

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



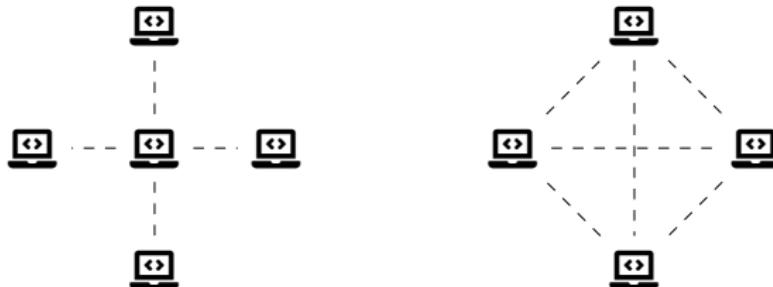
Université Claude Bernard



Lyon 1

Motivation & Goal

Centralized systems and distributed systems:



Failures can be catastrophic:

- ▶ The Needham-Schroeder protocol flaw
(1978-1995)

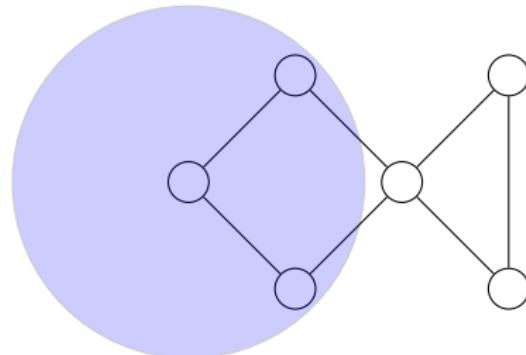
Ensuring correctness is hard.

- ▶ Distributed algorithms are complex.
- ▶ Human proofs are error-prone.

This thesis: automated verification.

- ▶ Mathematically rigorous
- ▶ Minimal user effort

Graph Transformation

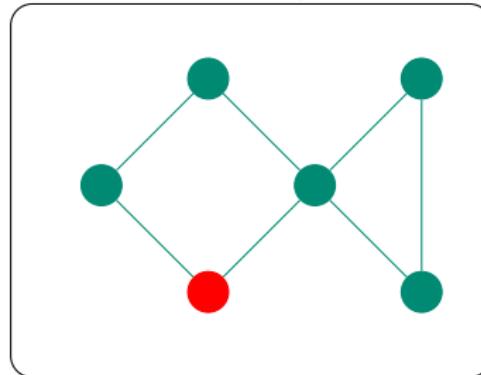


Intuitive modelization of distributed systems:

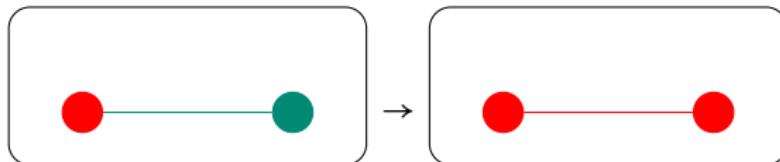
- ▶ computational units → nodes
- ▶ communication channels → edges
- ▶ system states → graphs
- ▶ algorithm behaviors
→ graph transformation rules according to local knowledge

Graph Transformation

Configuration of a distributed system:



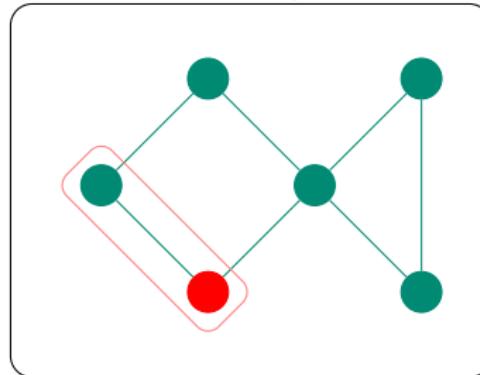
Apply the rule as long as possible:



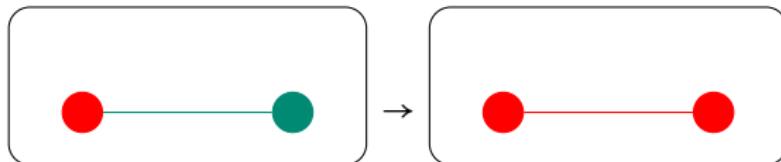
This rule replaces an occurrence of the left-hand side by the right-hand side.

Graph Transformation

Configuration of a distributed system:



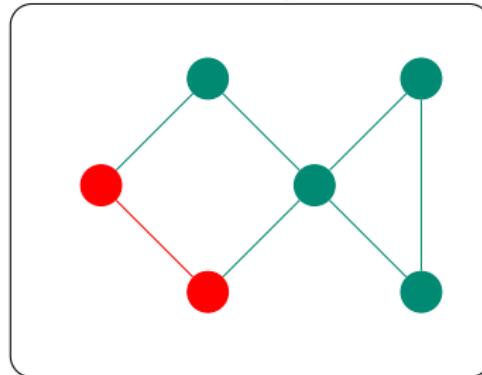
Apply the rule as long as possible:



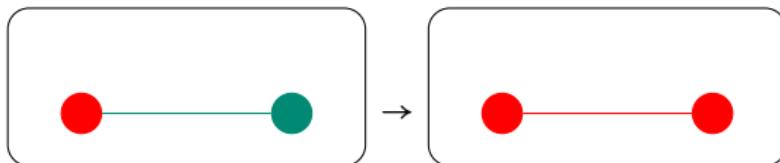
This rule replaces an occurrence of the left-hand side by the right-hand side.

Graph Transformation

Configuration of a distributed system:



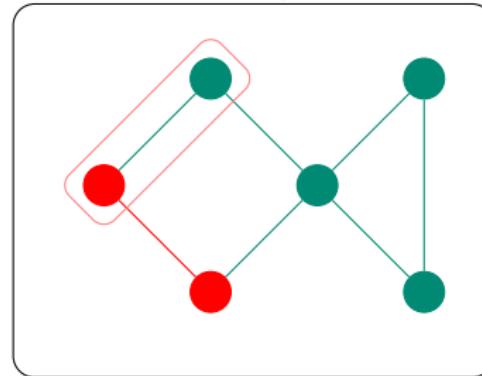
Apply the rule as long as possible:



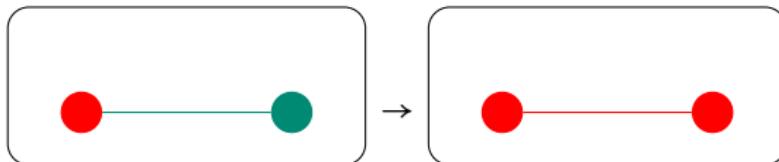
This rule replaces an occurrence of the left-hand side by the right-hand side.

Graph Transformation

Configuration of a distributed system:



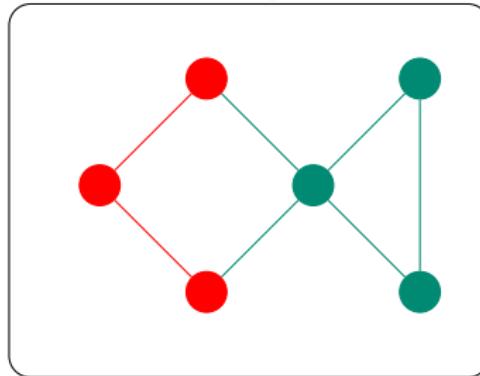
Apply the rule as long as possible:



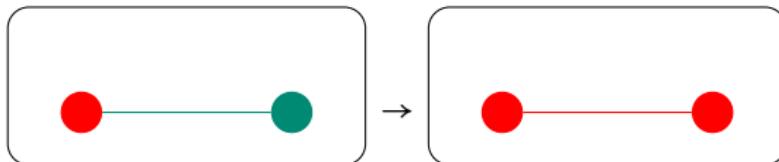
This rule replaces an occurrence of the left-hand side by the right-hand side.

Graph Transformation

Configuration of a distributed system:



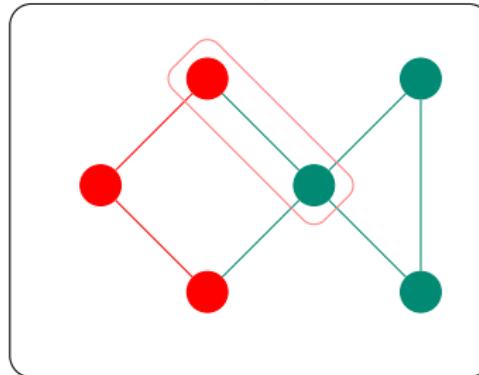
Apply the rule as long as possible:



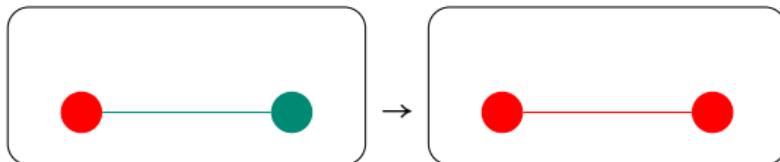
This rule replaces an occurrence of the left-hand side by the right-hand side.

Graph Transformation

Configuration of a distributed system:



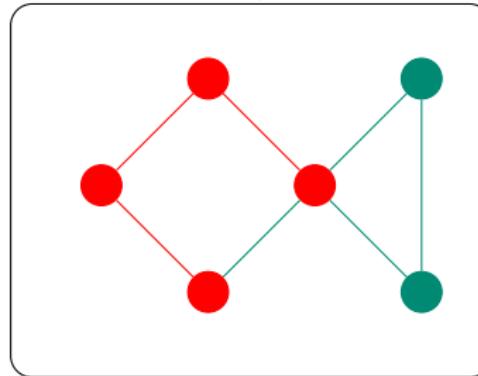
Apply the rule as long as possible:



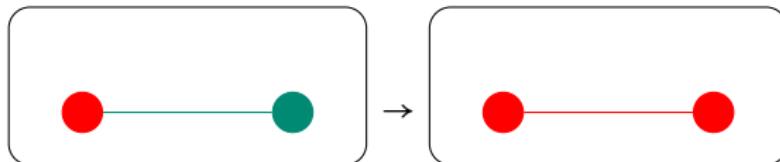
This rule replaces an occurrence of the left-hand side by the right-hand side.

Graph Transformation

Configuration of a distributed system:



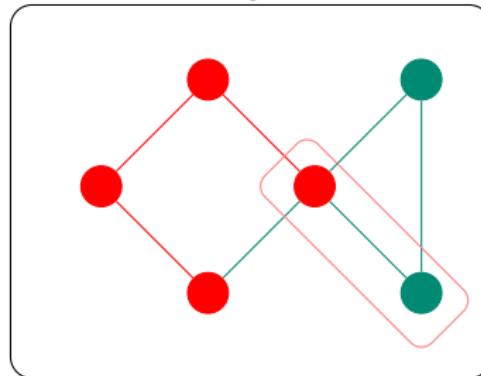
Apply the rule as long as possible:



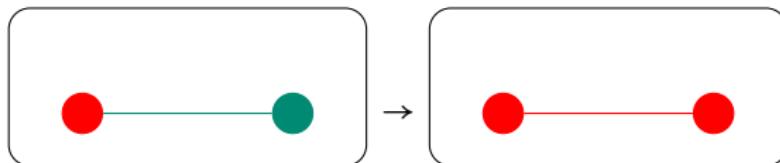
This rule replaces an occurrence of the left-hand side by the right-hand side.

Graph Transformation

Configuration of a distributed system:



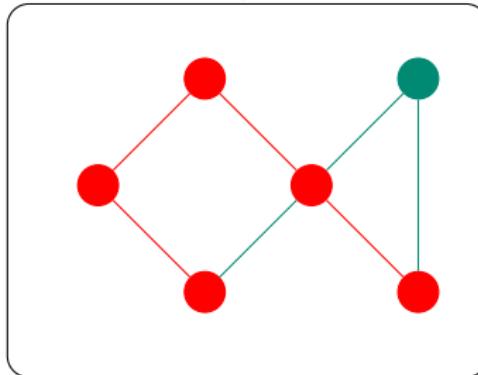
Apply the rule as long as possible:



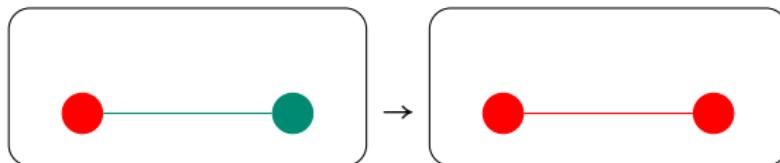
This rule replaces an occurrence of the left-hand side by the right-hand side.

Graph Transformation

Configuration of a distributed system:



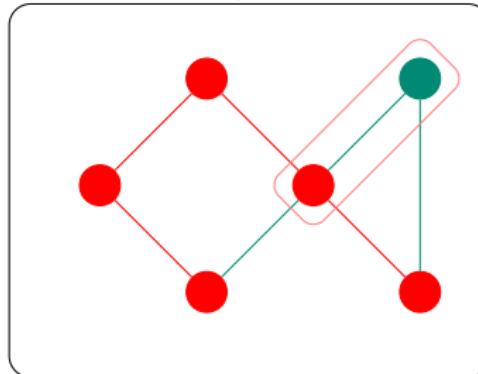
Apply the rule as long as possible:



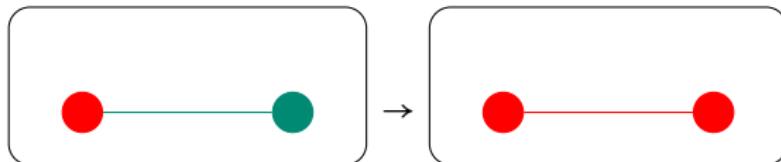
This rule replaces an occurrence of the left-hand side by the right-hand side.

Graph Transformation

Configuration of a distributed system:



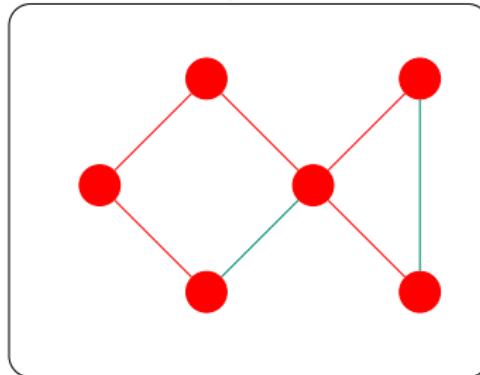
Apply the rule as long as possible:



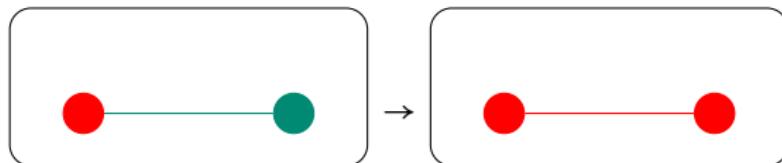
This rule replaces an occurrence of the left-hand side by the right-hand side.

Graph Transformation

Configuration of a distributed system:



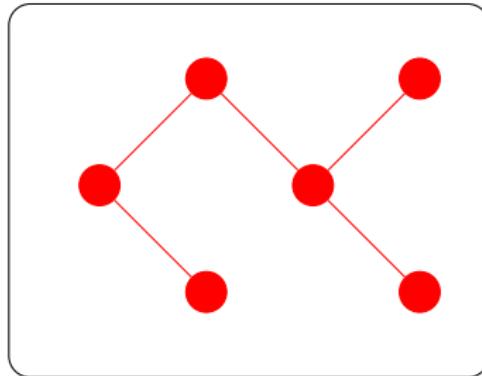
Apply the rule as long as possible:



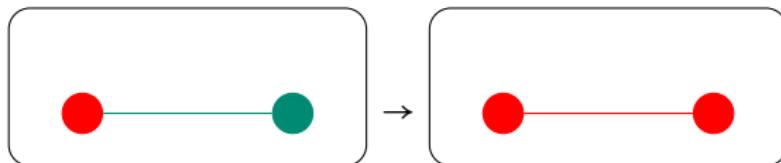
This rule replaces an occurrence of the left-hand side by the right-hand side.

Graph Transformation

A spanning tree is obtained when the rule cannot be applied:



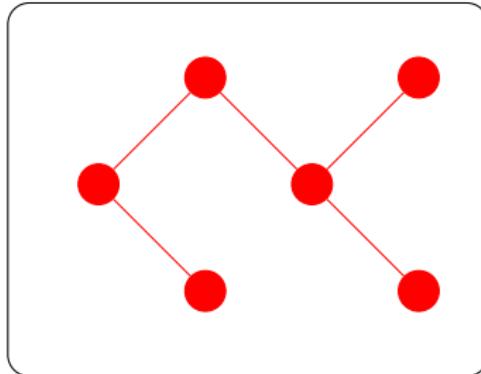
Apply the rule as long as possible:



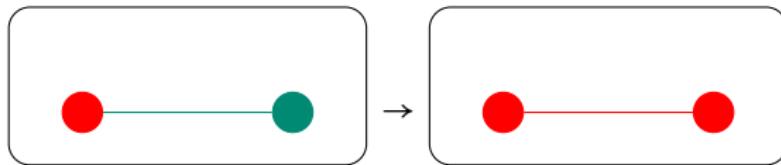
This rule replaces an occurrence of the left-hand side by the right-hand side.

Graph Transformation

A spanning tree is obtained when the rule cannot be applied:



Apply the rule as long as possible:



This rule replaces an occurrence of the left-hand side by the right-hand side.

Can a given set of rules transform a given initial graph forever?

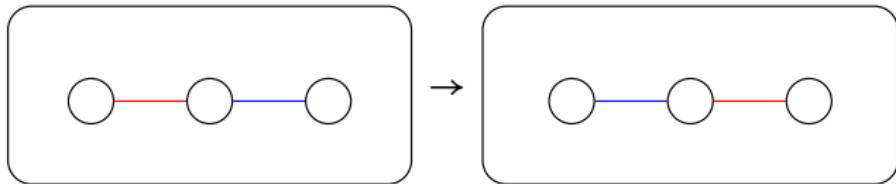
Termination

- ▶ No graph G_0 can be transformed forever

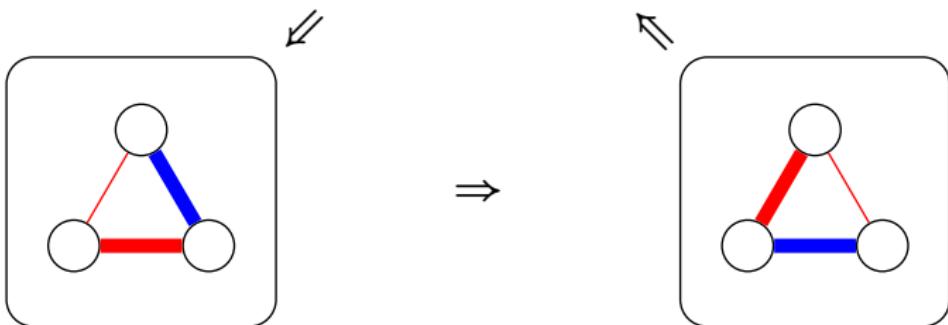
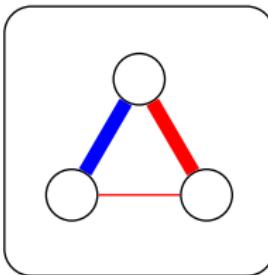
$$G_0 \Rightarrow G_1 \Rightarrow \dots$$

- ▶ Aligns with the notion of program termination:
“every execution (on any input) halts.”
- ▶ Undecidable in general

A non-terminating case



Loop:

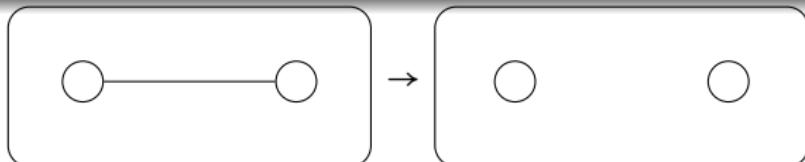


Automated Termination Proofs

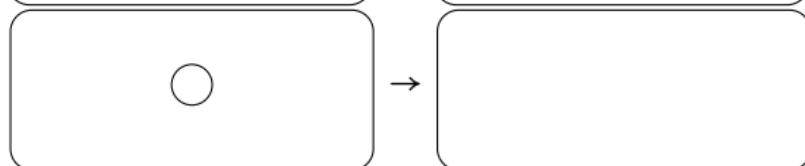
Termination by interpretations:

- ▶ interpret graphs as natural numbers;
- ▶ show that each transformation step decreases the value.

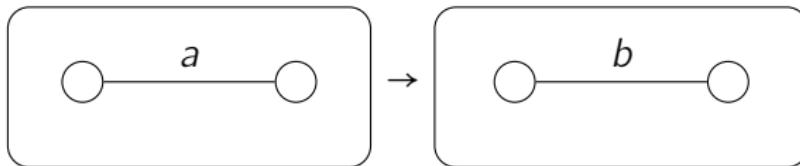
Number of edges:



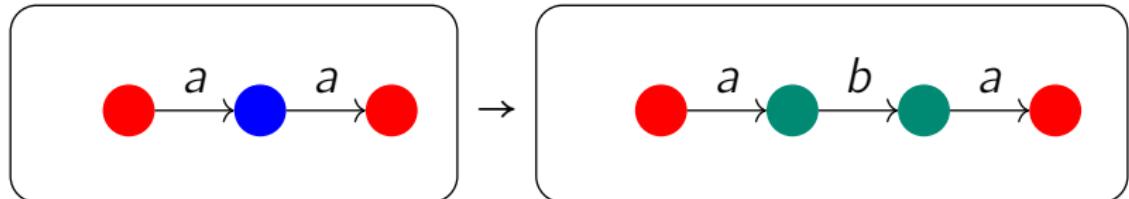
Number of nodes:



Number of edges labeled by a :



Limitation of Trivial Interpretations



Left-hand side graph: middle has no other incident edges

Right-hand side graph: middles are fresh nodes

Termination: number of $\textcircled{O} \xrightarrow{a} \textcircled{O} \xrightarrow{a} \textcircled{O}$ decreases

Can its termination be proved by interpretations?

- ▶ Weighted Type Graph Method
- ▶ Need a more powerful definition of graph transformations.

Preliminaries

Graphs and Graph Morphisms

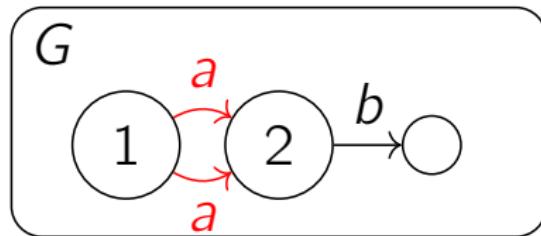
Graph Rewriting with Double-Pushout (DPO)

Toward greater usability

Toward greater power

LyonParallel—A Tool for Termination of Graph Rewriting

Graphs: Finite, Directed, Edge-Labeled Multigraphs

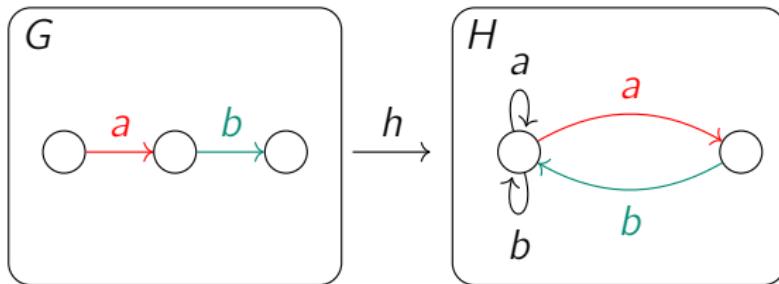


Edges with the same source, target and label are permitted.

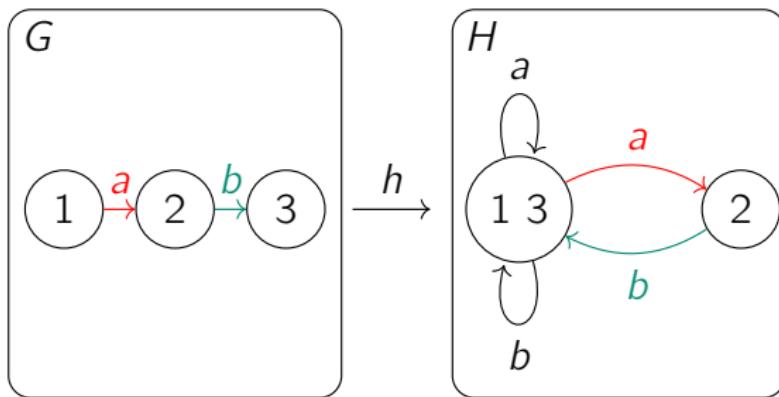
G : graph name

Numbers inside nodes: identifiers

Graph Morphisms: Structure-Preserving Functions



Colors show edge correspondence.



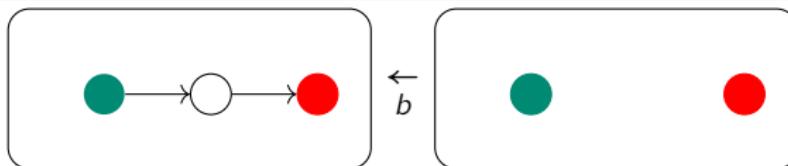
Numbers show node correspondence.

h : morphism name

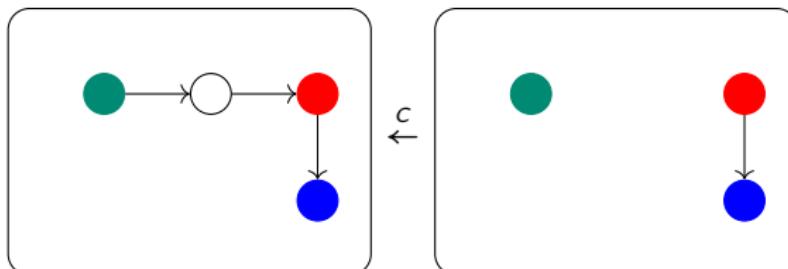
Commutative Diagram

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

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

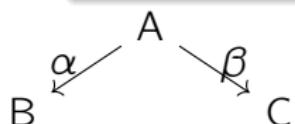


$$d \quad \downarrow$$



Pushouts: Gluing Graphs Along a common part

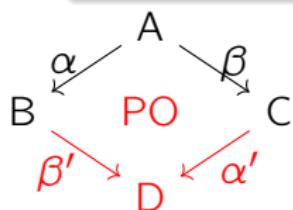
The **pushout** of (α, β) is



Pushouts: Gluing Graphs Along a common part

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

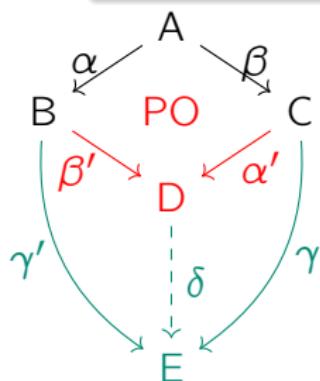
- $\square ABCD$ commutative,



Pushouts: Gluing Graphs Along a common part

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

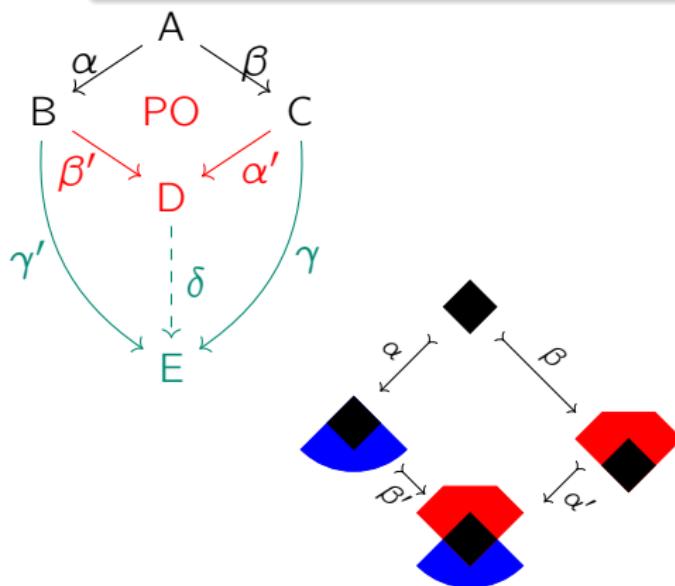
- ▶ $\square ABDC$ commutative,
- ▶ universality: for all (γ, γ') , if $\square ABEC$ commutes, then there is a unique δ such that $\triangle BDE$ and $\triangle CDE$ both commute.



Pushouts: Gluing Graphs Along a common part

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

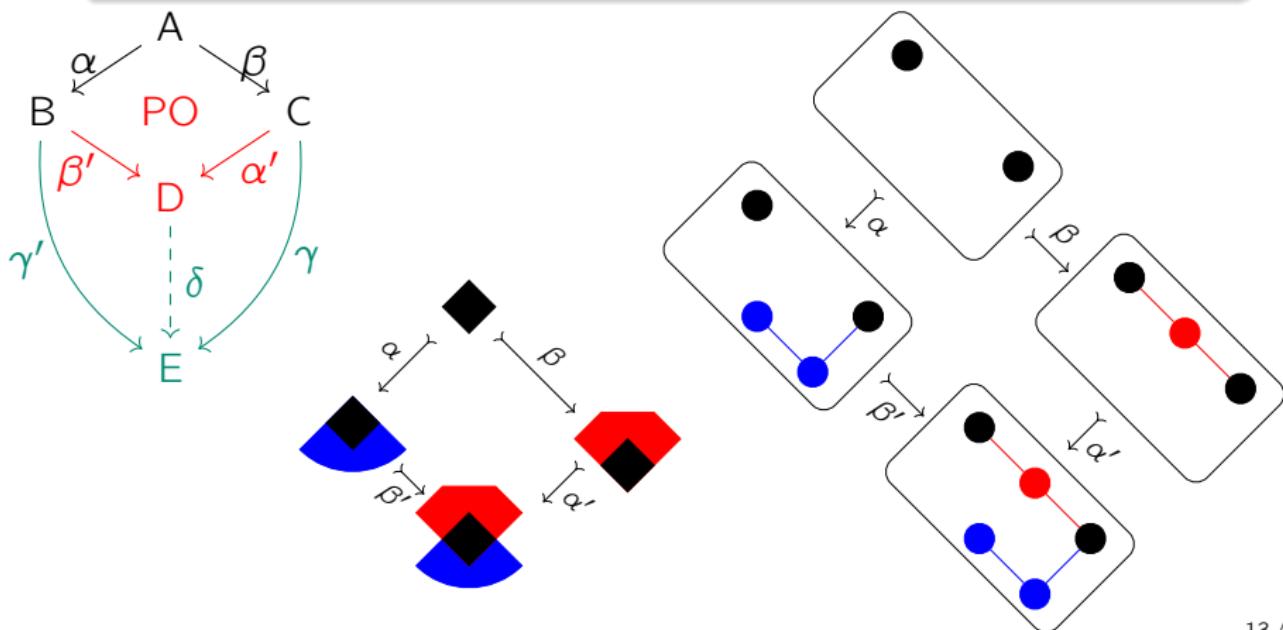
- $\square ABDC$ commutative,
- universality: for all (γ, γ') , if $\square ABEC$ commutes, then there is a unique δ such that $\triangle BDE$ and $\triangle CDE$ both commute.



Pushouts: Gluing Graphs Along a common part

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

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



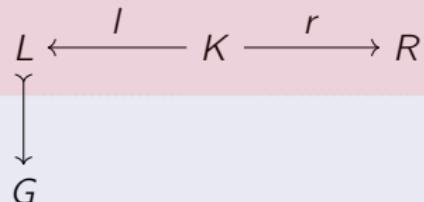
Graph Rewriting with Double-Pushout (DPO)

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

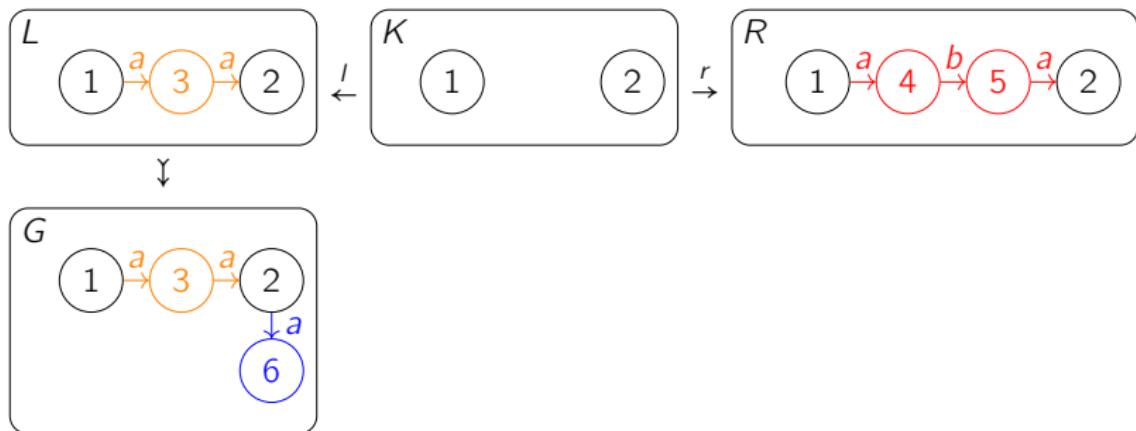
Rewriting rule with interface K



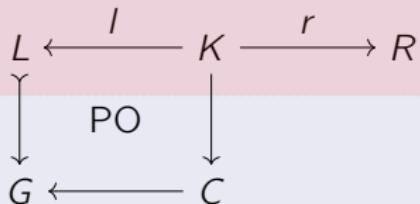
Graph Rewriting with Double-Pushout (DPO)



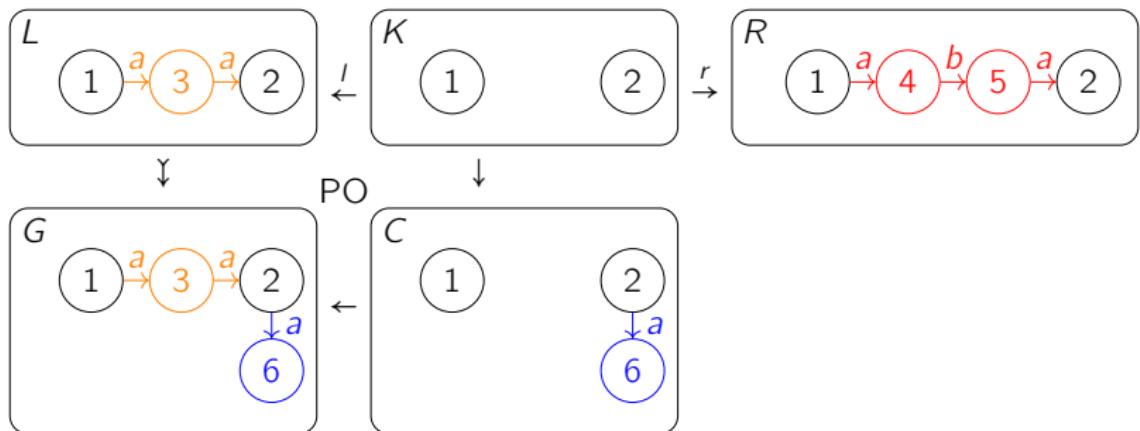
Rewriting rule with interface K



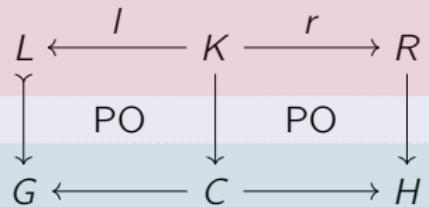
Graph Rewriting with Double-Pushout (DPO)



Rewriting rule with interface K

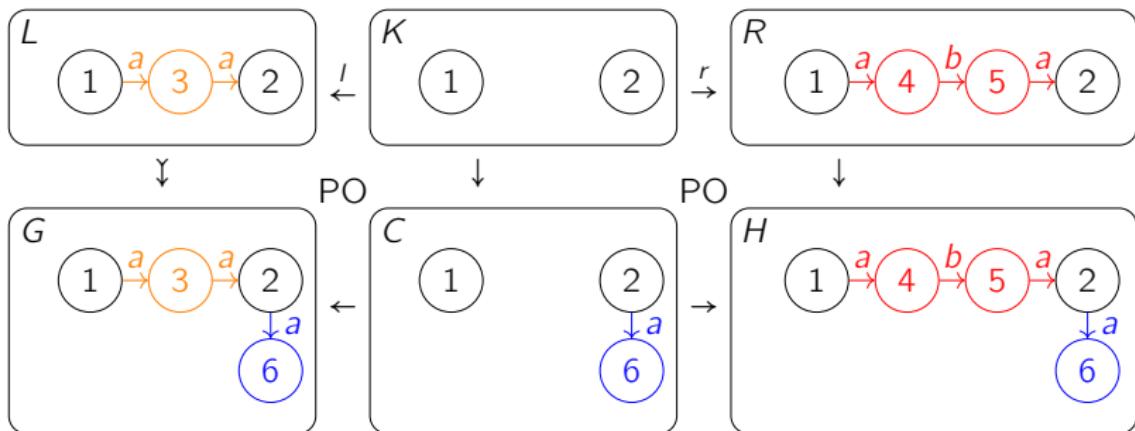


Graph Rewriting with Double-Pushout (DPO)

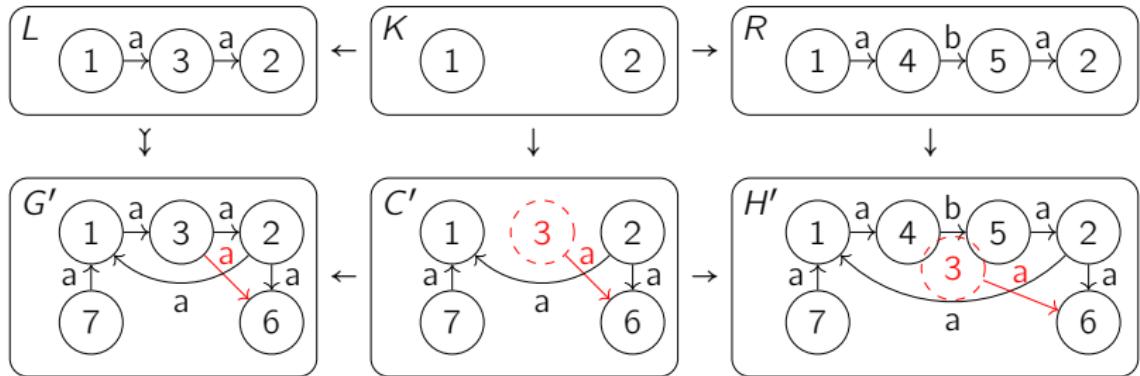


Rewriting rule with interface K

rewriting step $G \Rightarrow H$



An Invalid Rewriting Step



No implicit edge deletion by construction

Preliminaries

Toward greater usability

Toward greater power

LyonParallel—A Tool for Termination of Graph Rewriting

Weighted Type Graph Method [Bru14; Bru+15; EO24b]

Termination by interpretation

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

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

Interpretation:

$$G \rightsquigarrow \mathcal{F}(G, T)$$

$$\rightsquigarrow \text{weight}(\mathcal{F}(G, T))$$

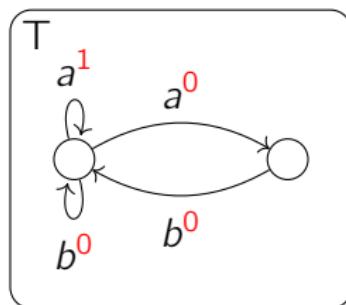
$$\rightsquigarrow \text{aggregator}(\text{weight}(\mathcal{F}(G, T))) \in \mathbb{N}$$

What is the morphism weight?

What is the graph weight?

Weighted Type Graph

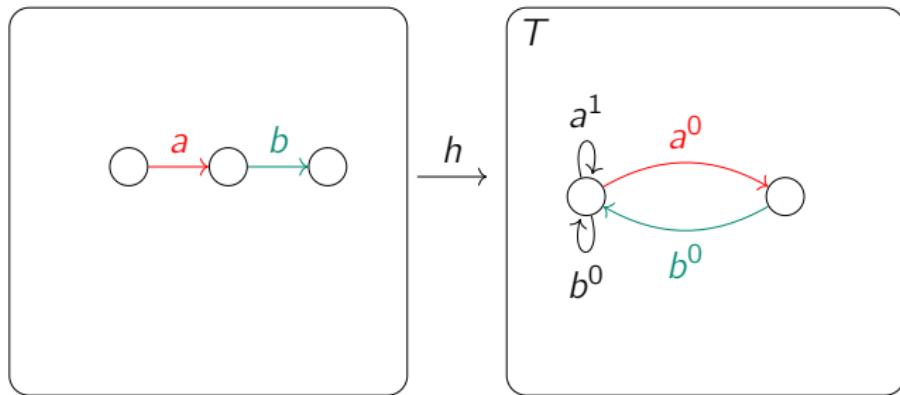
A weighted type graph is a graph with weights on edges.



Morphism Weight

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

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

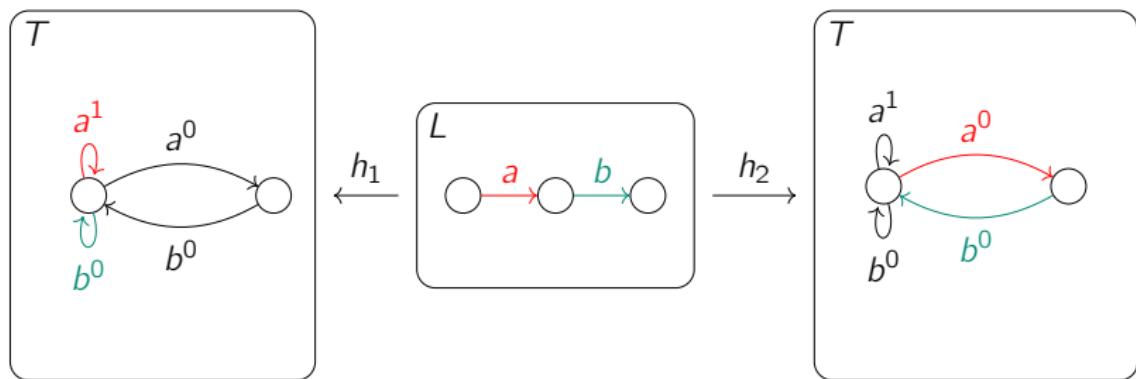


$$\text{weight}_T(h) = 0 + 0 = 0$$

Graph Weight

The weight of a graph L is

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

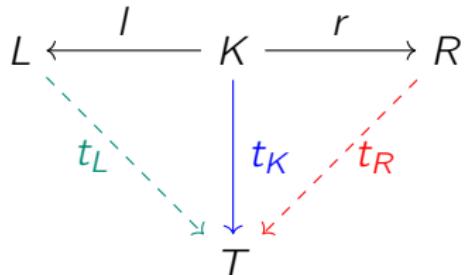


$$\text{weight}_T(h_1) = 1 + 0 = 1$$

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

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

Termination Criterion [Bru+15]



Every rewriting step strictly decreases the weight if

- ▶ 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 $n \in \mathbb{N}$

Assumption:

- ▶ no parallel edges of the same label

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

Challenge:

- ▶ $2^{k^2m} \cdot n^{k^2m}$ possible assignments of weights
- ▶ maximum edge weight hard to guess

Problem of the Size of the Search Space

With natural numbers as weights:

# nodes (k)	# labels (m)	# weights	# possibilities
2	2	2	$\approx 10^4$
3	3	3	$\approx 10^{21}$
3	3	10	$\approx 10^{45}$
3	3	100	$\approx 10^{87}$
4	4	4	$\approx 10^{57}$
4	4	10	$\approx 10^{95}$
4	4	100	$\approx 10^{181}$

Problems can be solved by Z3 in exponential-time with respect to the number of variables $2k^2m$.

Idea

Using positive real numbers as weights

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

Searching for Weighted Type Graphs over $\mathbb{X} \mathbb{R}$

User-specified parameters:

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

Assumption:

- ▶ no parallel edges of the same label

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:

- ▶ ~~there are $2^{k^2 m} \cdot n^{k^2 m}$ possible assignments of weights.~~ There
~~are $2^{k^2 m}$ linear programs which have polynomial-time~~
~~average-case complexity.~~

Complexity Comparison

With weights in \mathbb{N} :

# nodes (k)	# labels (m)	# weights	# possibilities
2	2	2	$\approx 10^4$
3	3	3	$\approx 10^{21}$
3	3	10	$\approx 10^{45}$
3	3	100	$\approx 10^{87}$
4	4	4	$\approx 10^{57}$
4	4	10	$\approx 10^{95}$
4	4	100	$\approx 10^{181}$

With weights in \mathbb{R} :

# nodes (k)	# labels (m)	# variables	# linear programs in \mathbb{R}
2	2	8	$\approx 10^2$
3	3	27	$\approx 10^8$
4	4	64	$\approx 10^{19}$

Linear programs can be solved in polynomial time with respect to the number of variables on average.

Experimental Results

	A	a	T	t	N	n
[EO24a, Example 6.3]					2.74	1.16
[EO24a, Example D.3]	2.25	1.18			2.24	1.18
[Plu95, Example 3.8]	2.95	1.90	2.94	1.87	3.49	1.87
[Plu18, Example 4]	4.26	3.19	4.24	3.13	5.82	timeout
[Plu18, Example 5]	5.54	5.55	5.53	5.50	9.11	5.62
[Bru+15, Example 4]	2.44	2.46	2.47	2.54	4.58	2.46
[Bru+15, Example 5]					7.80	timeout
[Bru+15, Example 6]					9.75	timeout
[Bru14, Example 1]	2.26	1.18			2.24	1.18
[Bru14, Example 4]	2.25	1.22	2.24	1.18	2.25	1.19
[Bru14, Example 5]	4.23	3.23	4.25	3.28	5.82	timeout

“A”, “T”, “N” : different configurations with weights over the natural numbers. “a”, “t”, “n” : corresponding configurations over the real numbers.

Implementation

LyonParallel

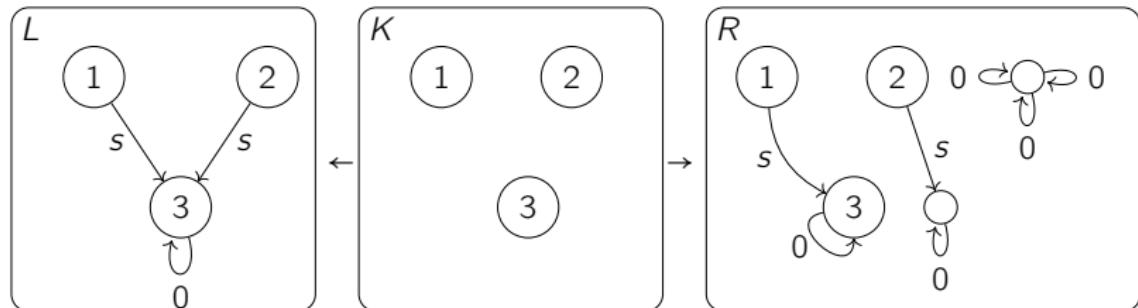
Tool in Ocaml

Relative termination

Search parallel with 6 configurations

Z3 for constraint solving

A Limitation of the Weighted Type Graph Method



All existing automated methods fail.

Intuition: the number of morphisms from $\bullet \xrightarrow{s} \bullet \xleftarrow{s} \bullet$ strictly decreases.

Preliminaries

Toward greater usability

Toward greater power

LyonParallel—A Tool for Termination of Graph Rewriting

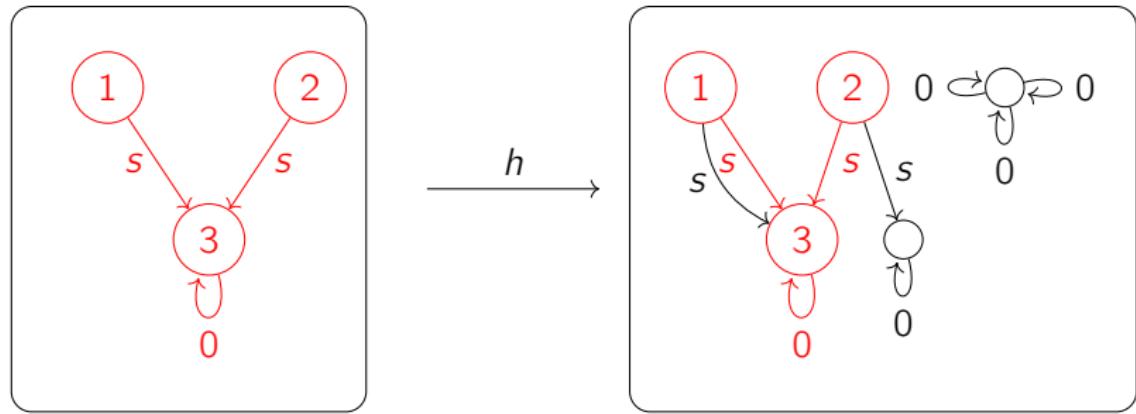
Morphism Counting

Termination by interpretation

Parameter: a graph X

Interpretation of a graph G : number of morphisms from X to G

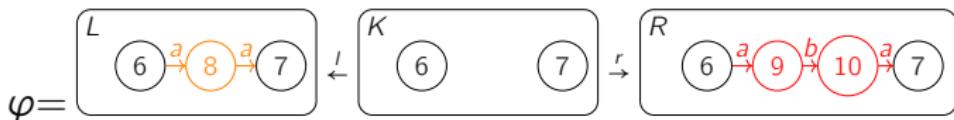
Inclusions: morphisms h with $h(x) = x$ for all x .



Subgraph

Graph Rewriting with Injective Rules

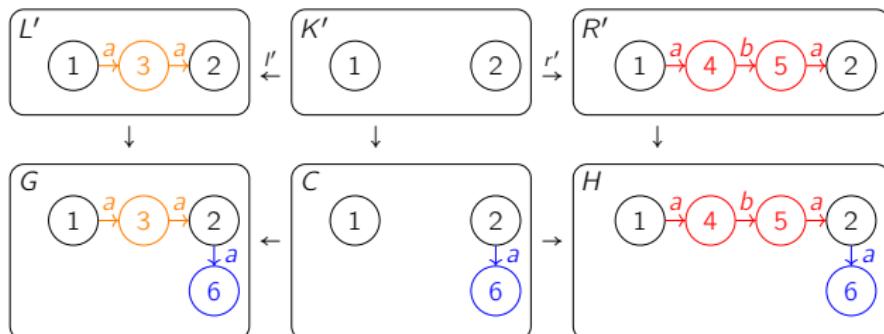
A rewriting rule consists of two inclusions.



An equivalent rewriting rule expresses the same transformation.

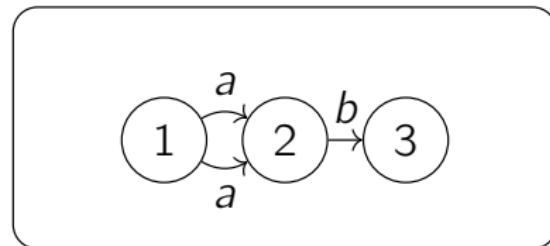


A rewriting step with φ is defined by a DPO diagram with inclusions and φ' .

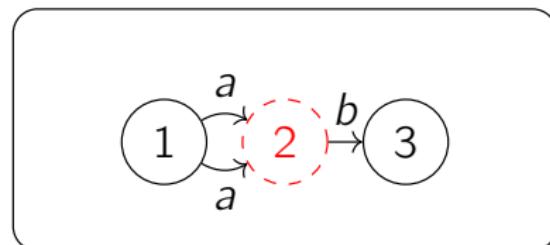


Pre-Graphs

Graph:



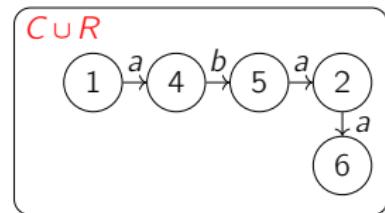
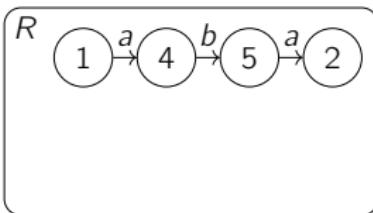
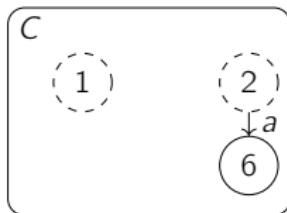
Pre-graphs obtained by removing node 2:



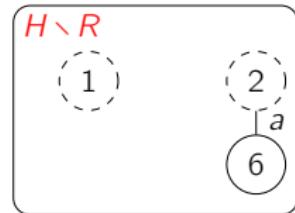
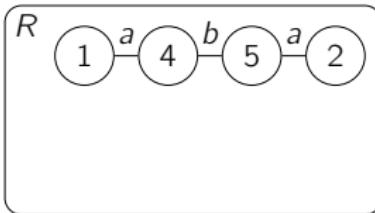
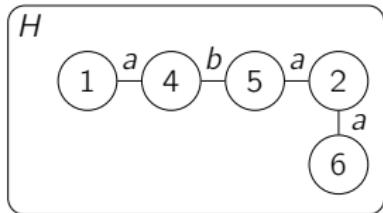
All edges are dangling.

Pre-Graph Operations

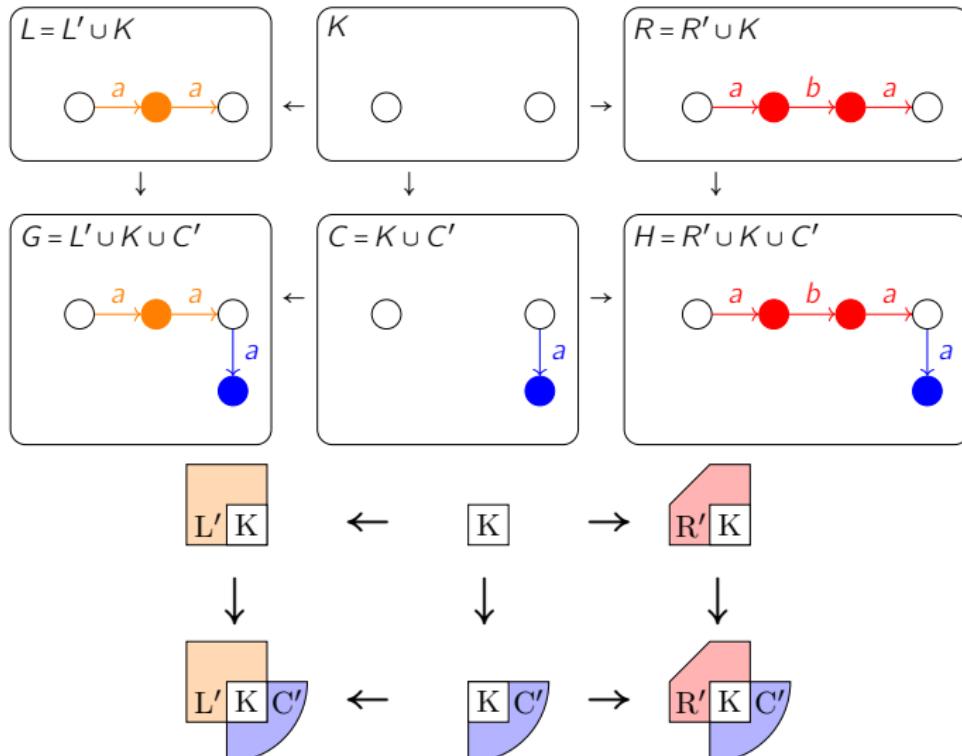
Union of two pre-graphs $C \subseteq G$ and $R \subseteq G$, denoted $C \cup R$.



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



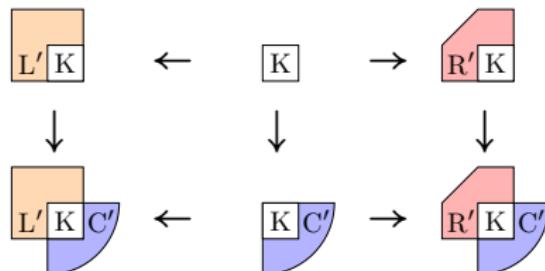
Decomposition of Graphs in Rewriting Steps



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

Morphisms by Image Node Colors

An X -occurrence is an injective morphism from X .

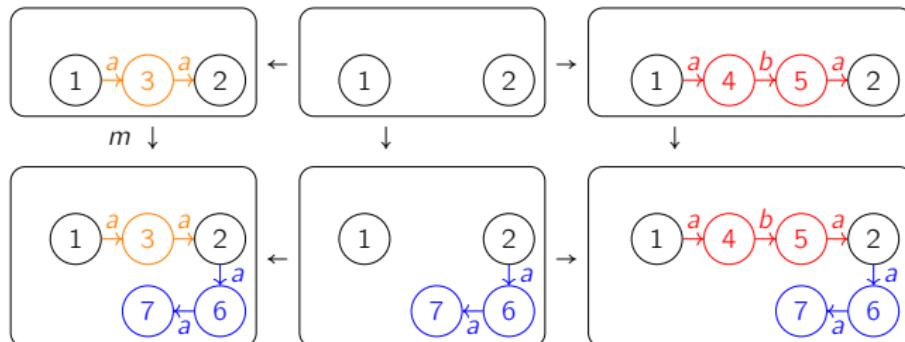


An X -occurrence x is

- ▶ orange if $\text{Im}(x)$ contains only white and orange nodes;
- ▶ red if $\text{Im}(x)$ contains only white and red nodes;
- ▶ blue if $\text{Im}(x)$ contains only white and blue nodes;
- ▶ blue-and-orange if $\text{Im}(x)$ contains both blue and orange nodes;
- ▶ blue-and-red if $\text{Im}(x)$ contains both blue and red nodes.

Morphisms by Image Node Colors

Let X be the graph $\bullet \xrightarrow{a} \bullet \xrightarrow{a} \bullet$.



Blue X -morphisms:

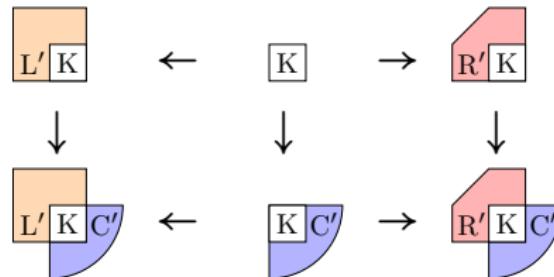
Orange X -morphisms:

Blue-and-Orange X -morphisms:

Red X -morphisms: none.

Blue-and-red X -morphisms:

A Sufficient Condition for Termination

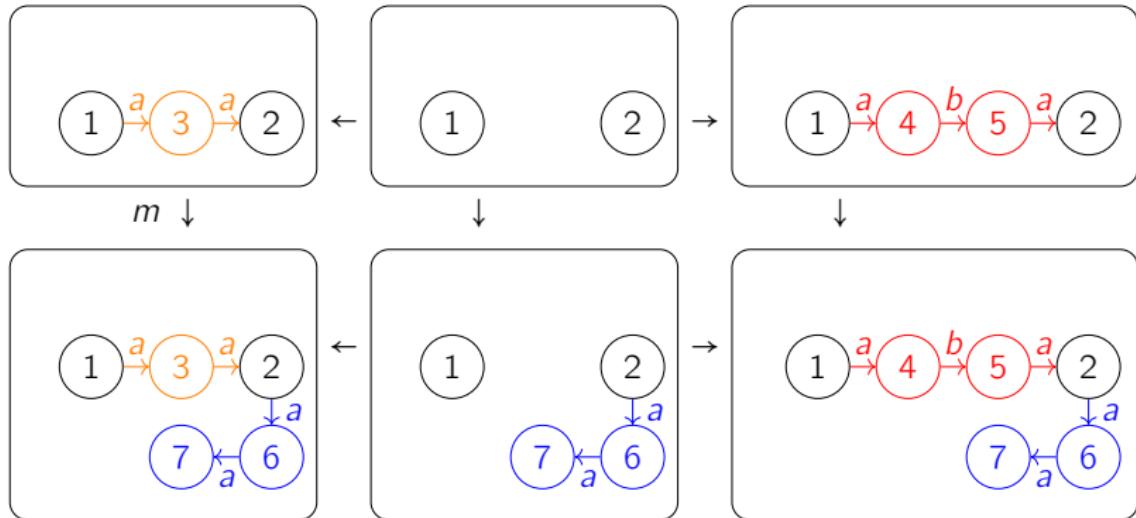


A rule terminates if, for every rewriting step, there are

- ▶ strictly more orange X-morphisms than red X-morphisms,
- ▶ more blue-and-orange X-morphisms than blue-and-red X-morphisms.

Challenge: C' is unknown before rewriting.

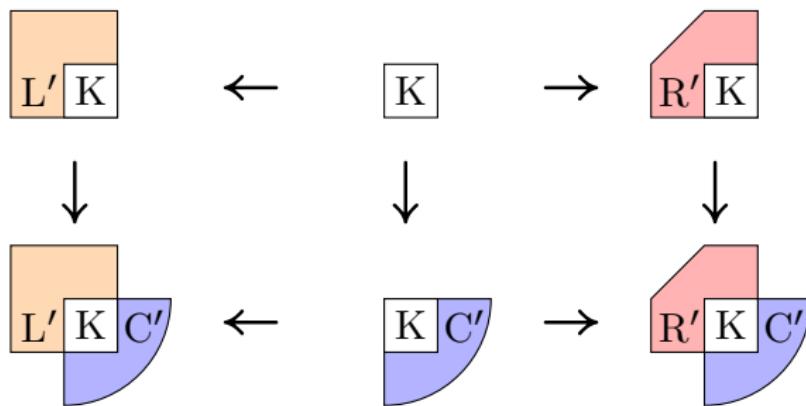
Analysis of Implicit Occurrences



Blue-and-red X-morphisms:

Blue-and-Orange X-morphisms:

A Sufficient Condition for the Second Condition

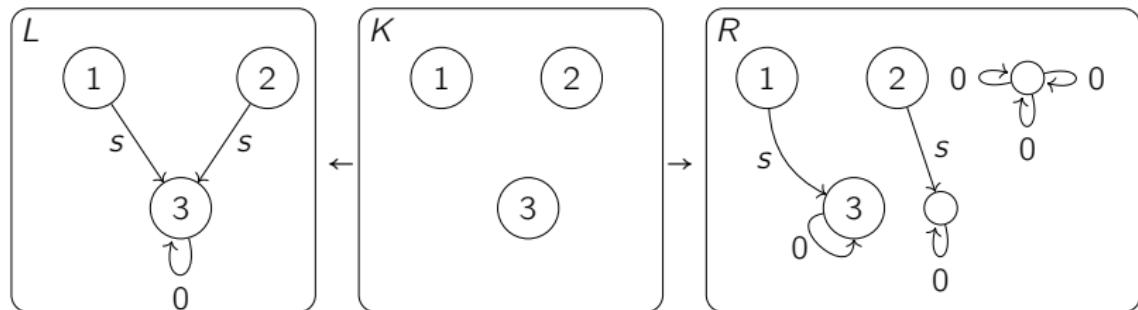


There are more blue-and-orange X-morphisms than blue-and-red X-morphisms, if all subgraphs of $R'K$ that can form an implicit X-occurrence in some rewriting step can be mapped to distinct subgraphs in $L'K$ while preserving elements in K .

Imcomparable with Existing Methods

Fail in some cases where other methods succeed.

Succeed in the following case where other methods fail:



Termination proved by counting morphisms from $\bullet \xrightarrow{s} \bullet \xleftarrow{s} \bullet \dots$.

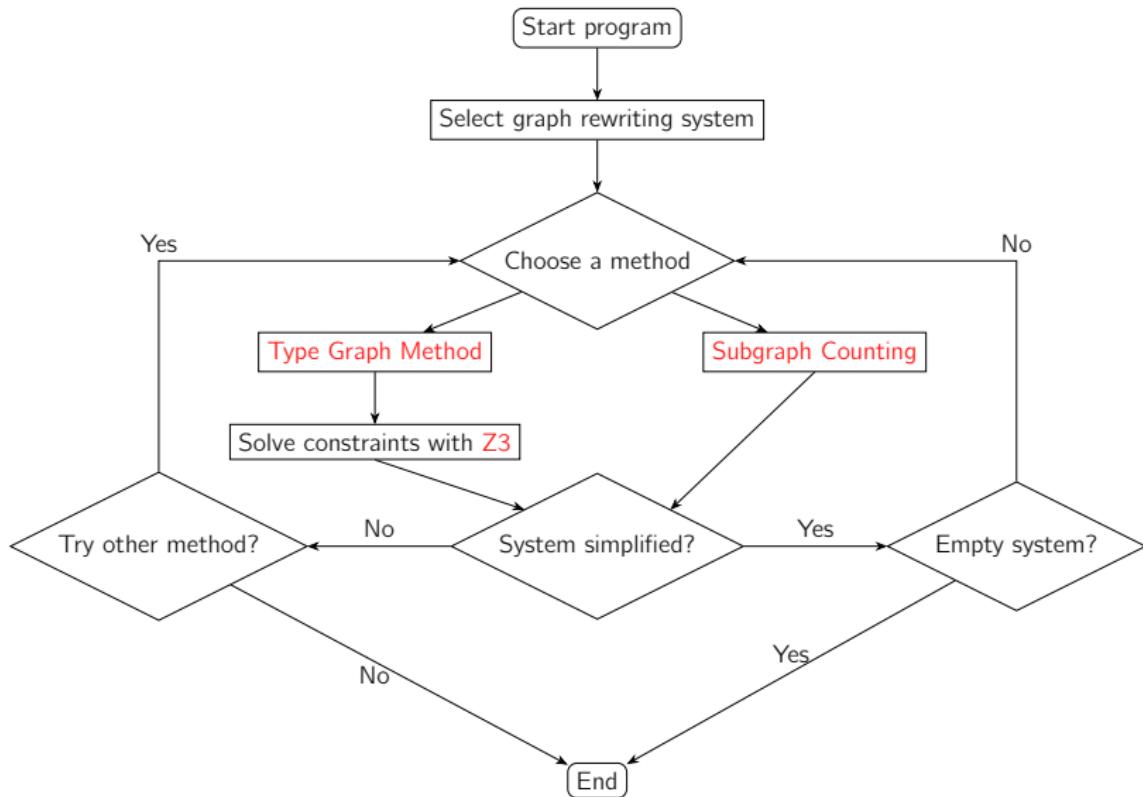
LyonParallel

Automated tool in Ocaml

Iterative elimination of graph rewriting rules

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

Process Flowchart of LyonParallel



Conclusion and Future Work

Contributions

- ▶ Extended the Weighted Type Graph Method to improve usability.
- ▶ Proposed a termination criterion applicable to new cases.
- ▶ Implemented an automated tool for termination analysis.

Future work

- ▶ Formally verify the methods.
- ▶ Generalize Morphism Counting Method to Multiple Forbidden Contexts.
- ▶ Extend the approach to other rewriting frameworks.

References I

- [Bru+15] H. J. Sander Bruggink et al. “Proving Termination of Graph Transformation Systems using Weighted Type Graphs over Semirings”. In: *CoRR abs/1505.01695* (2015). arXiv: 1505.01695.
- [Bru14] H. J. Sander Bruggink. “Towards Process Mining with Graph Transformation Systems”. In: *Graph Transformation*. Ed. by Holger Giese and Barbara König. Cham: Springer International Publishing, 2014, pp. 253–268. ISBN: 978-3-319-09108-2.
- [EO24a] J. Endrullis and R. Overbeek. *Generalized Weighted Type Graphs for Termination of Graph Transformation Systems*. 2024. arXiv: 2307.07601v2 [cs.LO].

References II

- [EO24b] Jorg Endrullis and Roy Overbeek. “Generalized Weighted Type Graphs for Termination of Graph Transformation Systems”. In: *Graph Transformation - 17th International Conference, ICGT 2024, Held as Part of STAF 2024, Enschede, The Netherlands, July 10-11, 2024, Proceedings*. Ed. by Russ Harmer and Jens Kosiol. Vol. 14774. Lecture Notes in Computer Science. Springer, 2024, pp. 39–58. DOI: [10.1007/978-3-031-64285-2_3](https://doi.org/10.1007/978-3-031-64285-2_3).
- [Plu18] Detlef Plump. “Modular Termination of Graph Transformation”. In: *Graph Transformation, Specifications, and Nets - In Memory of Hartmut Ehrig*. Ed. by Reiko Heckel and Gabriele Taentzer. Vol. 10800. Lecture Notes in Computer Science. Springer, 2018, pp. 231–244. DOI: [10.1007/978-3-319-75396-6_13](https://doi.org/10.1007/978-3-319-75396-6_13).

References III

- [Plu95] Detlef Plump. “On Termination of Graph Rewriting”. In: *Graph-Theoretic Concepts in Computer Science, 21st International Workshop, WG '95, Aachen, Germany, June 20-22, 1995, Proceedings*. Ed. by Manfred Nagl. Vol. 1017. Lecture Notes in Computer Science. Springer, 1995, pp. 88–100. DOI: [10.1007/3-540-60618-1_68](https://doi.org/10.1007/3-540-60618-1_68).