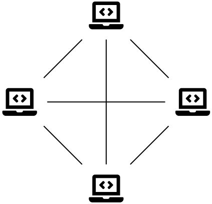
Automated Termination Proving: Contributions to Graph Rewriting via Extended Weighted Type Graphs and Morphism Counting

Qi QIU

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

Motivation & Goal

Distributed systems:



Ensuring correctness is difficult.

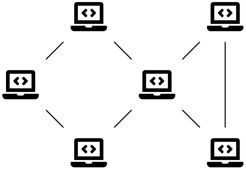
- The Needham-Schroeder protocol proved insecure 17 years after its publication.

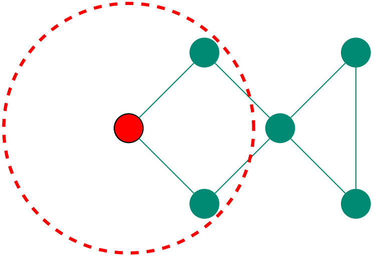
Failures can be catastrophic:

This thesis: automated verification.

- Minimal user effort
- No expertise required
- Mathematically rigorous

Graph Transformation





Modelization of distributed systems

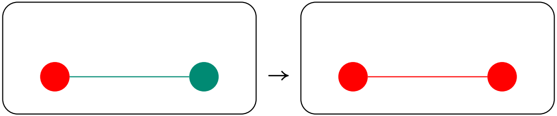
System configurations: graphs

Algorithm behaviors:

graph transformation according to local knowledge

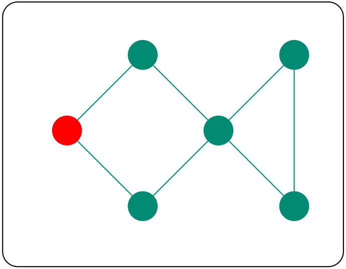
Graph Transformation

Graph transformation rule:



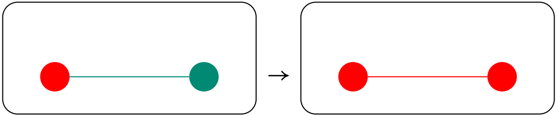
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



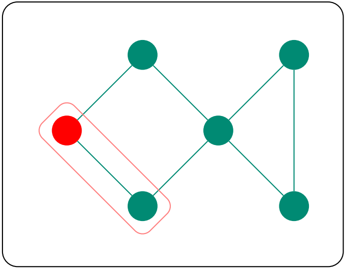
Graph Transformation

Graph transformation rule:



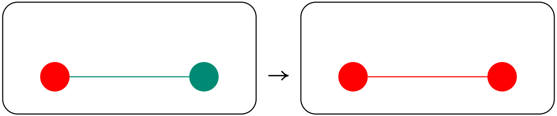
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



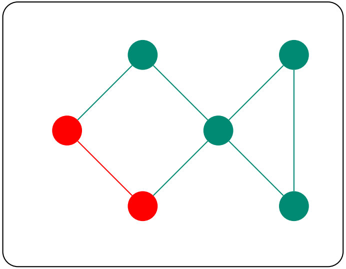
Graph Transformation

Graph transformation rule:



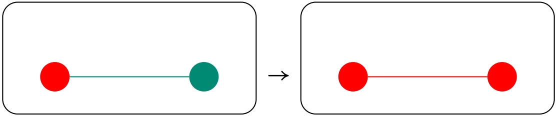
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



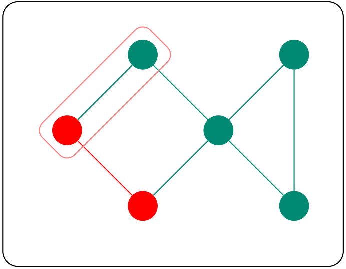
Graph Transformation

Graph transformation rule:



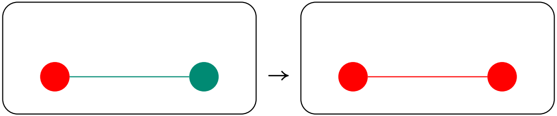
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



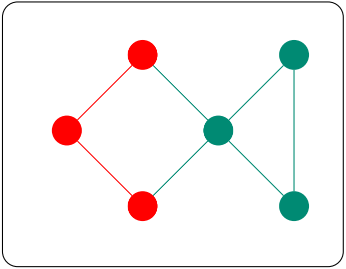
Graph Transformation

Graph transformation rule:



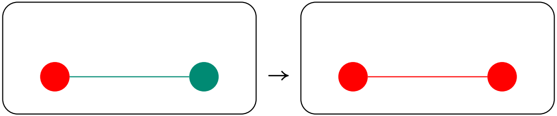
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



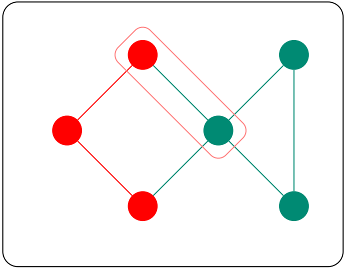
Graph Transformation

Graph transformation rule:



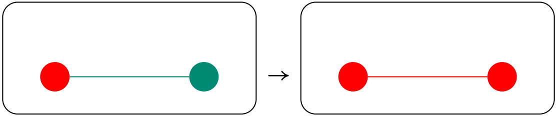
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



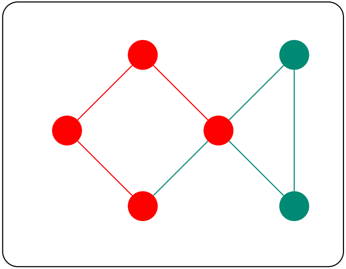
Graph Transformation

Graph transformation rule:



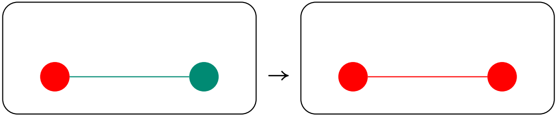
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



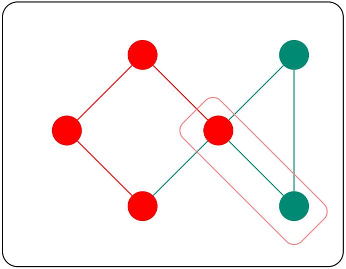
Graph Transformation

Graph transformation rule:



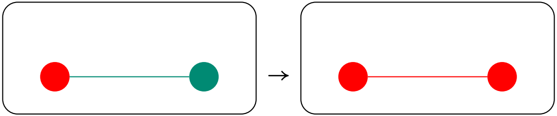
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



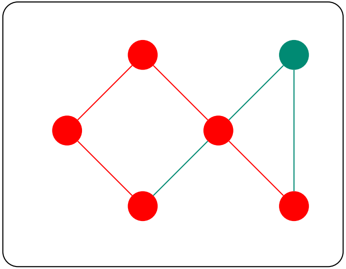
Graph Transformation

Graph transformation rule:



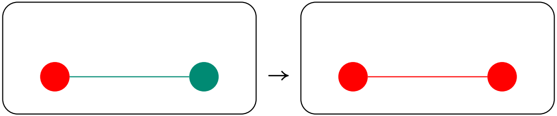
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



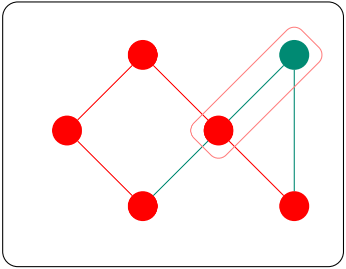
Graph Transformation

Graph transformation rule:



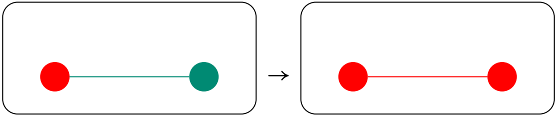
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



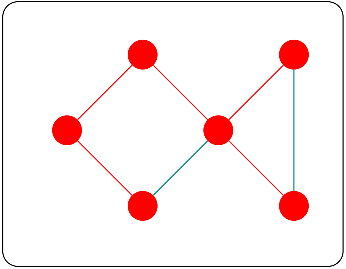
Graph Transformation

Graph transformation rule:



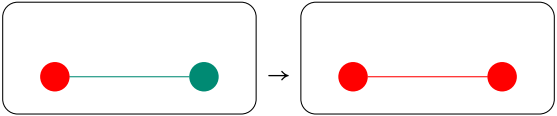
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



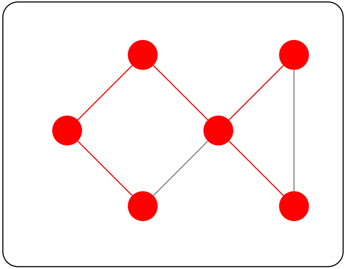
Graph Transformation

Graph transformation rule:



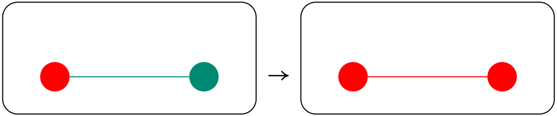
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



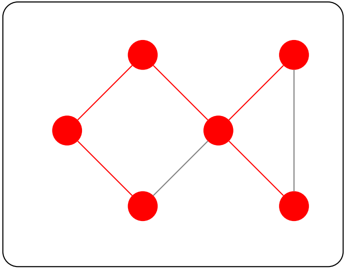
Graph Transformation

Graph transformation rule:



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

Spanning-tree construction:



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:

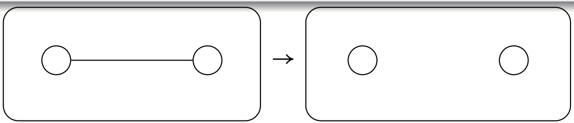
  "every execution (on any input) halts."

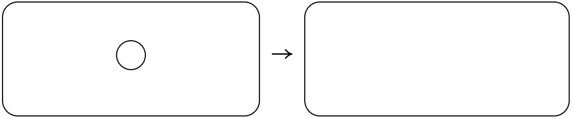
- Undecidable in general [Plu95]
  - Automated techniques for specific subclasses

Termination by interpretations

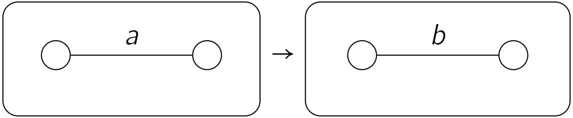
Interpret graphs as natural numbers.

Show each transformation step decreases the value.

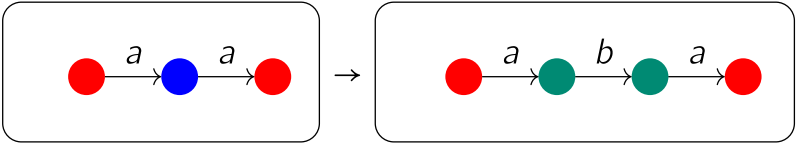
Number of edges: 

Number of nodes: 

Number of edges labeled by a:



Limitation



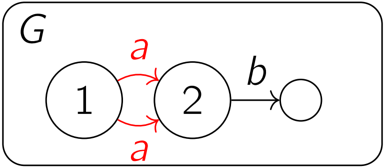
Left-hand side graph: middle has no other incident edges

Right-hand side graph: middles are fresh nodes

Can its termination be proved by interpretations?

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

Graphs: Finite, Directed, Edge-Labeled Multigraphs

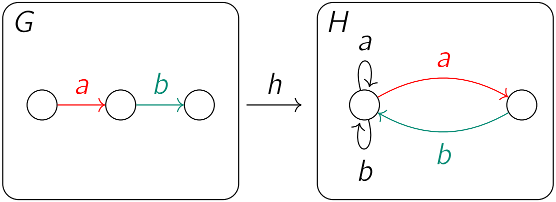


Edges with the same source, target and label are permitted.

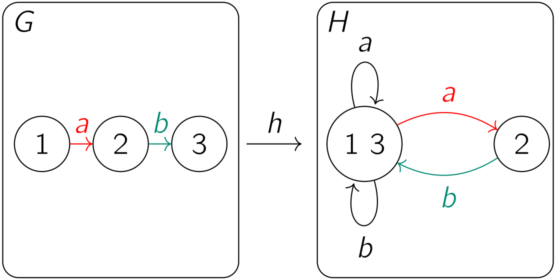
G: graph name

Numbers inside nodes: identifiers

Graph Morphisms: Structure-Preserving Functions



Colors show edge correspondence.



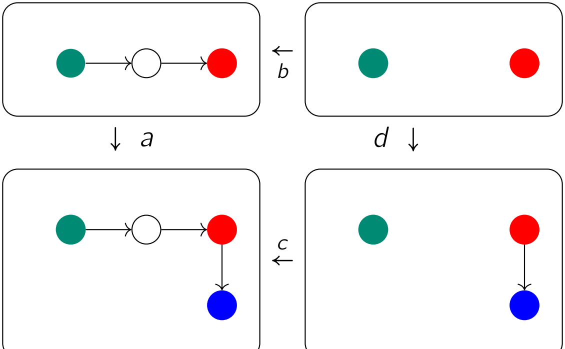
Numbers show node correspondence.

h : morphism name

Commutative Diagram

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

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



Pushouts: Gluing Graphs Along a common part

The **pushout** of (α, β) is

$$\begin{tikzcd} & A \arrow[dl, "\alpha"'] \arrow[dr, "\beta"] & \\ B & & C \end{tikzcd}$$

Pushouts: Gluing Graphs Along a common part

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

- $\square$ABDC commutes,

$$\begin{tikzcd}
 & A \arrow[dl, "\alpha"'] \arrow[dr, "\beta"] & \\
B \arrow[dr, "\beta'"'] \arrow[rr, phantom, "\mathrm{PO}"] & & C \arrow[dl, "\alpha'"] \\
 & D &
\end{tikzcd}$$

Pushouts: Gluing Graphs Along a common part

The **pushout** of (α, β) is $\color{red}{(\beta', \alpha')}$ 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.

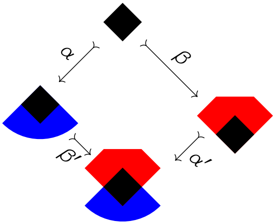
$$
\begin{tikzcd}
 & A \arrow[dl, "\alpha"'] \arrow[dr, "\beta"] \arrow[dd, phantom, "\color{red}{\mathrm{PO}}"] & \\
B \arrow[dr, red, "\beta'"'] \arrow[ddr, bend right, "\gamma'"'] & & C \arrow[dl, red, "\alpha'"] \arrow[ddl, bend left, "\gamma"] \\
 & \color{red}{D} \arrow[d, dashed, "\delta"] & \\
 & E &
\end{tikzcd}
$$

Pushouts: Gluing Graphs Along a common part

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.

$$\begin{tikzcd}
 & A \arrow[dl, "\alpha"'] \arrow[dr, "\beta"] & \\
B \arrow[dr, "\beta'"'] \arrow[ddr, bend right, "\gamma'"'] \arrow[rr, phantom, "\text{PO}"] & & C \arrow[dl, "\alpha'"] \arrow[ddl, bend left, "\gamma"] \\
 & D \arrow[d, dashed, "\delta"] & \\
 & E &
\end{tikzcd}$$

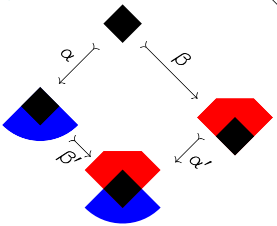


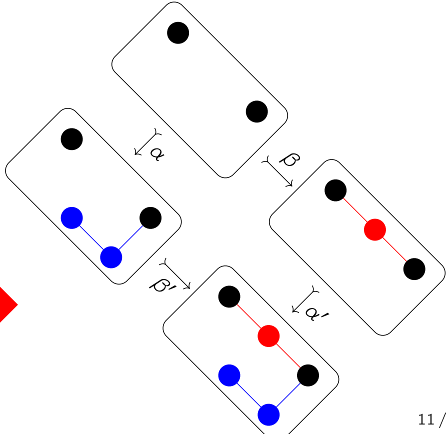
Pushouts: Gluing Graphs Along a common part

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.

$$
\begin{tikzcd}
 & A \arrow[dl, "\alpha"'] \arrow[dr, "\beta"] \arrow[dd, phantom, "\mathrm{PO}"] & \\
B \arrow[dr, "\beta'"'] \arrow[ddr, bend right, "\gamma'"'] & & C \arrow[dl, "\alpha'"] \arrow[ddl, bend left, "\gamma"] \\
 & D \arrow[d, dashed, "\delta"] & \\
 & E &
\end{tikzcd}
$$

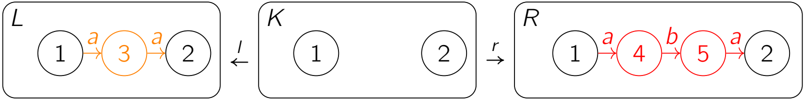




Graph Rewriting with Double-Pushout (DPO)

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

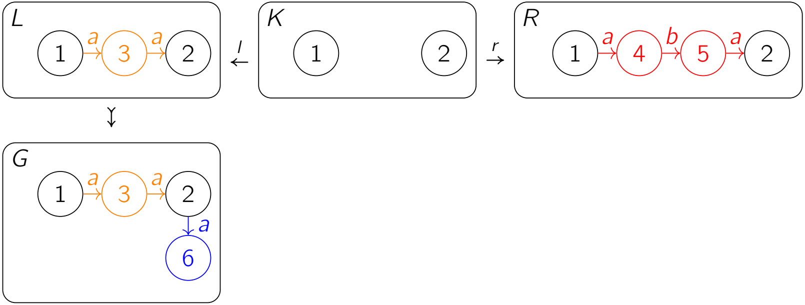
Rewriting rule with interface K



Graph Rewriting with Double-Pushout (DPO)

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

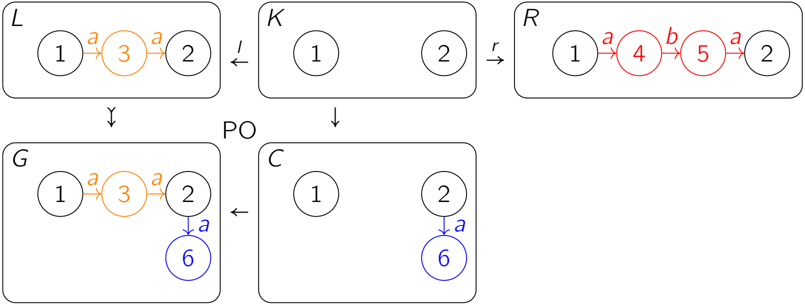
Rewriting rule with interface K



Graph Rewriting with Double-Pushout (DPO)

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

Rewriting rule with interface K

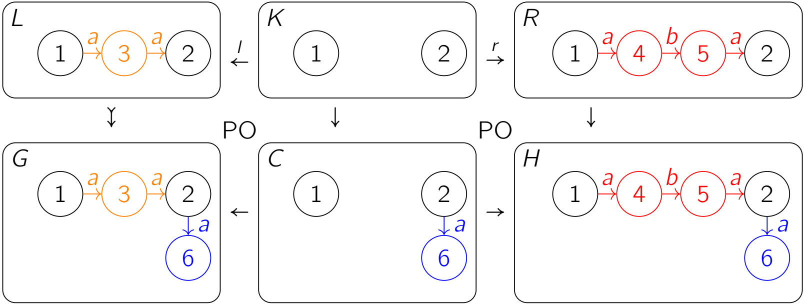


Graph Rewriting with Double-Pushout (DPO)

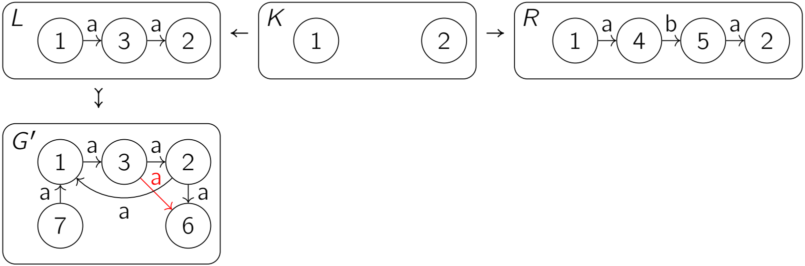
$$\begin{tikzcd}
L \arrow[d, tail] \arrow[dr, phantom, "\text{PO}"] & K \arrow[l, "l"'] \arrow[r, "r"] \arrow[d] \arrow[dr, phantom, "\text{PO}"] & R \arrow[d] \\
G & C \arrow[l] \arrow[r] & H
\end{tikzcd}$$

Rewriting rule with interface K

rewriting step $G \Rightarrow H$

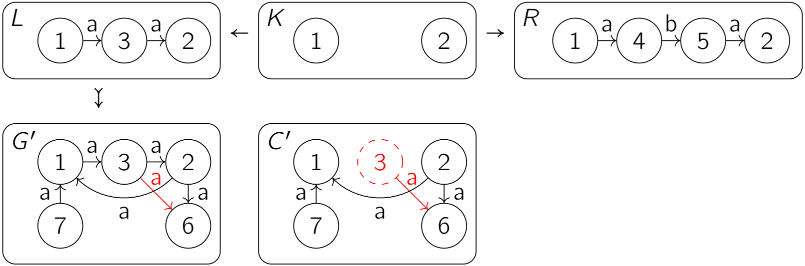


An Invalid Rewriting Step



No implicit edge deletion by construction

An Invalid Rewriting Step



No implicit edge deletion by construction

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:

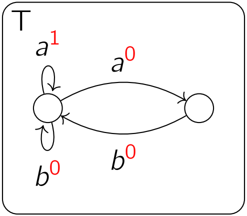
$$\begin{aligned} G &\rightsquigarrow \mathcal{F}(G, T) \\ &\rightsquigarrow \mathrm{weight}(\mathcal{F}(G, T)) \\ &\rightsquigarrow \mathrm{aggregator}(\mathrm{weight}(\mathcal{F}(G, T))) \in \mathbb{N} \end{aligned}$$

What is the morphism weight?

What is the graph weight?

Weighted Type Graph

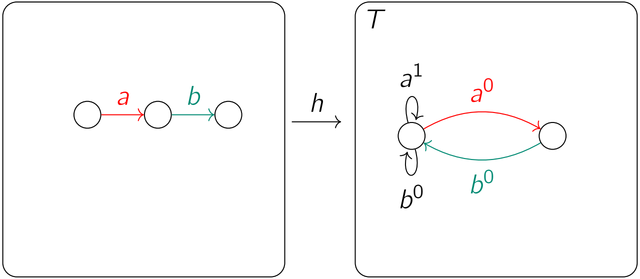
A weighted type graph is a graph with weights on edges.



Morphism Weight

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

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

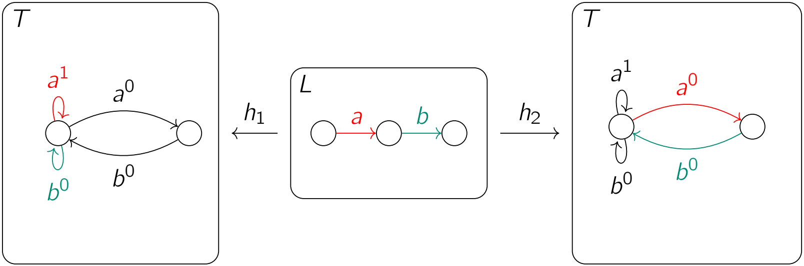


$\mathrm{weight}_T(h) = 0 + 0 = 0$

Graph Weight

The weight of a graph L is

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



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

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

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

Termination Criterion [Bru+15]

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

Every rewriting step strictly decreases the weight if

- for all t_K, if there is t_L such that $\triangle KLT$ commutes, then

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

How to find a suitable weighted type graph?

Searching for Weighted Type Graphs over $\mathbb{N}$

Restricted search space:

- no parallel edges of the same label

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:

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

Challenge:

- $2^{k^2 m} \cdot n^{k^2 m}$ 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 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 $\cancel{\mathbb{N}}$ $\mathbb{R}$

User-specified parameters:

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

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

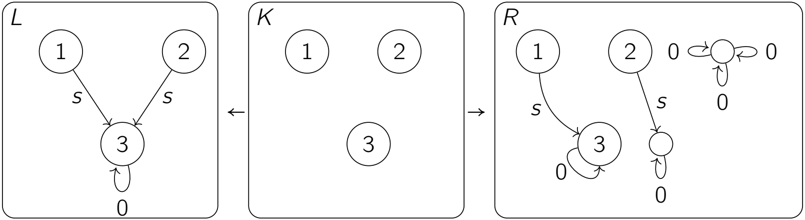
Tool in Ocaml

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.

Toward greater usability

Toward greater power

LyonParallel—A Tool for Termination of Graph Rewriting

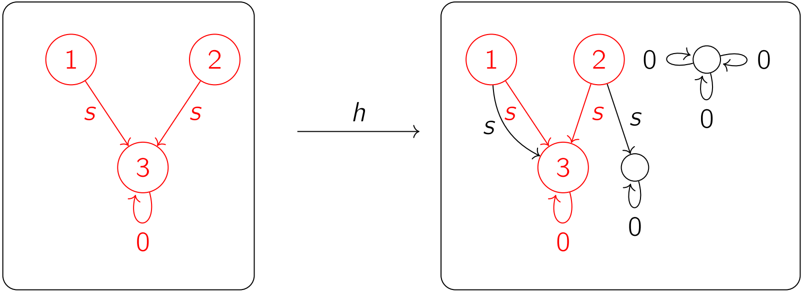
Morphism Counting

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.

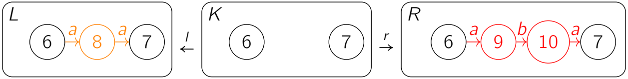


Subgraph

Graph Rewriting with Injective Rules

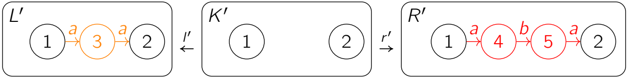
A rewriting rules consists of two inclusions.

$\varphi =$

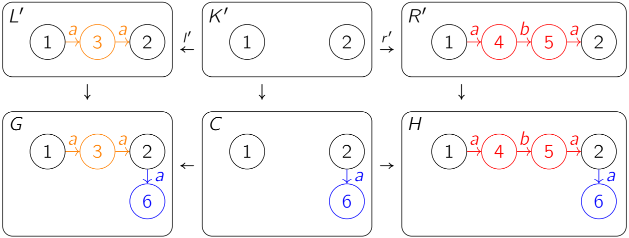


An equivalent rewriting rule expresses the same transformation.

$\varphi' =$

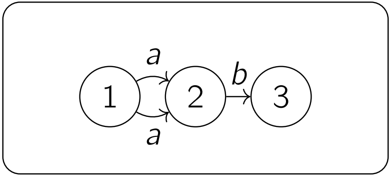


A rewriting step with φ is defined by a DPO diagram with inclusions and φ'.

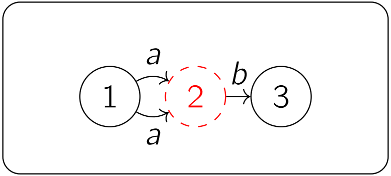


Pre-Graphs

Graph:



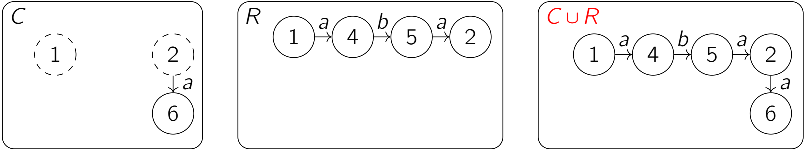
Pre-graphs obtained by removing node 2:



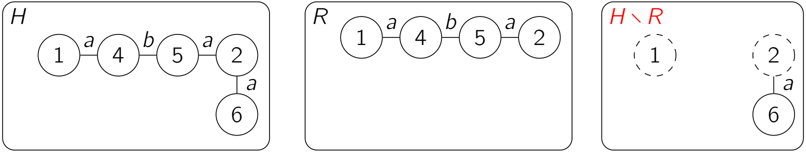
All edges are dangling.

Pre-Graph Operations

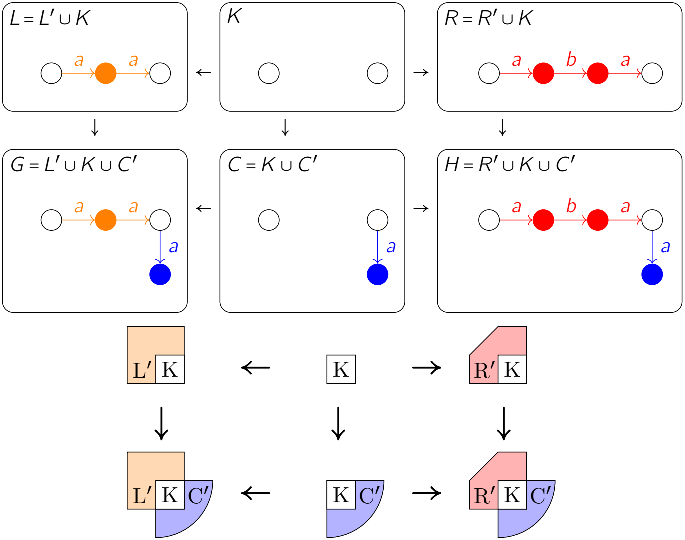
Union of two pre-graphs $C \subseteq G$ and $R \subseteq G$, denoted $C \cup R$.



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



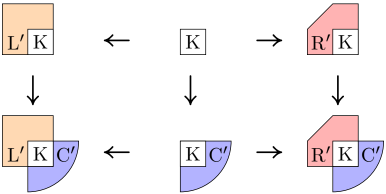
Decomposition of Graphs in Rewriting Steps



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.

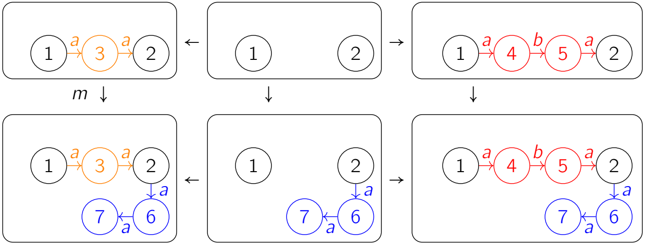


X-morphisms are classified by the colors of their image nodes:

- orange: only white and orange nodes;
- red: only white and red nodes;
- blue: only white and blue nodes;
- blue-and-orange: blue and orange nodes;
- blue-and-red: blue and red nodes.

Morphisms by Image Node Colors

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



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

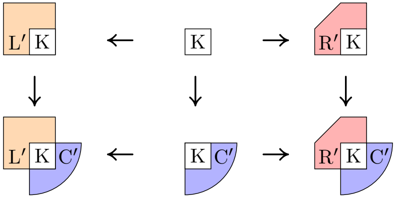
Orange X-morphisms: $1 \xrightarrow{a} 3 \xrightarrow{a} 2$

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

Red X-morphisms: none.

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

A Sufficient Condition for Termination

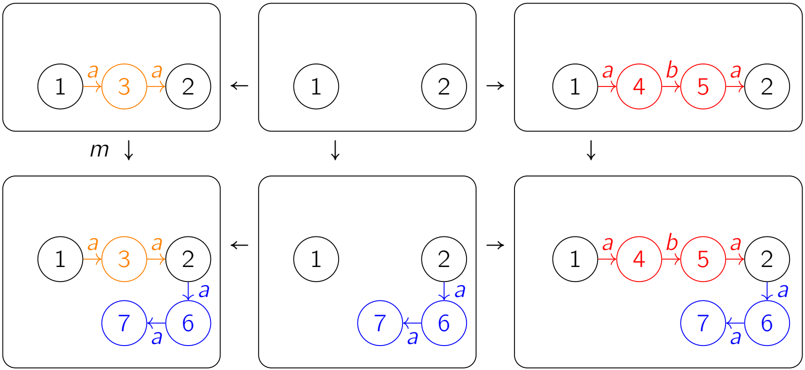


A rule terminates if, for every rewriting step, there are

- 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.

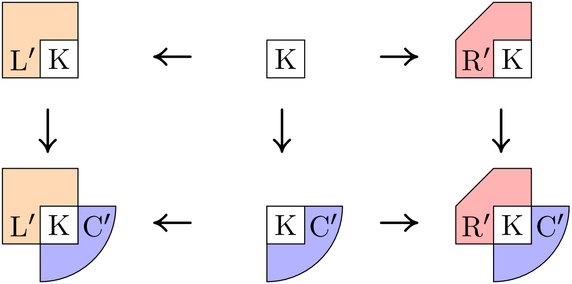
Analysis of Implicit Occurrences

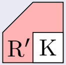


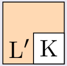
Blue-and-red X-morphisms: $5 \xrightarrow{a} 2 \xrightarrow{a} 6$

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

A Sufficient Condition for the Second Condition



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

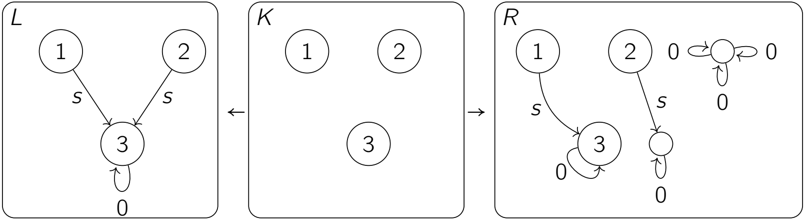
that can form an implicit X-occurrence in some rewriting step can be mapped to distinct subgraphs in 

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

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 $\bullet \xrightarrow{s} \bullet \xleftarrow{s} \bullet$.

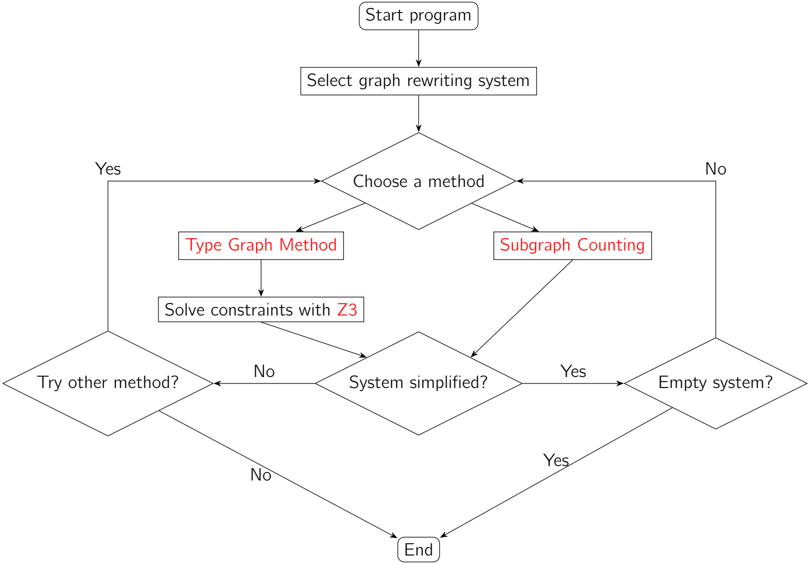
LyonParallel

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.

References III

[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.

[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.