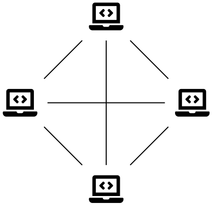
Automated Termination Proving: Contributions to Graph Rewriting via Extended Weighted Type Graphs and Morphism Counting

Qi QIU

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

Motivation & Goal

Distributed systems:



Failures can be catastrophic:

Ensuring correctness is difficult.

- The Needham-Schroeder protocol proved insecure 17 years after its publication.

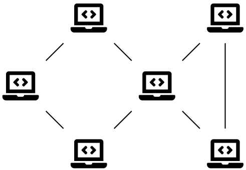
This thesis: automated verification.

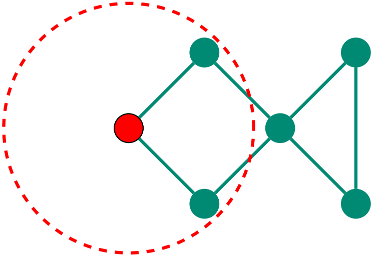
- Minimal user effort
- No expertise required
- Mathematically rigorous

Graph Transformation

Modelization of distributed systems

System configurations: graphs



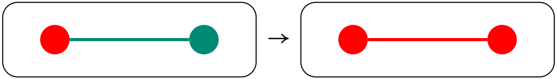


Algorithm behaviors:

graph transformation according to local knowledge

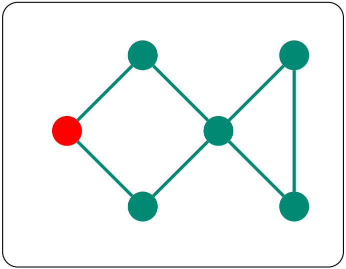
Graph Transformation

Graph transformation rule:



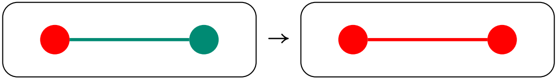
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



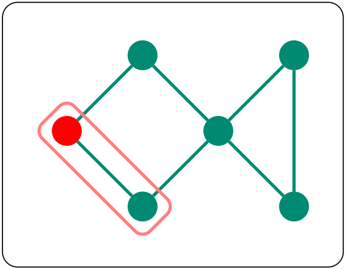
Graph Transformation

Graph transformation rule:



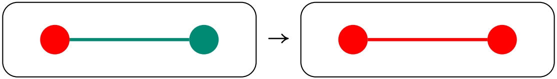
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



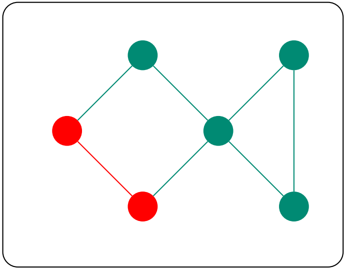
Graph Transformation

Graph transformation rule:



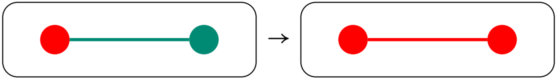
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



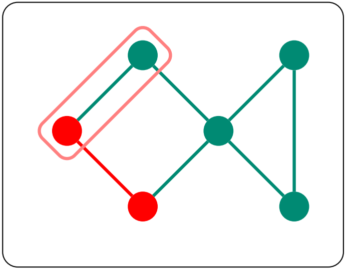
Graph Transformation

Graph transformation rule:



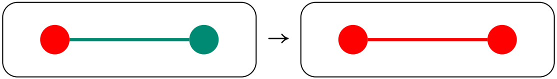
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



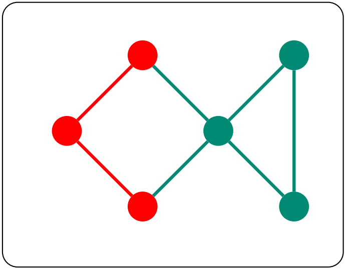
Graph Transformation

Graph transformation rule:



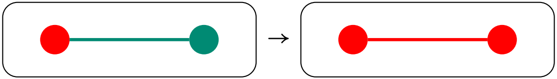
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



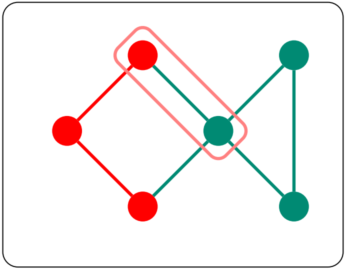
Graph Transformation

Graph transformation rule:



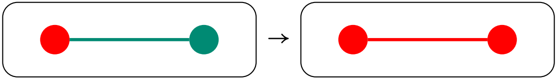
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



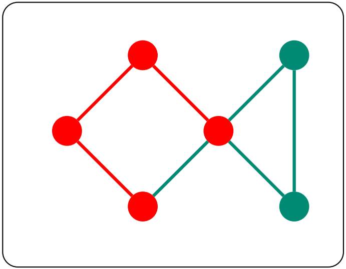
Graph Transformation

Graph transformation rule:



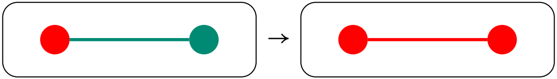
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



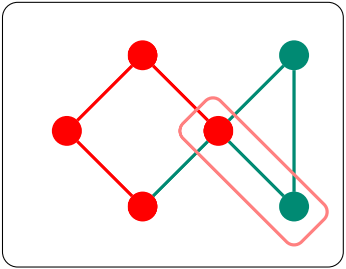
Graph Transformation

Graph transformation rule:



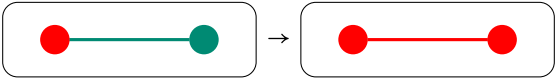
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



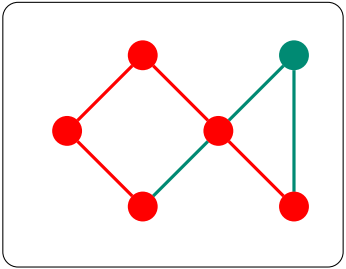
Graph Transformation

Graph transformation rule:



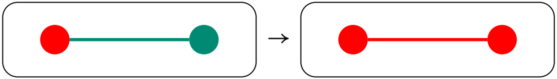
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



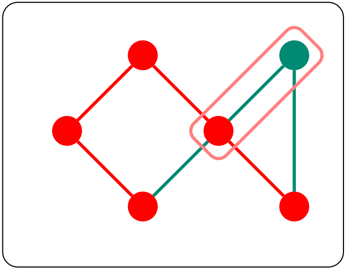
Graph Transformation

Graph transformation rule:



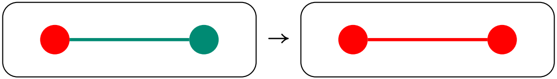
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



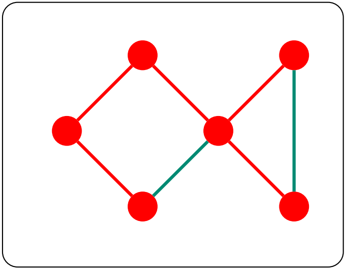
Graph Transformation

Graph transformation rule:



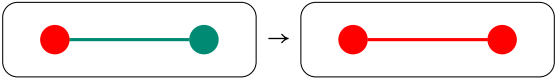
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



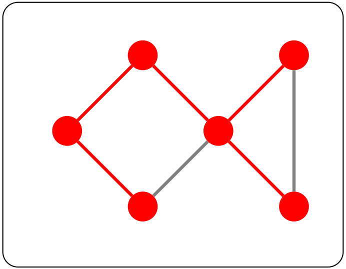
Graph Transformation

Graph transformation rule:



Replace the left-hand side by the right-hand side.

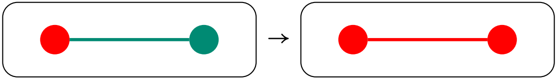
Spanning-tree construction:



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

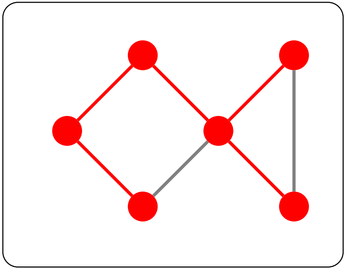
Graph Transformation

Graph transformation rule:



Replace the left-hand side by the right-hand side.

Spanning-tree construction:



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

Does the transformation process terminate for any initial graph?

Termination of Graph Transformation Systems

- No graph G_0 can be transformed forever

$$G_0 \Rightarrow G_1 \Rightarrow \cdots$$

- Aligns with the notion of program termination:

  "every execution (on any input) halts."

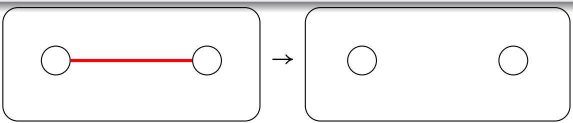
- Undecidable in general [10]
  - Automated techniques for specific subclasses

Termination by interpretations [12, 1]

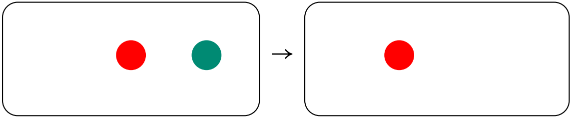
Interpret graphs as natural numbers.

Show each transformation step strictlydecreases the value.

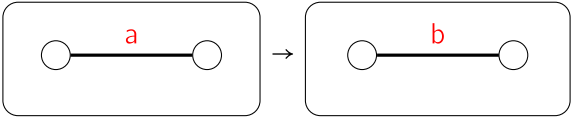
Number of edges:



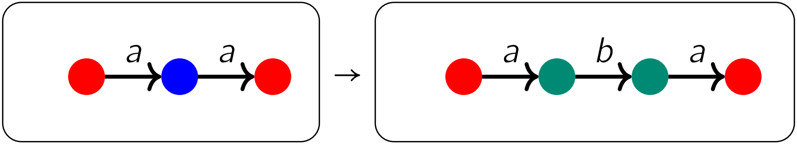
Number of nodes:



Number of edges labeled by *a*:



Limitation



Number of nodes/edges/labels do not decrease.

Can its termination be proved by interpretations?

- Need a formal definition of graph transformations.

Structure of the Remainder

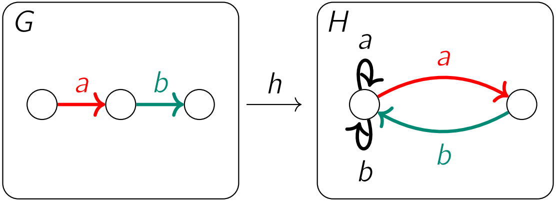
Graph Rewriting with Double-Pushout (DPO)

Toward Greater Usability

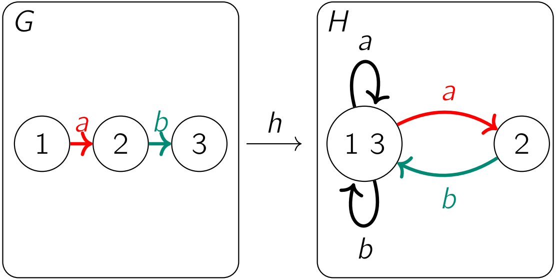
Toward Greater Power

LyonParallel—A Tool for Termination of Graph Rewriting

Graph Morphisms: Structure-Preserving Functions



Colors show edge correspondence.

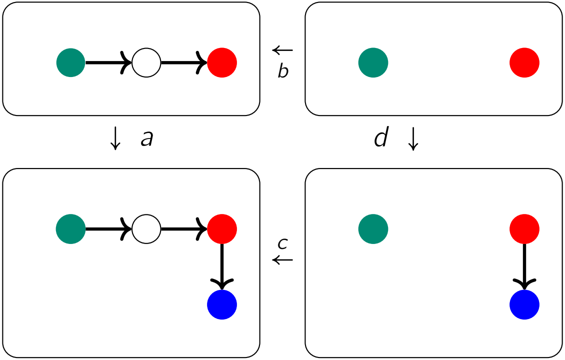


Numbers show node correspondence.

Commutative Diagram

$$
\begin{tikzcd}
L \arrow[d, "a"] & K \arrow[l, "b"'] \arrow[d, "d"'] \\
G & C \arrow[l, "c"]
\end{tikzcd}
$$

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



Pushouts: Gluing Graphs Along an Interface

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

- $\square$ABDC commutes,

$$\begin{tikzcd} A \arrow[r, "\beta"] \arrow[d, "\alpha"'] & C \arrow[d, "\alpha'"] \\ B \arrow[r, "\beta'"] & D \end{tikzcd}$$

$\square$ABDC: pushout square
D: pushout object

Pushouts: Gluing Graphs Along an Interface

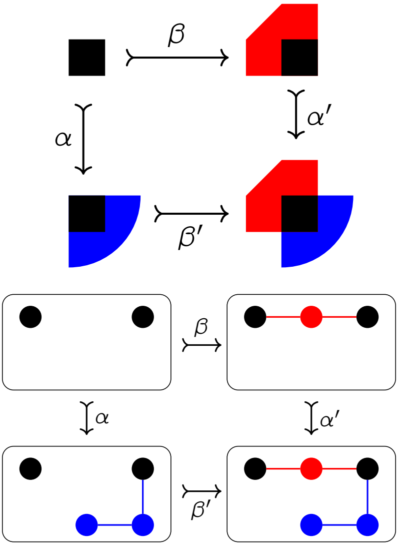
The **pushout** of (α, β) is (β', α') with

- $\square$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.

$$
\begin{tikzcd}
A \arrow[r, "\beta"] \arrow[d, "\alpha"'] & C \arrow[d, "\alpha'"'] \arrow[ddr, bend left, "\gamma'"'] & \\
B \arrow[r, "\beta'"] \arrow[drr, bend right, "\gamma"] & D \arrow[dr, dashed, "\delta"] & \\
& & E
\end{tikzcd}
$$

$\square$ABDC: pushout square
D: pushout object

Pushouts: Gluing Graphs Along an Interface



Graph Rewriting with Double-Pushout (DPO)

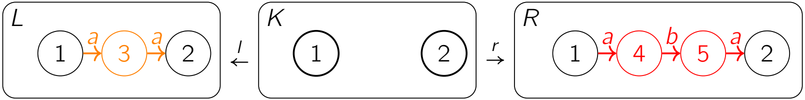
The first algebraic approach to graph rewriting [4]

One of the most studied approaches to graph rewriting [5]

Graph Rewriting with Double-Pushout (DPO) [4]

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

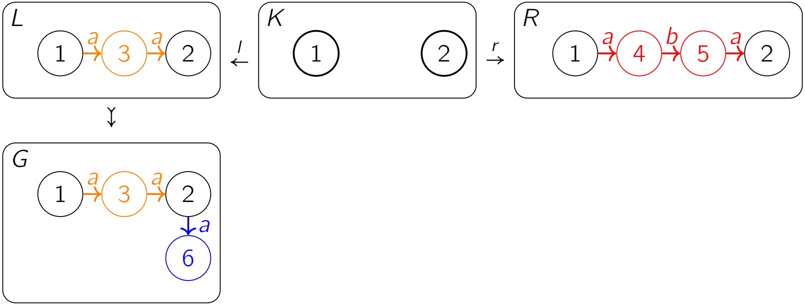
Rewriting rule with interface K



Graph Rewriting with Double-Pushout (DPO) [4]

$$\begin{tikzcd} L \arrow[d, tail] & K \arrow[l, "l"'] \arrow[r, "r"] & R \\ G & & \end{tikzcd}$$

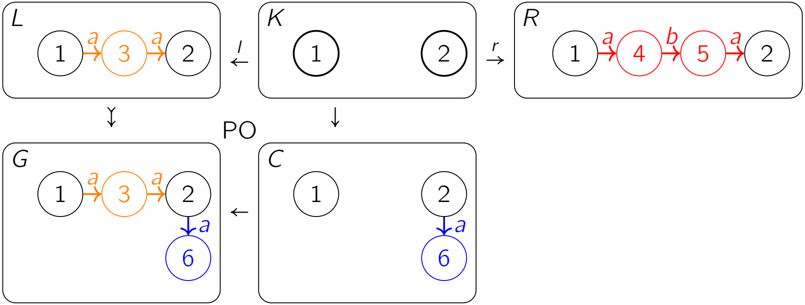
Rewriting rule with interface K



Graph Rewriting with Double-Pushout (DPO) [4]

$$\begin{tikzcd} L \arrow[d, tail] \arrow[dr, phantom, "\mathrm{PO}"] & K \arrow[l, "l"'] \arrow[r, "r"] \arrow[d] & R \\ G & C \arrow[l] & \end{tikzcd}$$

Rewriting rule with interface K

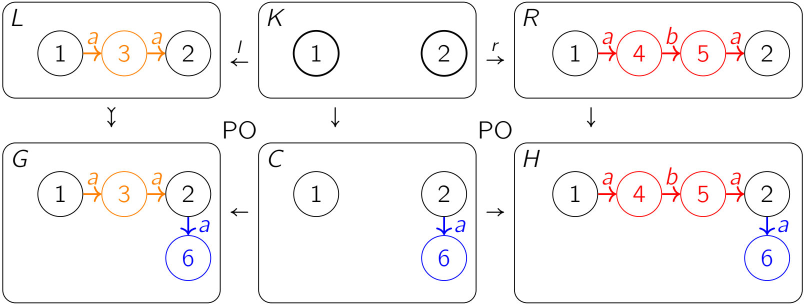


Graph Rewriting with Double-Pushout (DPO) [4]

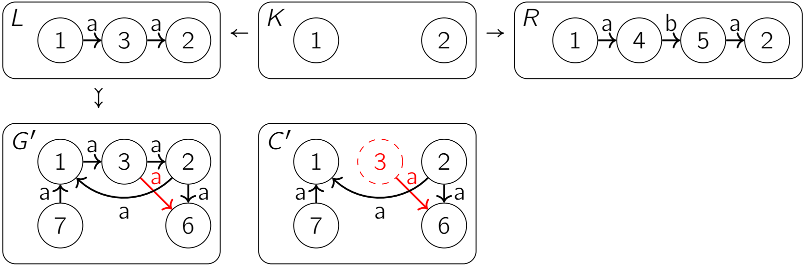
$$\begin{tikzcd} L \arrow[d, tail] \arrow[dr, phantom, "\mathrm{PO}"] & K \arrow[l, "l"'] \arrow[r, "r"] \arrow[d] \arrow[dr, phantom, "\mathrm{PO}"] & R \arrow[d] \\ G & C \arrow[l] \arrow[r] & H \end{tikzcd}$$

Rewriting rule with interface K

rewriting step $G \Rightarrow H$



An Invalid Rewriting Step



Weighted Type Graph Method [2, 3, 7]

Termination by interpretation

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

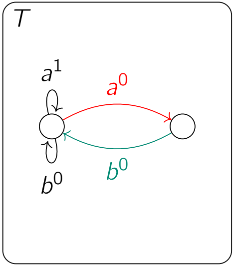
Terminology: every graph is "typed" as morphisms to T

Interpretation:

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

How to choose the type graph T?
What is the morphism weight?
What is the graph weight?

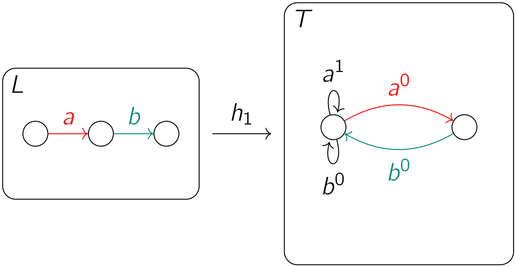
Type Graph with Weights on Edges



Morphism Weight

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

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

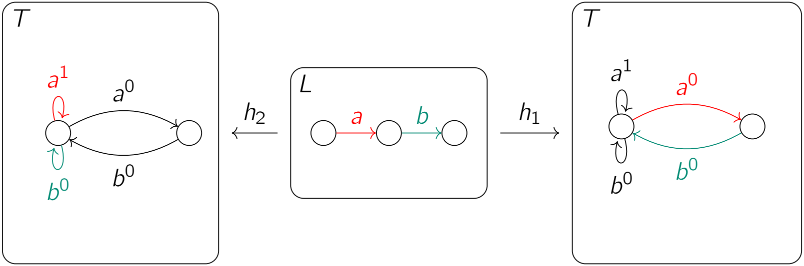


$$\mathrm{weight}_T(h_1) = 0 + 0 = 0$$

Graph Weight

The weight of a graph L is

$$\min\{h : L \to T \mid \mathrm{weight}_T(h)\}$$



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

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

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

Termination Criterion [3]

$$\begin{tikzcd}
L & K \arrow[l, "l"'] \arrow[r, "r"] \arrow[d, "t_K"] & R \arrow[dl, dashed, "t_R"] \\
& T \arrow[from=1-1, dashed, "t_L"'] &
\end{tikzcd}$$

A rule terminates if there is T such that for all t_K, if there is t_L such that $\triangle KLT$ commutes, then

$$\min\{\mathrm{weight}_T(t_L) \mid t_L.\ \triangle\ KLT \text{ commutes}\}$$
$$> \min\{\mathrm{weight}_T(t_R) \mid t_R.\ \triangle\ KRT \text{ commutes}\}$$

How to find a suitable weighted type graph?

Searching for Weighted Type Graphs over $\mathbb{N}$ [12, 3]

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

Challenge:

- expertise: impossible to guess k and n
- complexity: $2^{k^2m} \cdot n^{k^2m}$ possible assignments of weights

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	$\approx 10^{21}$
4	4	4	$\approx 10^{57}$
5	5	5	$\approx 10^{125}$

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

Usability Improvement

Using positive real numbers as weights [8]

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

Complexity Comparison

m: number of labels
Parameters:

- k nodes

With weights in $\mathbb{N}$:

- User-specified parameter: maximum weight $n \in \mathbb{N}$
- Satisfiability of an existential Presburger arithmetic theory with $k^2 m$ variables
- Exponential-time

With weights in $\mathbb{R}$:

- Solve a linear program in $\mathbb{R}$ with $k^2 m$ variables
- Polynomial time on average (e.g by Z3)

Experimental Results

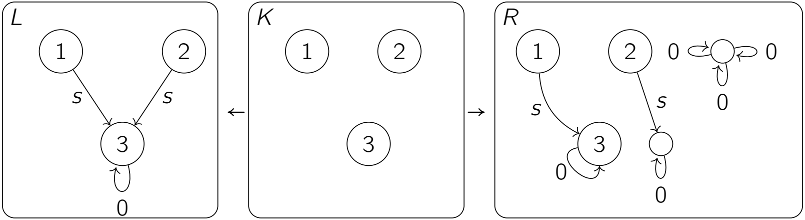
	$A_{\mathbb{N}}$	$a_{\mathbb{R}}$	$T_{\mathbb{N}}$	$t_{\mathbb{R}}$	$N_{\mathbb{N}}$	$n_{\mathbb{R}}$
[6, Example 6.3]					2.74	1.16
[6, Example D.3]	2.25	1.18			2.24	1.18
[10, Example 3.8]	2.95	1.90	2.94	1.87	3.49	1.87
[9, Example 4]	4.26	3.19	4.24	3.13	5.82	timeout
[9, Example 5]	5.54	5.55	5.53	5.50	9.11	5.62
[3, Example 4]	2.44	2.46	2.47	2.54	4.58	2.46
[3, Example 5]					7.80	timeout
[3, Example 6]					9.75	timeout
[2, Example 1]	2.26	1.18			2.24	1.18
[2, Example 4]	2.25	1.22	2.24	1.18	2.25	1.19
[2, Example 5]	4.23	3.23	4.25	3.28	5.82	timeout

Implementation

LyonParallel

Search parallel with 6 configurations

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 L strictly decreases.

Capability Improvement: Morphism Counting

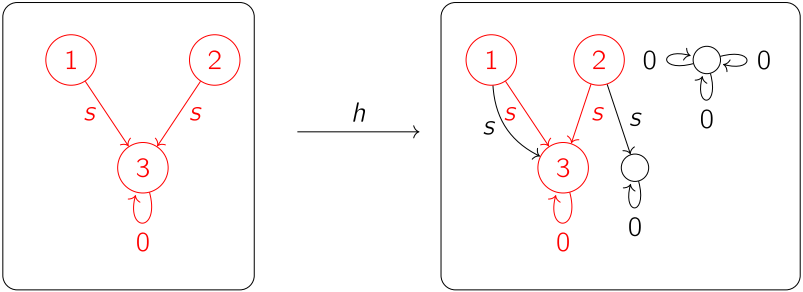
Termination by interpretation

Parameter: graph X

Interpretation:

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

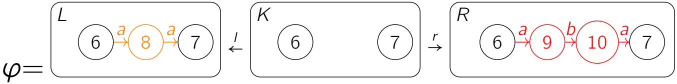
Inclusions



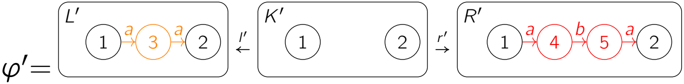
Subgraph

Graph Rewriting Systems

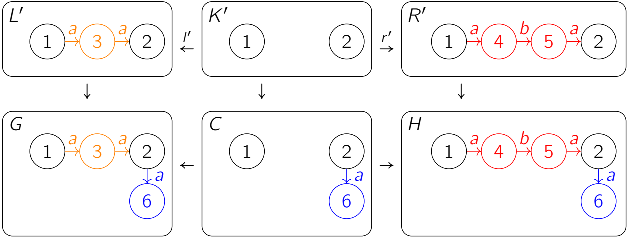
A rewriting rules consists of two inclusions.



An equivalent rewriting rule expresses the same transformation.

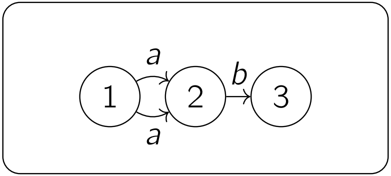


A rewriting step with φ is defined by a DPO diagram with inclusions and φ'.

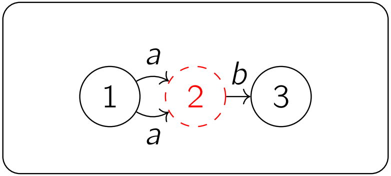


Pre-Graphs

Graph:

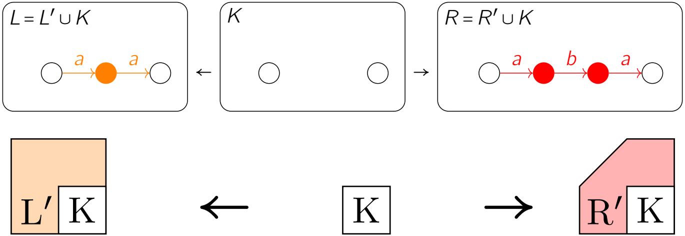


Pre-graphs obtained by removing node 2:

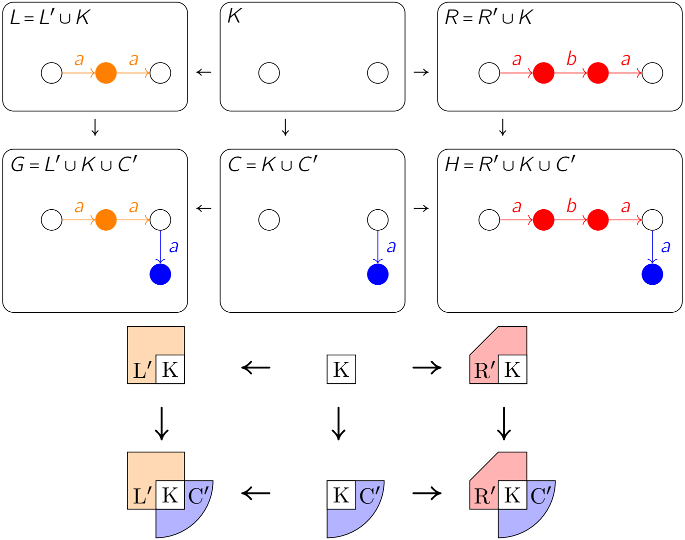


All edges are dangling.

Decomposition of Graphs in Rewriting Rules



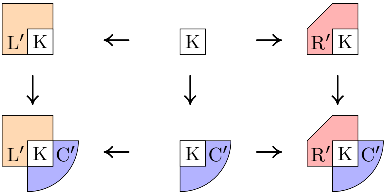
Decomposition of Graphs in Rewriting Steps



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.

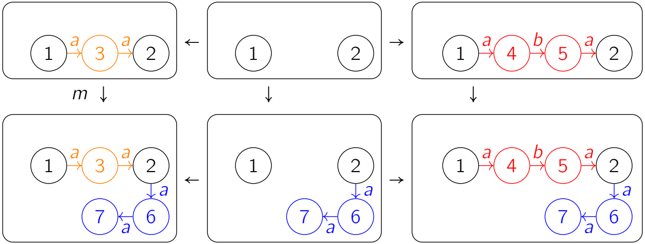


X-occurrence are classified by the colors of their image nodes:

- white: only white;
- orange: only white and at least one orange;
- blue: only white and at least one blue;
- red: only white and at least one red;
- blue-and-orange: at least one blue and at least one orange;
- blue-and-red: at least one blue and at least one red

Morphisms by Image Node Colors

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

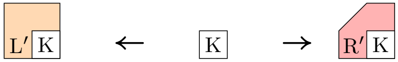


Blue X-occurrence: $2 \xrightarrow{a} 6 \xrightarrow{a} 7$

Red X-occurrences: none.

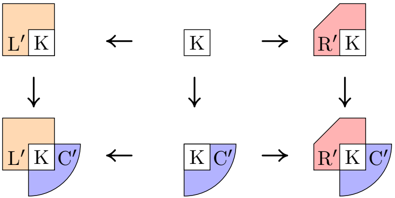
Blue-and-red X-occurrences: $5 \xrightarrow{a} 2 \xrightarrow{a} 6$

A New Sufficient Condition for Termination [11]



terminates if

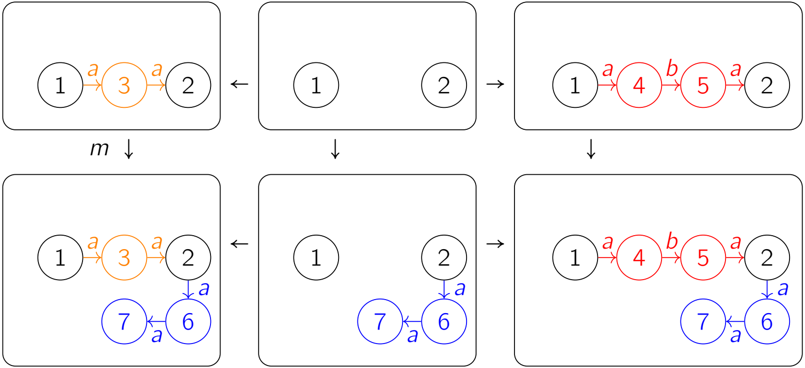
- strictly more orange X-occurrences than red X-occurrences,
- for every rewriting step:



has more blue-and-orange X-occurrences than blue-and-red X-occurrences.

Challenge: check the second condition under the unknown C’

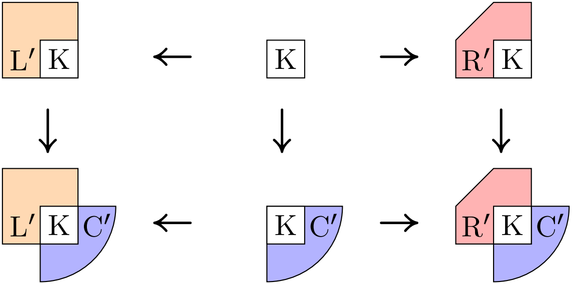
Analysis of Implicit Occurrences

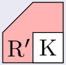


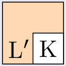
Blue-and-red X-occurrences: $5 \xrightarrow{a} 2 \xrightarrow{a} 6$

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

Sufficient Condition for the Second Condition [11]

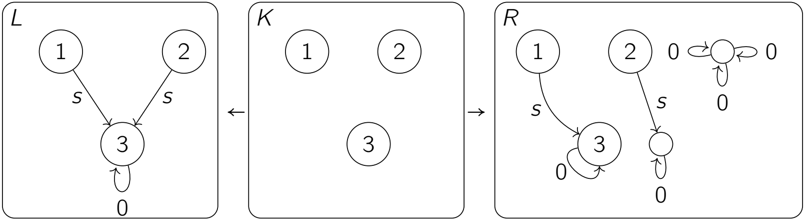


There are more blue-and-orange X-morphisms than blue-and-red X-morphisms, if all subgraphs of 

that can form an blue-and-red X-occurrence in any rewriting step can be mapped to distinct subgraphs in 

while preserving elements in $\boxed{\mathrm{K}}$.

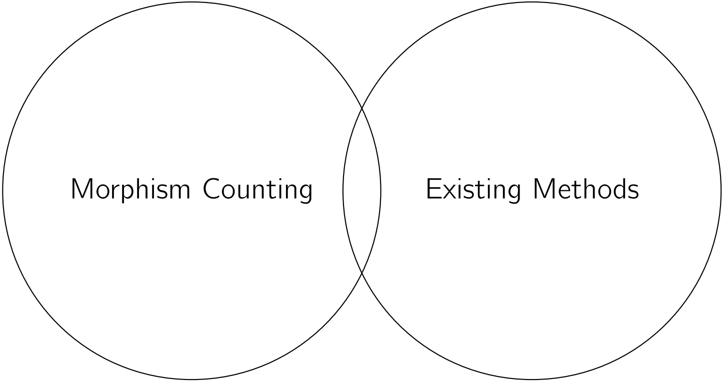
Termination of Motivating Example



Existing automated methods fail.

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

Imcomparable with Existing Methods



Succeed in some cases where all existing automated methods fail.
Fail in some cases where other methods succeed.

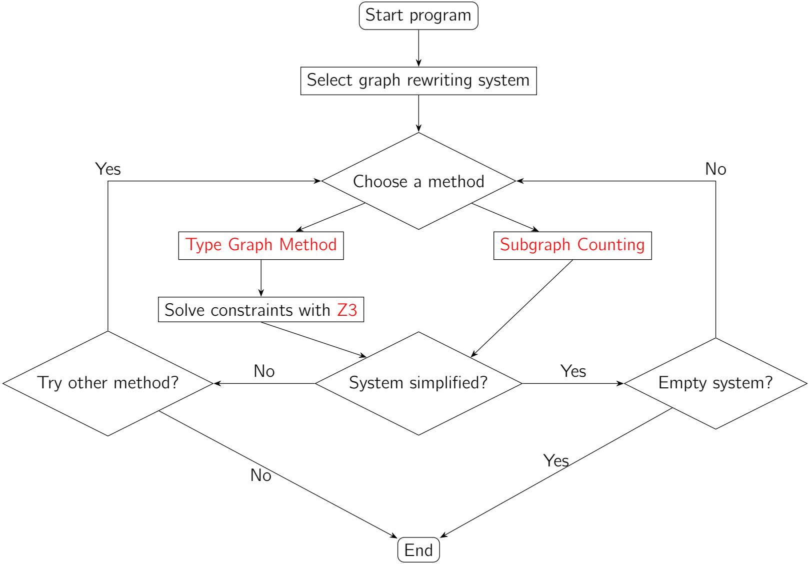
LyonParallel

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.
- Extended Morphism Counting to count morphisms with a forbidden context.
- Implemented an automated tool for termination analysis.

Future work

- Short term: Morphism Counting with forbidden contexts.
- Mid term: Certificate-generation mechanism.
- Long term: Extension to other graph rewriting frameworks (e.g., PBPO+)

References I

[1] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, 1998. ISBN: 978-0-521-45520-6.

[2] 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.

[3] 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.

References II

[4] Hartmut Ehrig, Michael Pfender, and Hans Jurgen Schneider. “Graph-Grammars: An Algebraic Approach”. In: *14th Annual Symposium on Switching and Automata Theory, Iowa City, Iowa, USA, October 15-17, 1973*. IEEE Computer Society, 1973, pp. 167–180. DOI: 10.1109/SWAT.1973.11.

[5] Hartmut Ehrig et al. *Fundamentals of Algebraic Graph Transformation*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2006. ISBN: 978-3-540-31187-4. DOI: 10.1007/3-540-31188-2.

[6] J. Endrullis and R. Overbeek. *Generalized Weighted Type Graphs for Termination of Graph Transformation Systems*. 2024. arXiv: 2307.07601v2 [cs.LO].

References III

[7] 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.

[8] Salvador Lucas. "On the relative power of polynomials with real, rational, and integer coefficients in proofs of termination of rewriting". In: *Appl. Algebra Eng. Commun. Comput.* 17.1 (2006), pp. 49–73. DOI: 10.1007/S00200-005-0189-5.

References IV

[9] 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.

[10] 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.

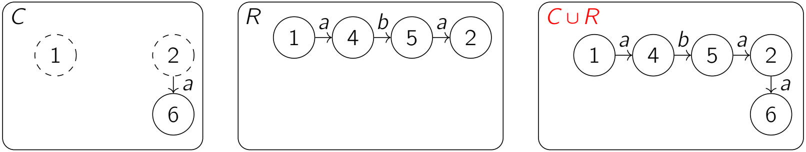
References V

[11] Qi Qiu. "Termination of Injective DPO Graph Rewriting Systems Using Subgraph Counting". In: *Graph Transformation*. Ed. by Jörg Endrullis and Matthias Tichy. Cham: Springer Nature Switzerland, 2025, pp. 3–23. ISBN: 978-3-031-94706-3.

[12] Hans Zantema, Barbara König, and H. J. Sander Bruggink. "Termination of Cycle Rewriting". In: *Rewriting and Typed Lambda Calculi - Joint International Conference, RTA-TLCA 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*. Ed. by Gilles Dowek. Vol. 8560. Lecture Notes in Computer Science. Springer, 2014, pp. 476–490. DOI: 10.1007/978-3-319-08918-8_33.

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 $H \setminus R$.