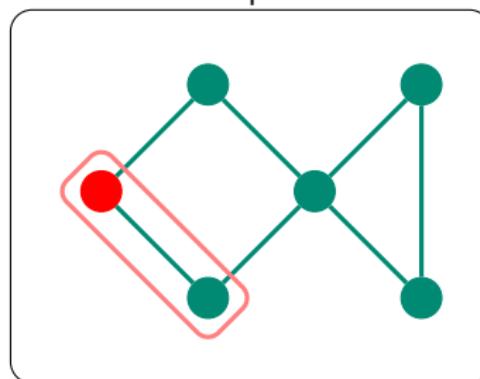
Graph Transformation: Spanning-tree Construction

Graph transformation rule:



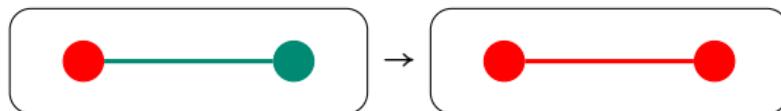
Replace the left-hand side with the right-hand side

Application of the rule while possible:



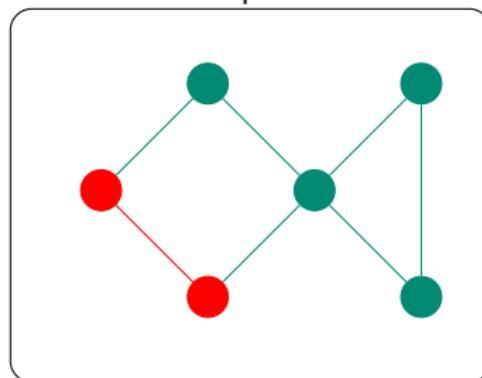
Graph Transformation: Spanning-tree Construction

Graph transformation rule:



Replace the left-hand side with the right-hand side

Application of the rule while possible:



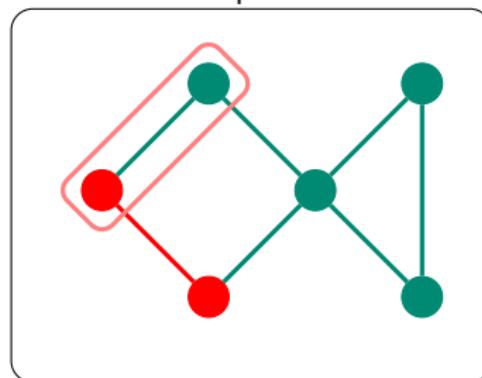
Graph Transformation: Spanning-tree Construction

Graph transformation rule:



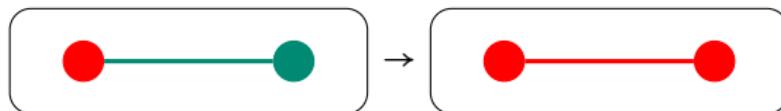
Replace the left-hand side with the right-hand side

Application of the rule while possible:



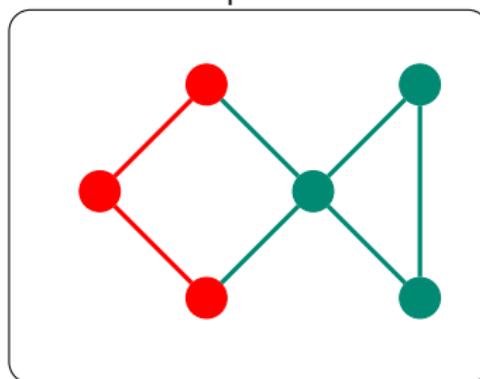
Graph Transformation: Spanning-tree Construction

Graph transformation rule:



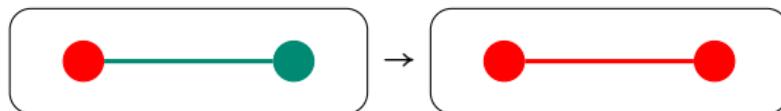
Replace the left-hand side with the right-hand side

Application of the rule while possible:



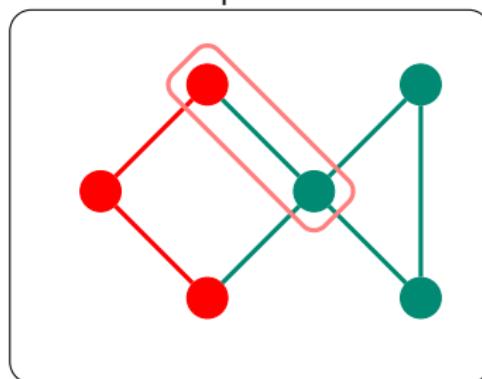
Graph Transformation: Spanning-tree Construction

Graph transformation rule:



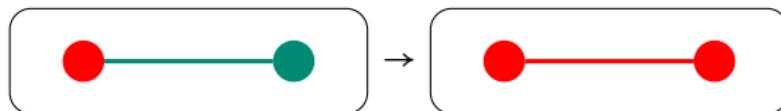
Replace the left-hand side with the right-hand side

Application of the rule while possible:



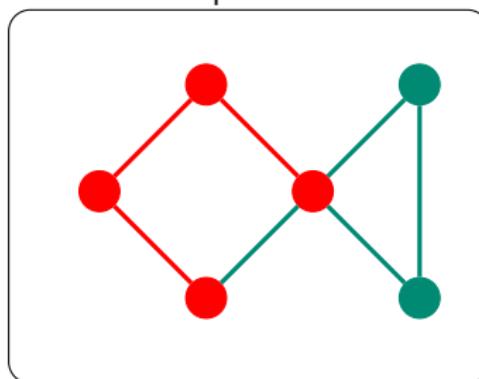
Graph Transformation: Spanning-tree Construction

Graph transformation rule:



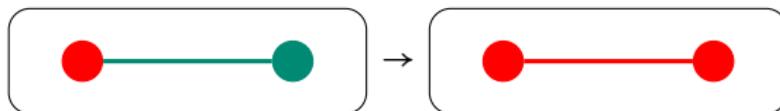
Replace the left-hand side with the right-hand side

Application of the rule while possible:



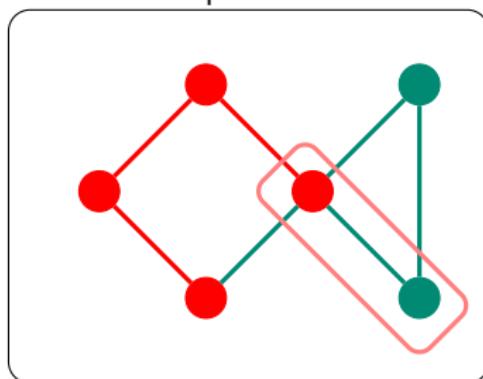
Graph Transformation: Spanning-tree Construction

Graph transformation rule:



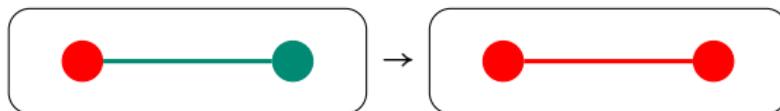
Replace the left-hand side with the right-hand side

Application of the rule while possible:



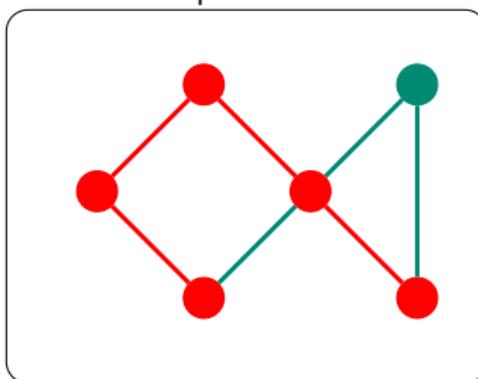
Graph Transformation: Spanning-tree Construction

Graph transformation rule:



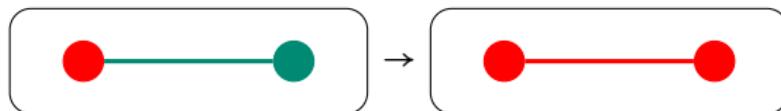
Replace the left-hand side with the right-hand side

Application of the rule while possible:



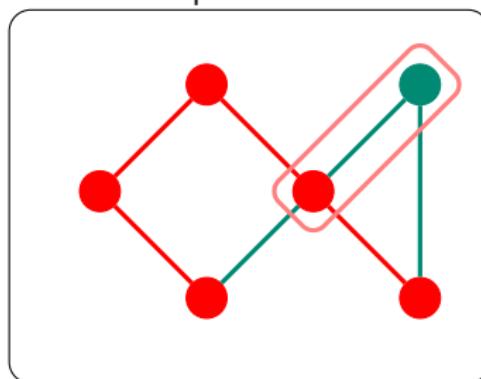
Graph Transformation: Spanning-tree Construction

Graph transformation rule:



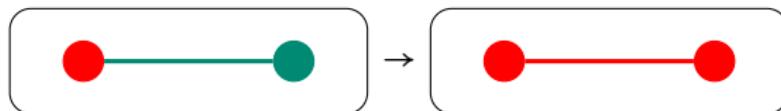
Replace the left-hand side with the right-hand side

Application of the rule while possible:



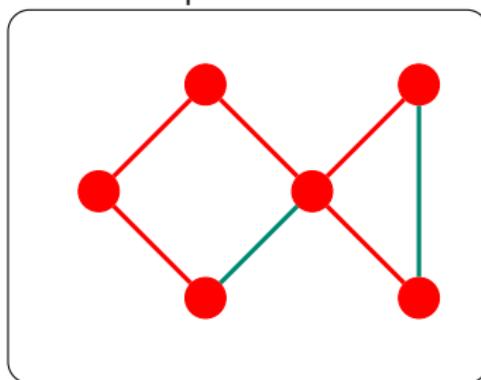
Graph Transformation: Spanning-tree Construction

Graph transformation rule:



Replace the left-hand side with the right-hand side

Application of the rule while possible:



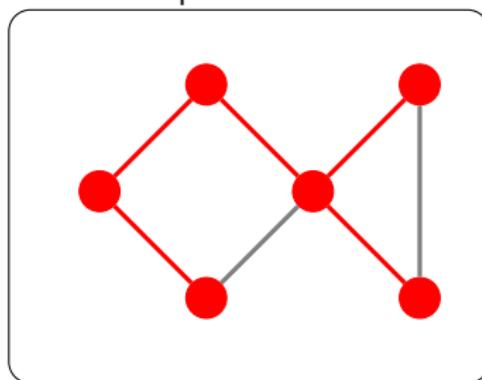
Graph Transformation: Spanning-tree Construction

Graph transformation rule:



Replace the left-hand side with the right-hand side

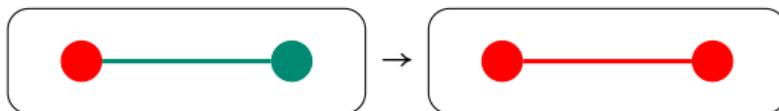
Application of the rule while possible:



The result is a spanning tree.

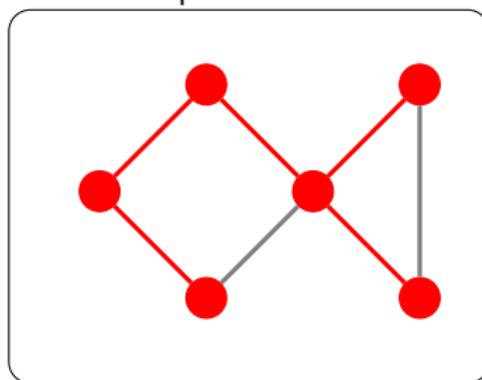
Graph Transformation: Spanning-tree Construction

Graph transformation rule:



Replace the left-hand side with the right-hand side

Application of the rule while possible:



The result is a spanning tree.

Does a result always exist?

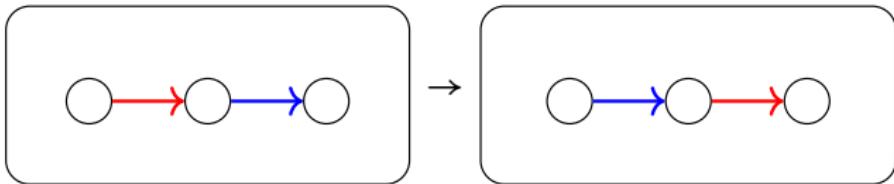
Graph Transformation: Termination

- ▶ No graph G_0 can be transformed forever

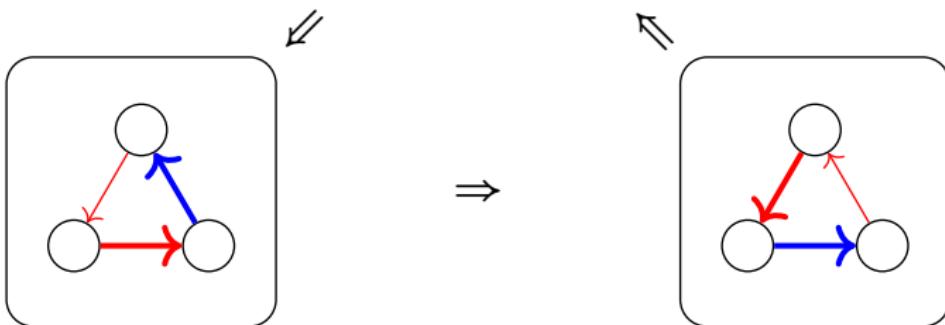
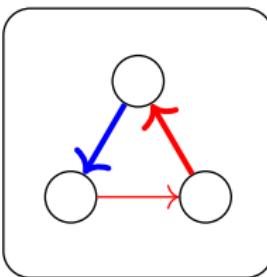
$$G_0 \Rightarrow G_1 \Rightarrow \dots$$

- ▶ Aligns with the notion of program termination:
“every execution (on any input) halts.”
- ▶ Undecidable in general
 - ▶ Automated techniques
 - ▶ Power: incomplete
 - ▶ Usability: rely on user-provided parameters

Graph Transformation: A Non-Terminating Rule



Loop:

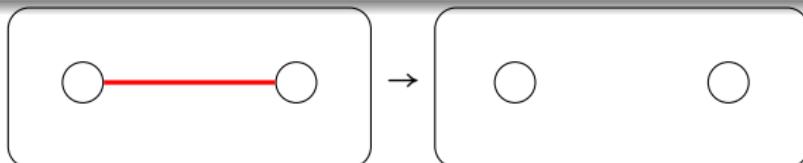


Graph Transformation: Termination by interpretations

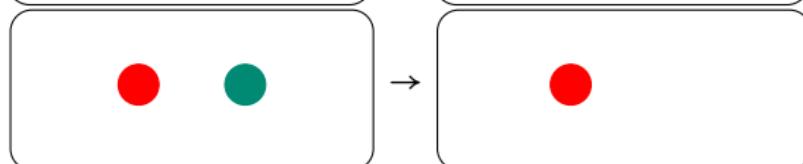
Interpret graphs as natural numbers.

Show that each transformation step strictly decreases the value.

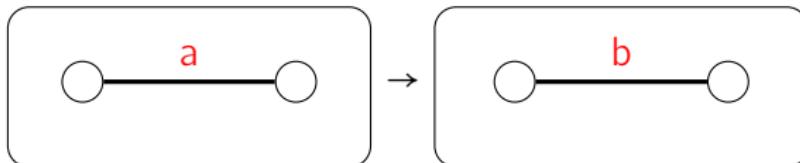
Number of edges:



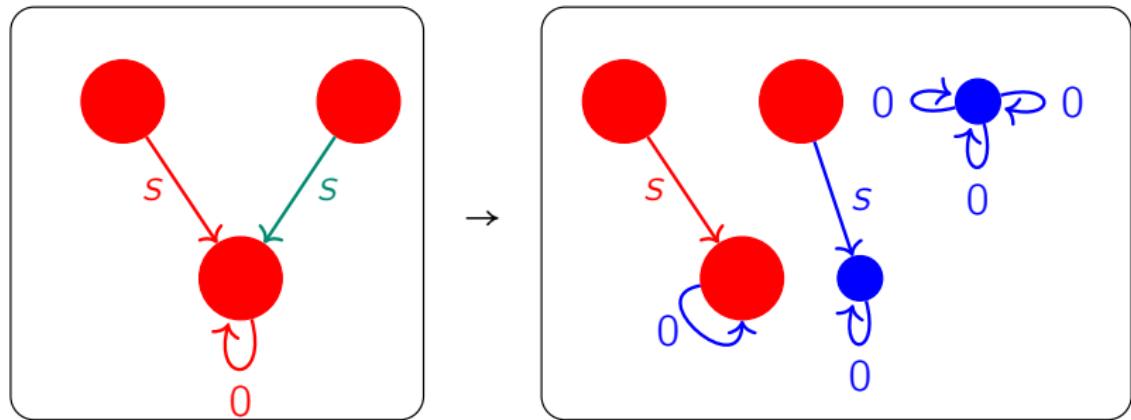
Number of nodes:



Number of edges labeled by a :



Limitations



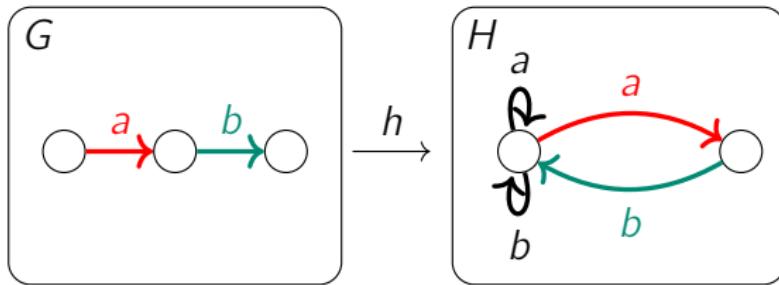
- ▶ The number of nodes/edges/labels does not decrease
- ▶ Can its termination be proved using interpretations?

Formal Definition of Graph Rewriting

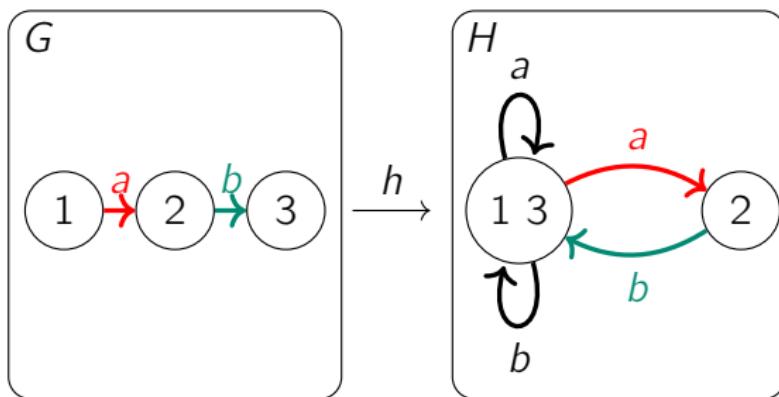
Toward Greater Usability

Toward Greater Power

Graph Morphisms: Structure-Preserving Functions

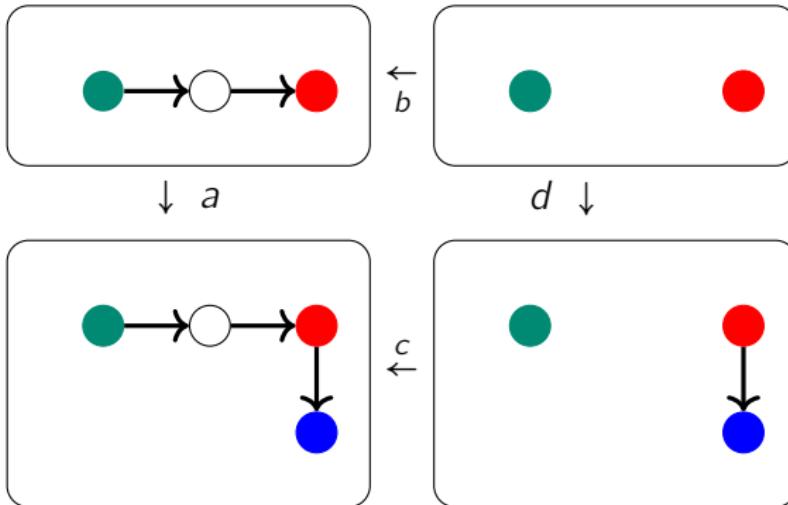


Colors indicate edge correspondence.



Numbers indicate node correspondence.

Commutative Diagram



$$\begin{array}{ccc} L & \xleftarrow{b} & K \\ \downarrow a & & \downarrow d \\ G & \xleftarrow{c} & C \end{array}$$

commutes if $a \circ b = c \circ d$.

Pushouts: Gluing Graphs Along an Interface

The **pushout** of (α, β) is (β', α') with

- $\square ABC$ commutes,

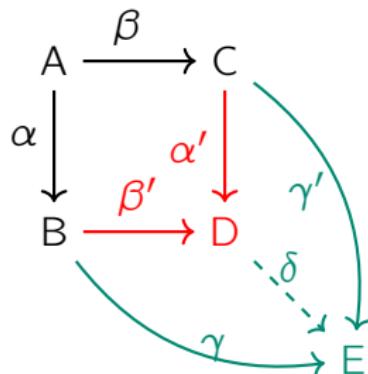
$$\begin{array}{ccc} A & \xrightarrow{\beta} & C \\ \alpha \downarrow & & \downarrow \alpha' \\ B & \xrightarrow{\beta'} & D \end{array}$$

D: pushout object

Pushouts: Gluing Graphs Along an Interface

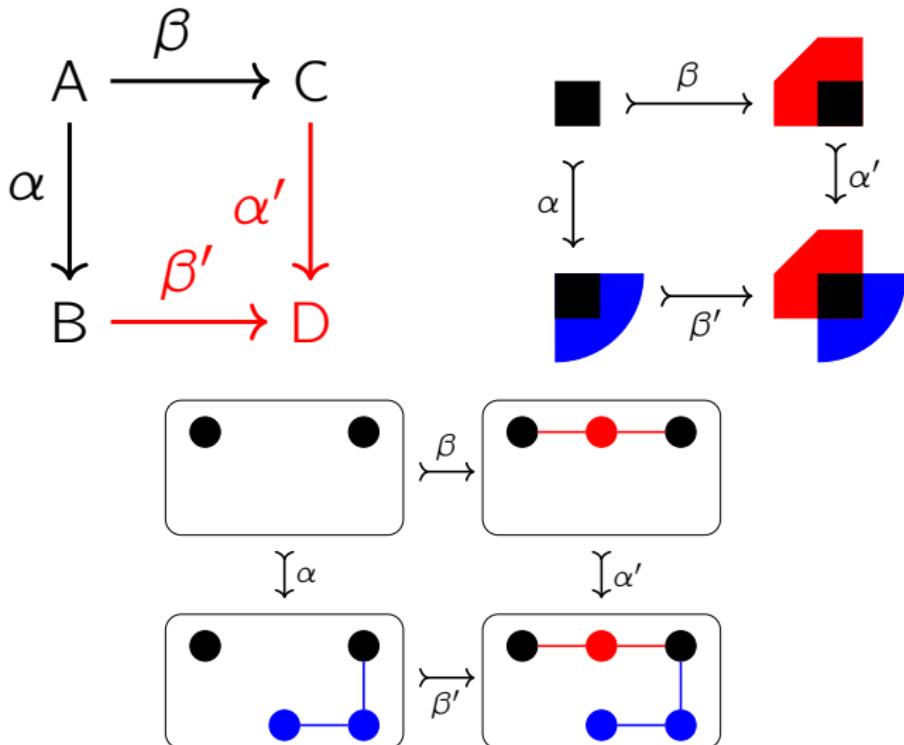
The **pushout** of (α, β) is (β', α') with

- ▶ $\square ABDC$ commutes,
- ▶ universality: for all (γ, γ') , if $\square ABEC$ commutes, then there is a unique δ such that $\triangle BDE$ and $\triangle CDE$ both commute.



D: pushout object

Pushouts: Gluing Graphs Along an Interface



Graph Rewriting with Double-Pushout (DPO)

First algebraic approach

One of the most studied approaches

$$L \xleftarrow{I} K \xrightarrow{r} R$$

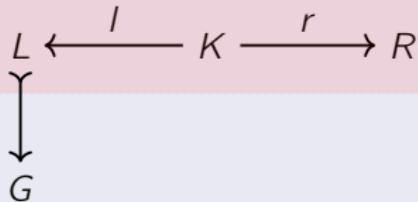
Rewriting rule with interface K



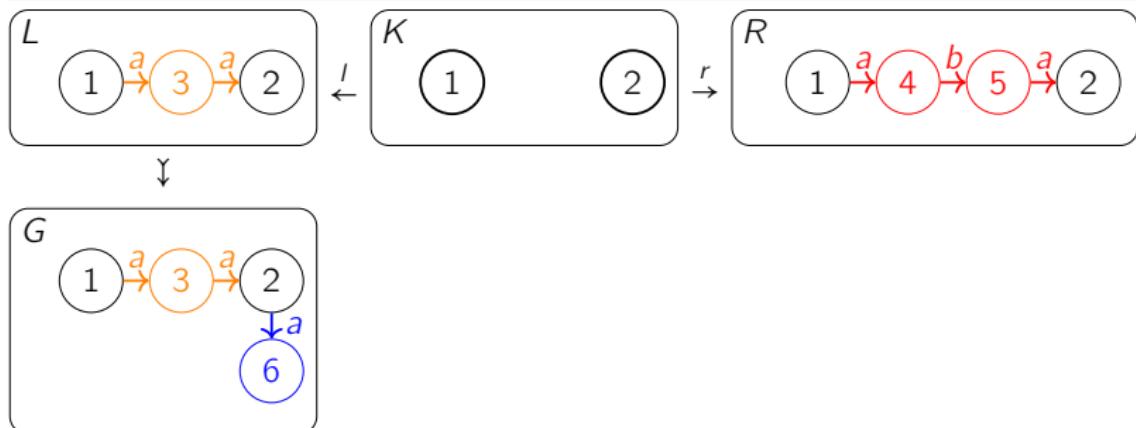
Graph Rewriting with Double-Pushout (DPO)

First algebraic approach

One of the most studied approaches



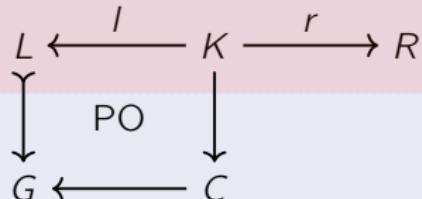
Rewriting rule with interface K



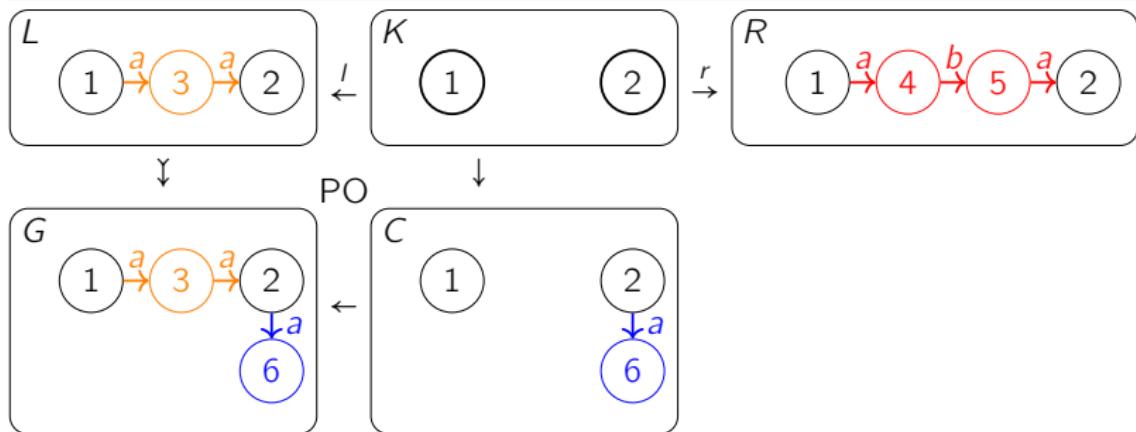
Graph Rewriting with Double-Pushout (DPO)

First algebraic approach

One of the most studied approaches



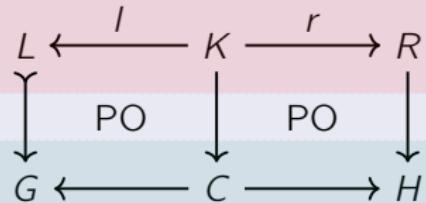
Rewriting rule with interface K



Graph Rewriting with Double-Pushout (DPO)

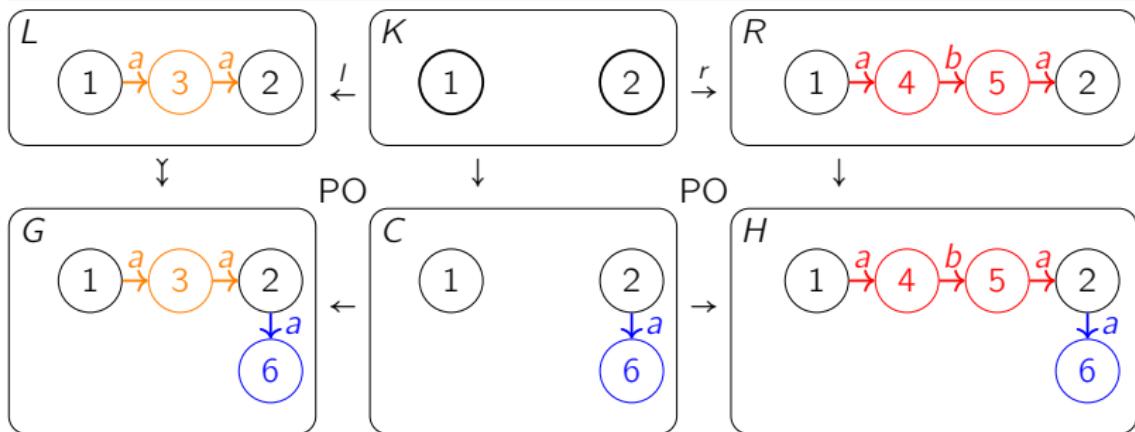
First algebraic approach

One of the most studied approaches

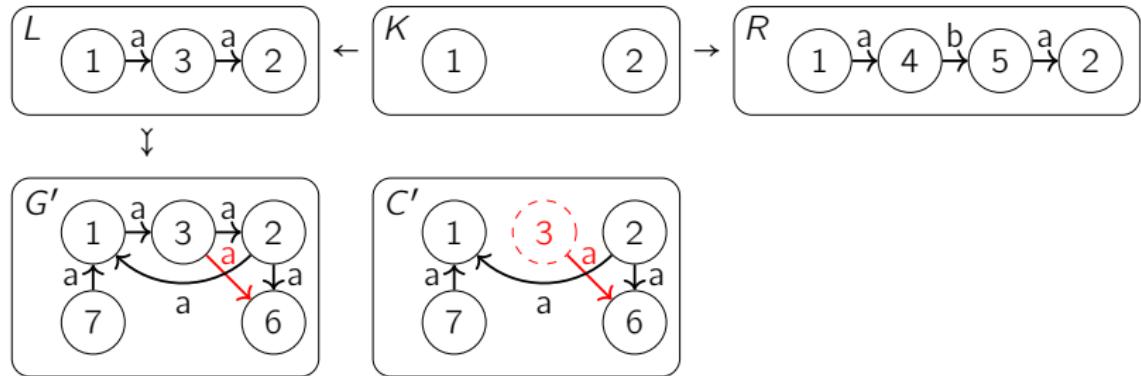


Rewriting rule with interface K

rewriting step $G \Rightarrow H$



An Invalid Rewriting Step



No dangling edges should be created.

Weighted Type Graph Method

Bruggink *et al.*, 2014

Parameter: an object T in the category, called the **type graph**.

Terminology: every graph is “typed” by morphisms to T

Interpretation:

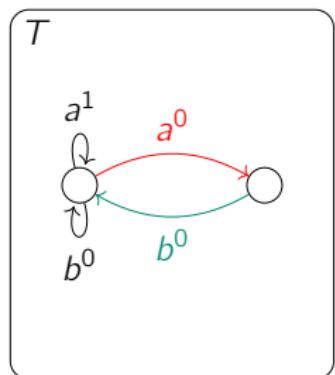
$$\begin{aligned} G &\rightsquigarrow \text{morphisms}(G, T) \\ &\rightsquigarrow \text{weight}(\text{morphisms}(G, T)) \\ &\rightsquigarrow \text{aggregator}(\text{weight}(\text{morphisms}(G, T))) \in \mathbb{N} \end{aligned}$$

How to choose the type graph T ?

How to define the morphism weight?

How to aggregate the morphism weights?

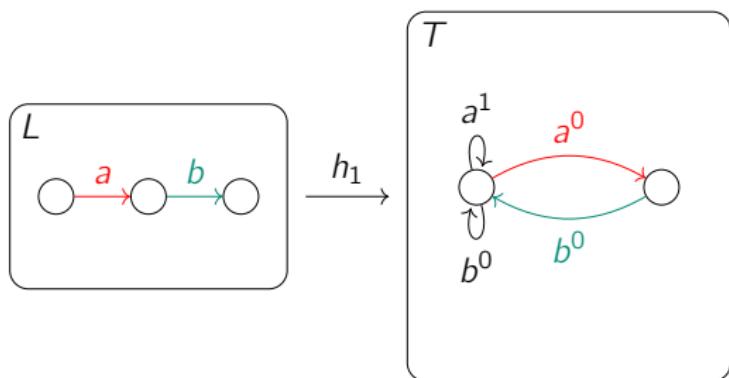
Type Graph with Weights on Edges



Morphism Weight

The weight of a morphism $h: G \rightarrow T$ is

$$\sum_{e \in \text{Edge}(G)} \text{weight}(h(e))$$

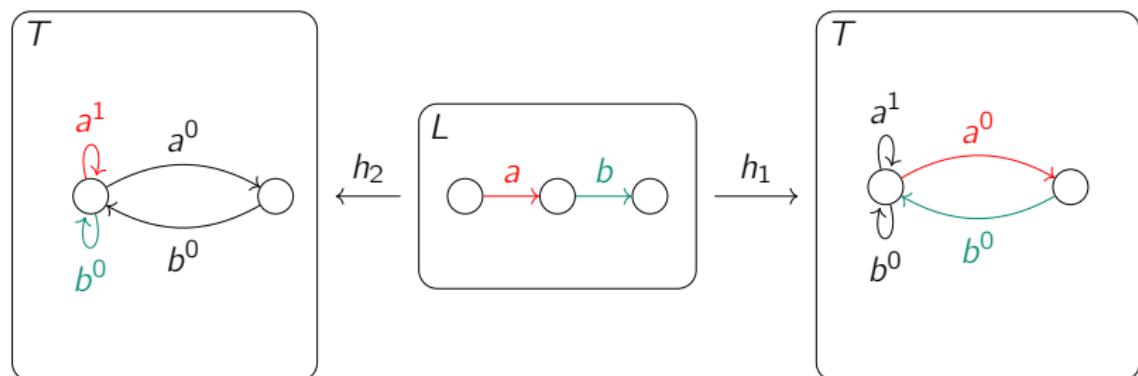


$$\text{weight}_T(h_1) = 0 + 0 = 0$$

Graph Weight

The weight of a graph L is

$$\min\{h : L \rightarrow T \mid \text{weight}_T(h)\}$$

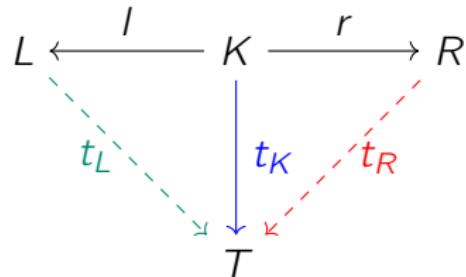


$$\text{weight}_T(h_2) = 1 + 0 = 1$$

$$\text{weight}_T(L) = \min\{1, 0\} = 0$$

$$\text{weight}_T(h_1) = 0 + 0 = 0$$

Termination Criterion [Bruggink *et al.*, 2014]



A rule terminates if there is a type graph T such that for all t_K , if there is t_L such that ΔKLT commutes, then

$$\begin{aligned} & \min\{\text{weight}_T(t_L) \mid t_L. \Delta KLT \text{ commutes}\} \\ & > \min\{\text{weight}_T(t_R) \mid t_R. \Delta KRT \text{ commutes}\} \end{aligned}$$

How to find a suitable weighted type graph?

Searching for Weighted Type Graphs over \mathbb{N}

User-specified parameters:

- ▶ k nodes
- ▶ maximum edge weight $n \in \mathbb{N}$

The problem amounts to checking the satisfiability of an existential Presburger arithmetic theory with:

- ▶ k^2m binary variables where m is the number of labels
- ▶ k^2m integer variables

Challenges:

- ▶ **Usability**: difficult to guess k and n
- ▶ **Search space**: $2^{k^2m} \cdot n^{k^2m}$ possible assignments of weights

Usability Improvement

Idea: Weights in \mathbb{R}^+

Additional constraint: there is $\delta > 0$ such that every rewriting step decreases the weight by at least δ .

Searching for Weighted Type Graphs over \mathbb{R}^+

User-specified parameters:

- ▶ k nodes
- ▶ ~~edge weights in $\{0, 1, \dots, n\}$~~

The problem amounts to checking the satisfiability of an
~~existential Presburger arithmetic theory~~ existential theory of the
reals with binary variables:

- ▶ $k^2 m$ binary variables where m is the number of labels
- ▶ $k^2 m$ ~~integer~~ real variables

Challenge:

- ▶ impossible to guess k and ~~maximum weight n~~
- ▶ complexity: ~~$2^{k^2 m} \cdot n^{k^2 m}$ possible assignments of weights~~ $2^{k^2 m}$ linear programs with $k^2 m$ variables which have polynomial-time average-case complexity.

Implementation ↵ Experiments

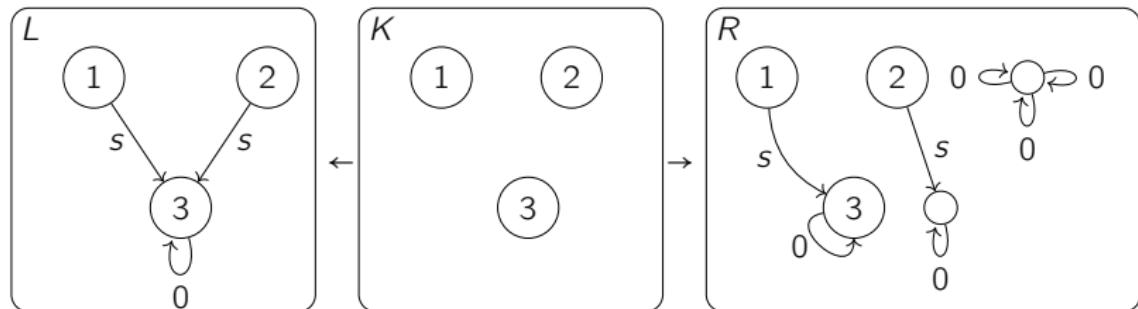
k from 1 to 4

For every k , n from 1 to 3 if over \mathbb{N}

	Configuration 1	Configuration 2	Configuration 3
[3, Example 6.3]			58%
[3, Example D.3]	48%		47%
[5, Example 3.8]	37%	36%	
[4, Example 4]		26%	timeout
[4, Example 5]	1%	1%	38%
[2, Example 4]	1%	1%	46%
[2, Example 5]			timeout
[2, Example 6]			timeout
[1, Example 1]	48%		47%
[1, Example 4]	46%	47%	49%
[1, Example 5]	24%	22%	timeout

Usability Improved ✓

A Limitation of the Weighted Type Graph Method



Type graph fails: existence of surjections from R to L

All existing automated methods fail.

Remark: the number of occurrences of strictly decreases.

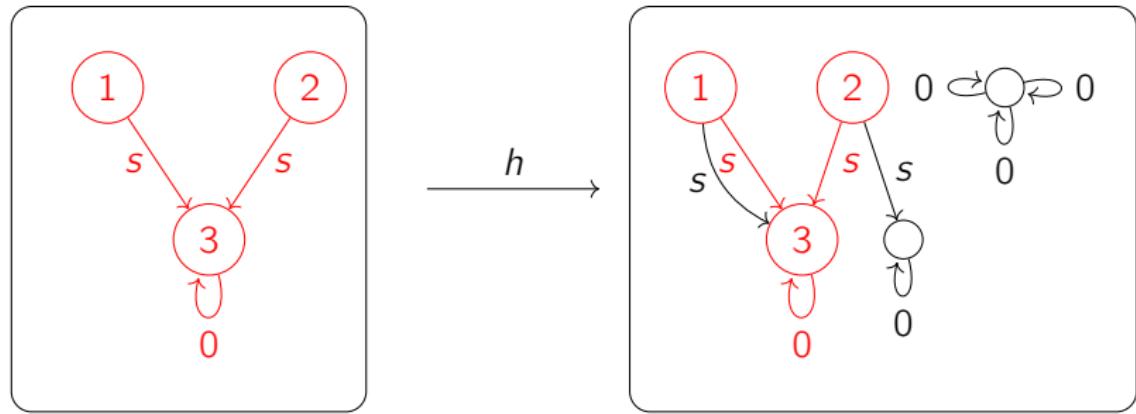
Capability Improvement: Morphism Counting

Parameter: graph X

Interpretation:

$$G \rightsquigarrow |\text{morphisms}(X, G)| \in \mathbb{N}$$

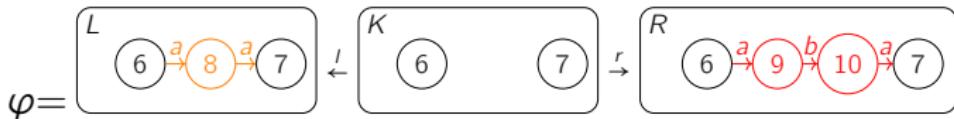
Inclusions



Subgraph

Graph Rewriting Systems

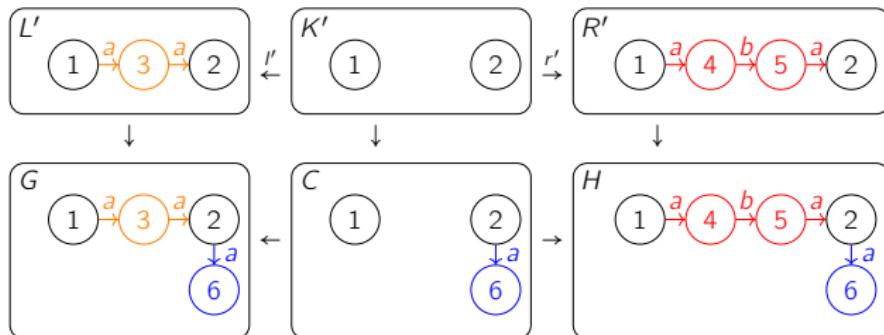
A rewriting rule consists of two inclusions.



An equivalent rewriting rule expresses the same transformation.

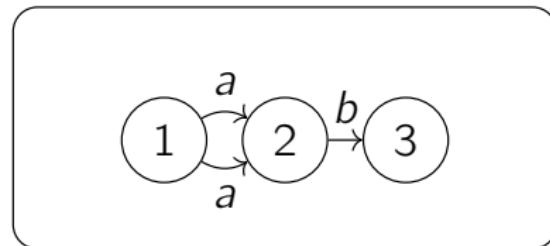


A rewriting step with φ is defined by a DPO diagram with inclusions and φ' .

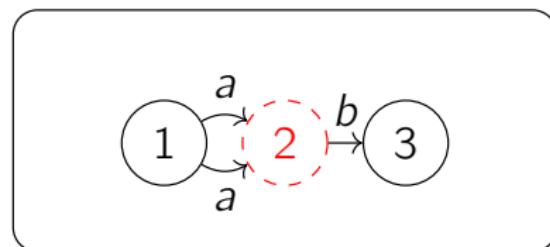


Pre-Graphs

Graph:

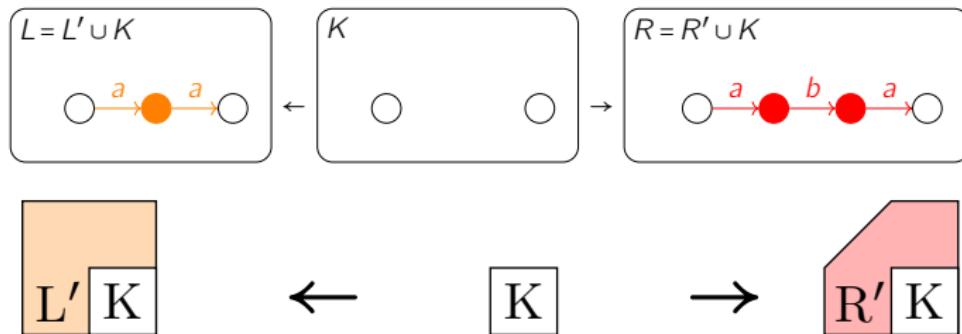


Pre-graphs obtained by removing node 2:

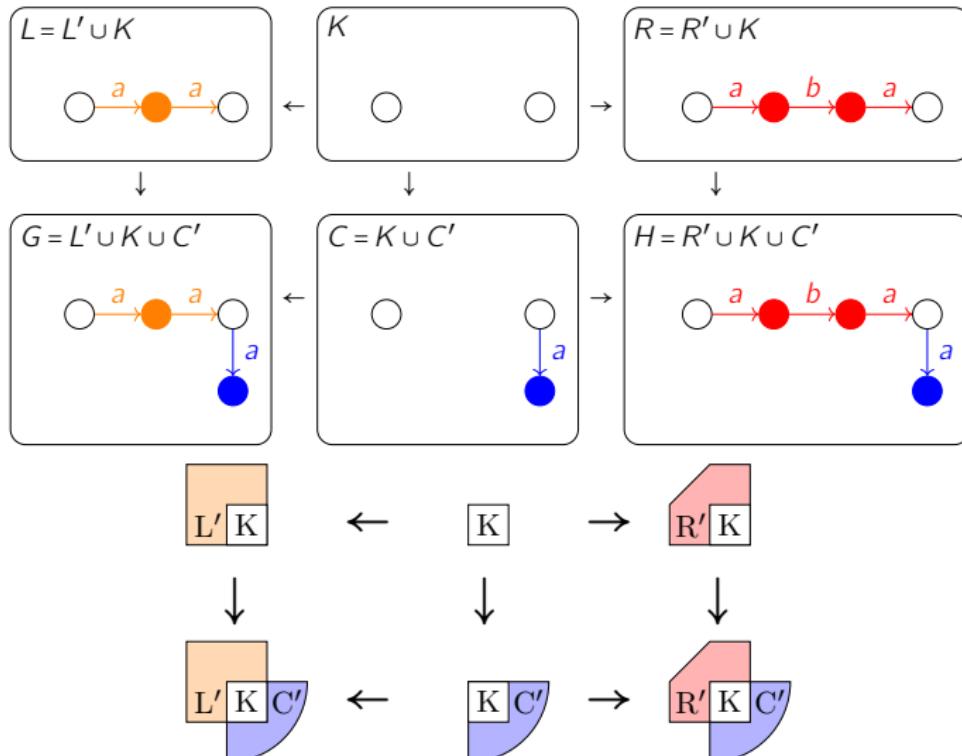


Dangling edges

Decomposition of Graphs in Rewriting Rules



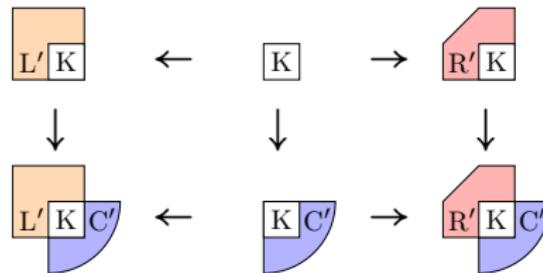
Decomposition of Graphs in Rewriting Steps



This coloring provides a classification of morphisms in rewriting steps by image node colors.

X -occurrences by Image Node Colors

An X -occurrence is an injective morphism from X .

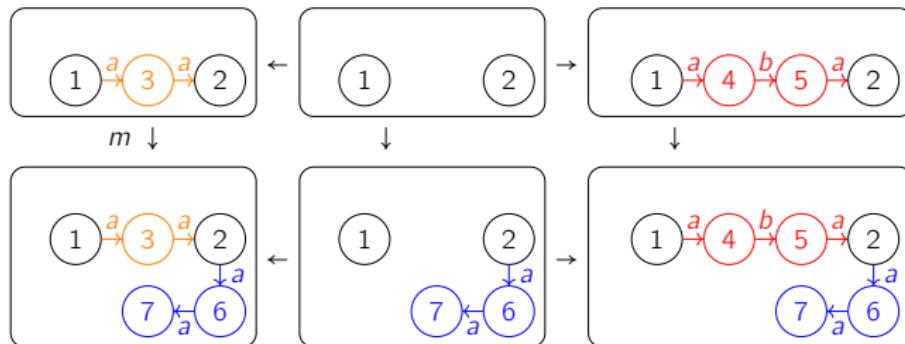


X -occurrence are classified by the colors of their image nodes:

- ▶ white: only white;
- ▶ blue: only white and at least one blue;
- ▶ blue-and-red: at least one blue and at least one red
- ▶ etc.

Morphisms by Image Node Colors

Let X be the graph

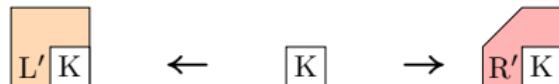


Blue X -occurrence:

Red X -occurrences: none.

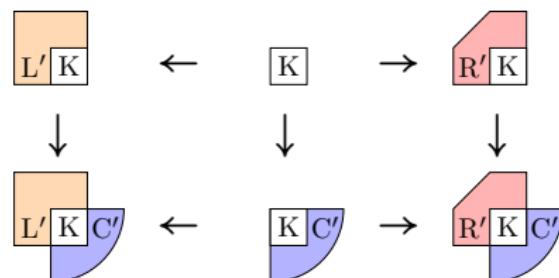
Blue-and-red X -occurrences:

A New Sufficient Condition for Termination [Qiu]



terminates if

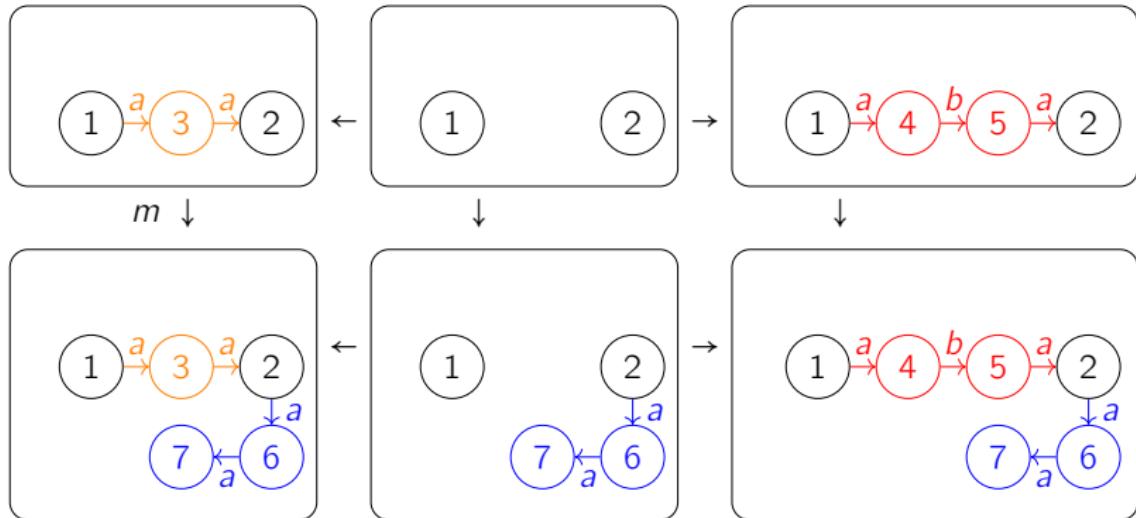
- ▶ it contains strictly more orange X-occurrences than red X-occurrences in the rule, and
- ▶ for every rewriting step:



there are more blue-and-orange X-occurrences than blue-and-red X-occurrences.

Challenge: verify the second condition for an unknown C'.

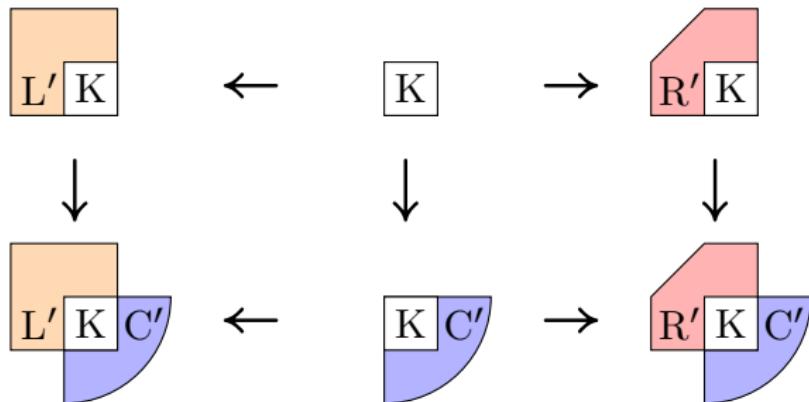
Analysis of Implicit Occurrences



Blue-and-red X-occurrences: $\textcolor{red}{5} \xrightarrow{a} 2 \xrightarrow{a} \textcolor{blue}{6}$

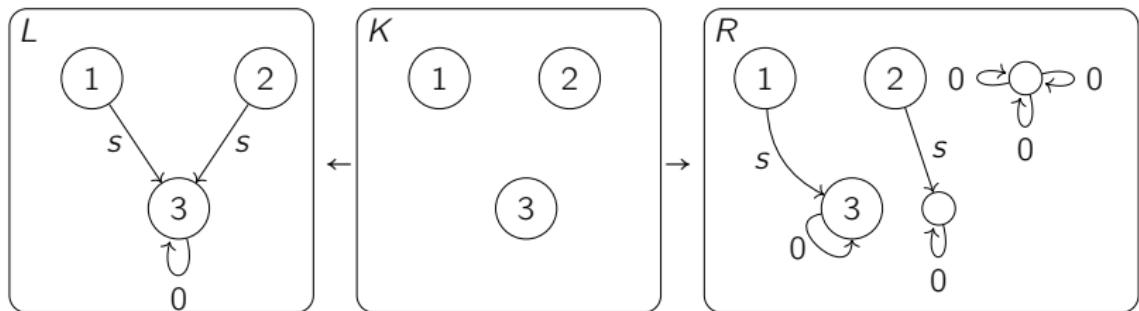
Blue-and-Orange X-occurrences: $\textcolor{orange}{3} \xrightarrow{a} 2 \xrightarrow{a} \textcolor{blue}{6}$

Sufficient Condition for the Second Condition [Qiu]



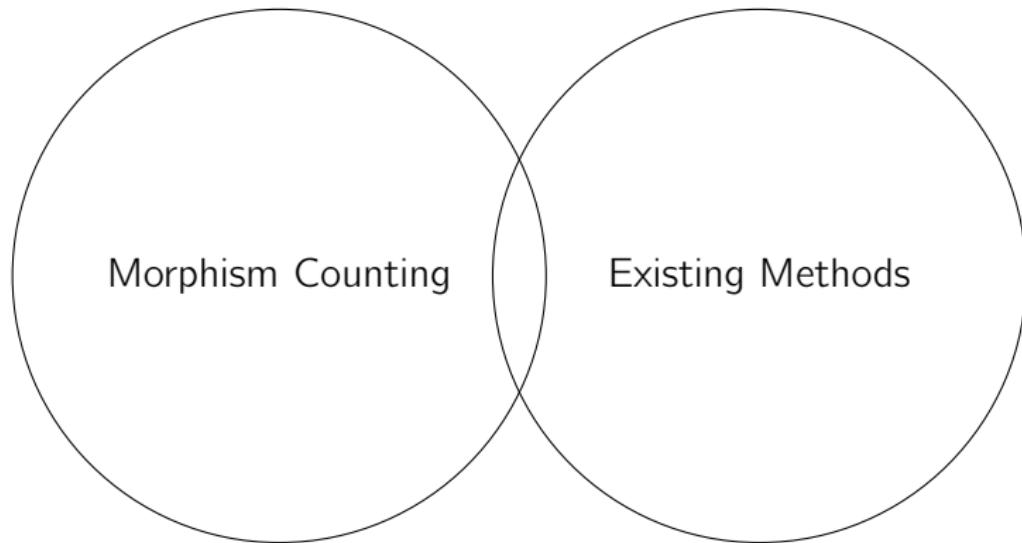
If all subgraphs of $\boxed{R'} \boxed{K}$ that can form an blue-and-red X-occurrence in any rewriting step can be mapped to distinct subgraphs in $\boxed{L'} \boxed{K}$ while preserving elements in \boxed{K} , then there are more blue-and-orange X-morphisms than blue-and-red X-morphisms.

Termination of Motivating Example



Terminating by counting morphisms from ✓

Imcomparable with Existing Methods



Succeed in some cases where all existing automated methods fail.

Fail in some cases where other methods succeed.

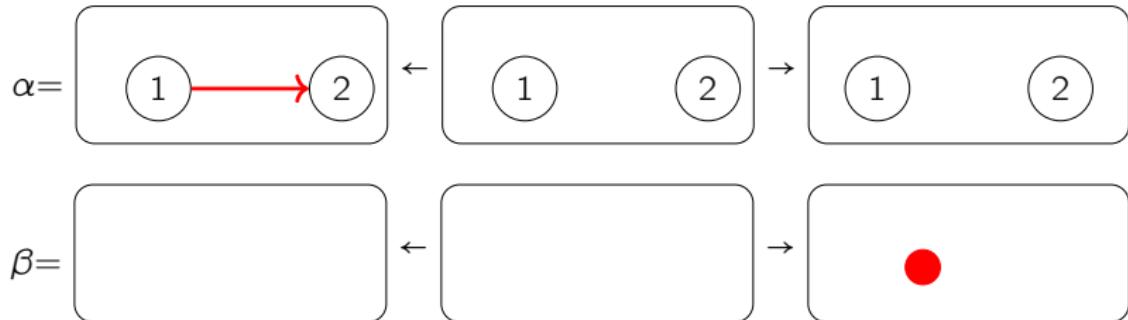
More power if search in parallel ✓

LyonParallel

Automated tool in Ocaml

Available : <https://github.com/Qi-tchi/LyonParallel>

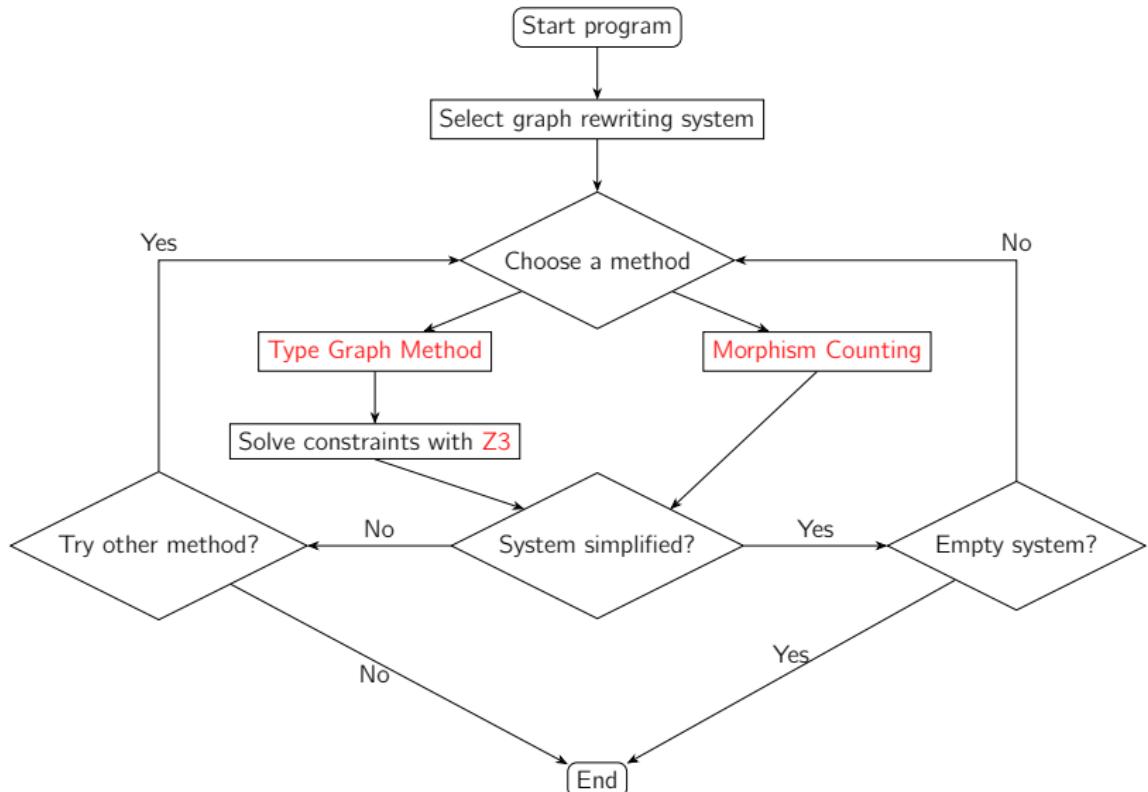
Relative Termination: Intuition



α can only be applied finitely many times.

α can be eliminated without affecting the result of termination analysis.

Process Flowchart of LyonParallel



Conclusion and Future Work

Contributions

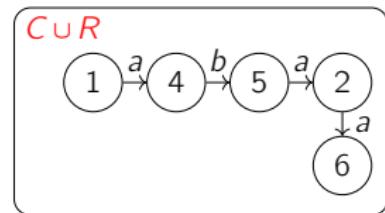
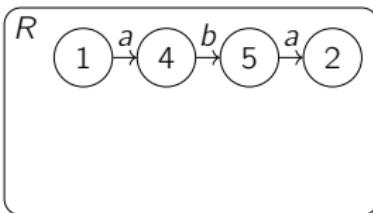
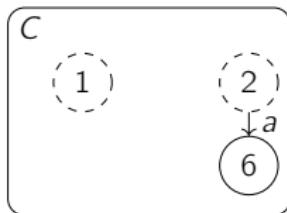
- ▶ Usability improvement,
- ▶ New termination criterion,
- ▶ Extension of the new termination criterion (not presented),
- ▶ Implementation of these contributions.

Future work

- ▶ Short term: comparison with the Subgraph-Counting method for PBPO+.
- ▶ Mid term: certificate-generation mechanism.
- ▶ Long term: extension to other graph rewriting frameworks (e.g., PBPO+)

Pre-Graph Operations

Union of two pre-graphs $C \subseteq G$ and $R \subseteq G$, denoted $C \cup R$.



Relative complement of R in H where $R \subseteq H$, denoted $H \setminus R$.

