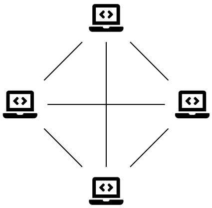
Automated Termination Proving: Contributions to Graph Rewriting via Extended Weighted Type Graphs and Morphism Counting

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.

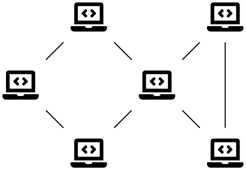
This thesis: automated verification.

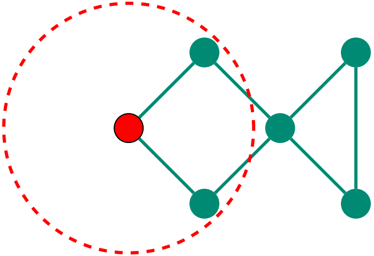
- Minimal user effort
- No expertise required
- Mathematically rigorous

Graph Transformation

Modelization of distributed systems

System configurations: graphs



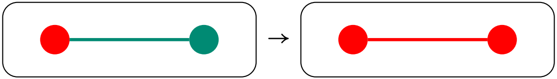


Algorithm behaviors:

graph transformation according to local knowledge

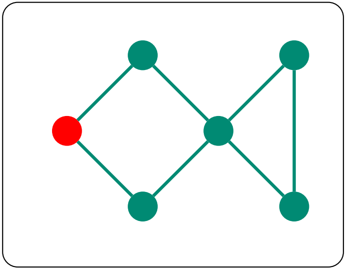
Graph Transformation

Graph transformation rule:



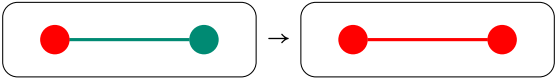
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



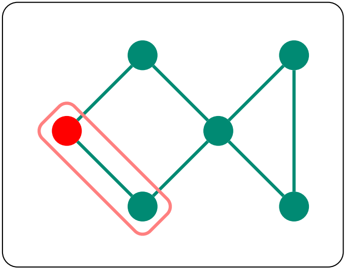
Graph Transformation

Graph transformation rule:



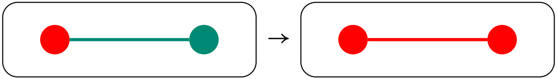
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



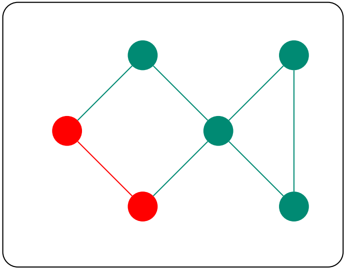
Graph Transformation

Graph transformation rule:



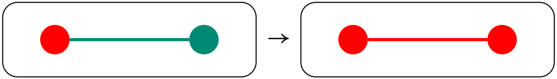
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



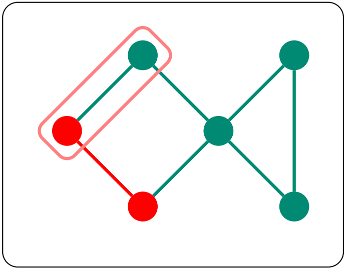
Graph Transformation

Graph transformation rule:



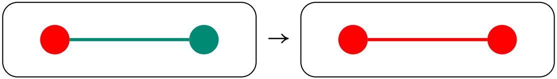
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



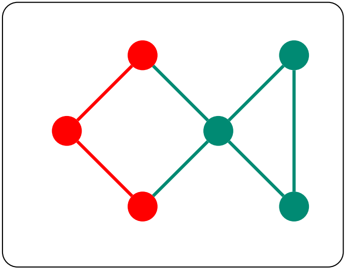
Graph Transformation

Graph transformation rule:



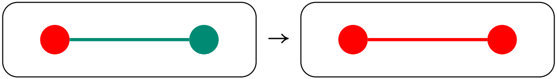
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



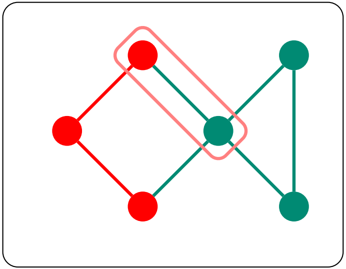
Graph Transformation

Graph transformation rule:



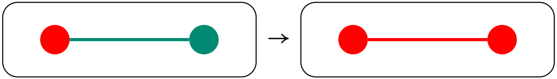
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



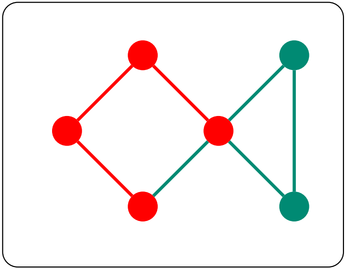
Graph Transformation

Graph transformation rule:



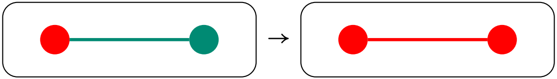
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



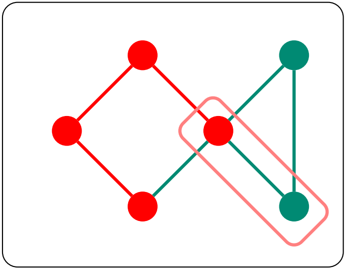
Graph Transformation

Graph transformation rule:



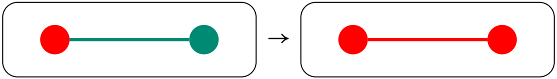
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



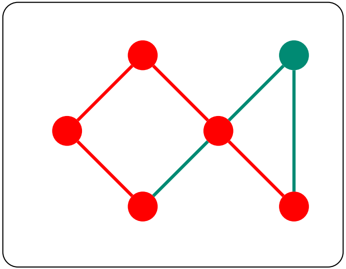
Graph Transformation

Graph transformation rule:



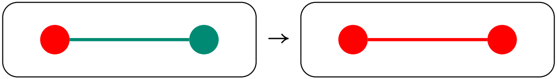
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



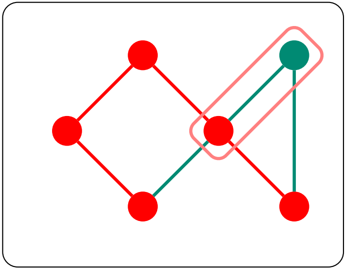
Graph Transformation

Graph transformation rule:



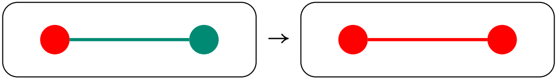
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



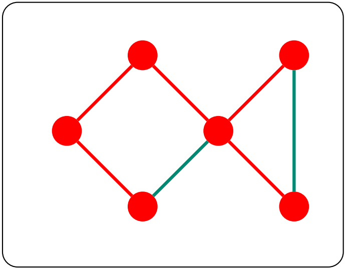
Graph Transformation

Graph transformation rule:



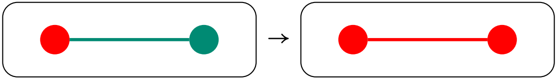
Replace the left-hand side by the right-hand side.

Spanning-tree construction:



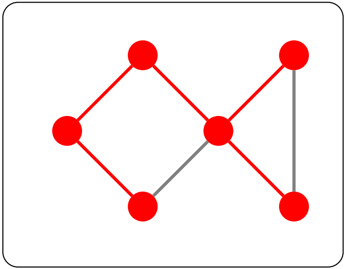
Graph Transformation

Graph transformation rule:



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

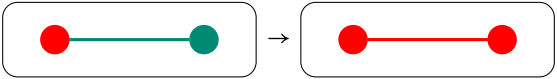
Spanning-tree construction:



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

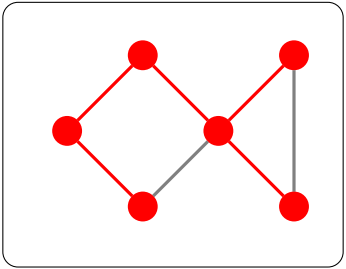
Graph Transformation

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."

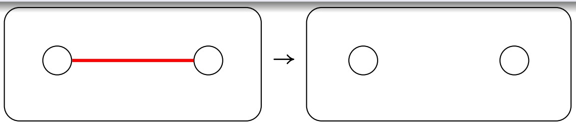
- Undecidable in general [11, Plump]
- Automated techniques
  - rely on user-privided parameters
  - incomplete

Termination by interpretations [13, 1, Zantema, Nipkow]

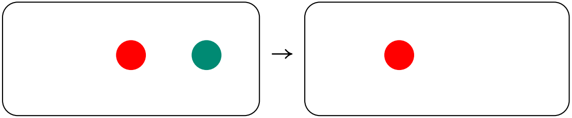
Interpret graphs as natural numbers.

Show each transformation step strictlydecreases the value.

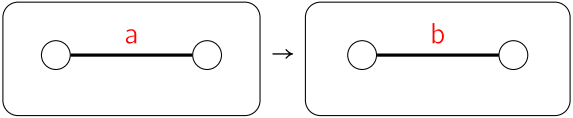
Number of edges:



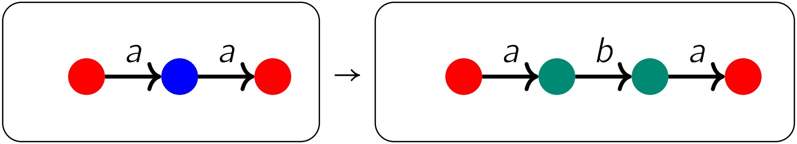
Number of nodes:



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

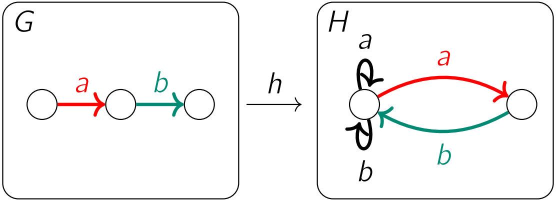
Formal Definition of Graph Rewriting with Double-Pushout

Toward Greater Usability

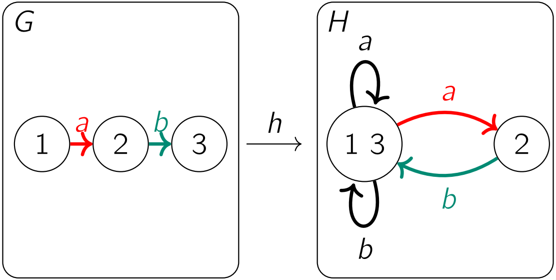
Toward Greater Power

LyonParallel—A Tool for Termination of Graph Rewriting

Graph Morphisms: Structure-Preserving Functions [2, Barr et al.]



Colors show edge correspondence.

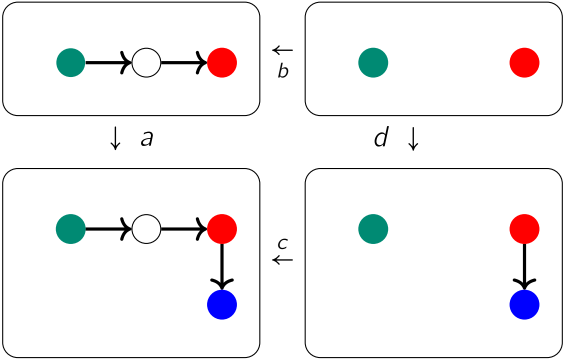


Numbers show node correspondence.

Commutative Diagram [2, Barr et al.]

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

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.

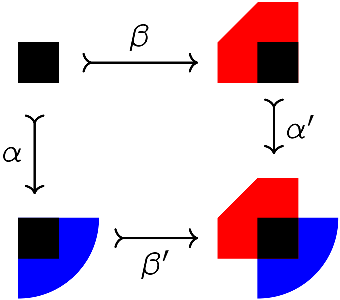
$$
\begin{tikzcd}
A \arrow[r, "\beta"] \arrow[d, "\alpha"'] & C \arrow[d, "\alpha'"'] \arrow[rdd, bend left, "\gamma'"] & \\
B \arrow[r, "\beta'"] \arrow[rrd, bend right, "\gamma"] & D \arrow[rd, dashed, "\delta"] & \\
 & & E
\end{tikzcd}
$$

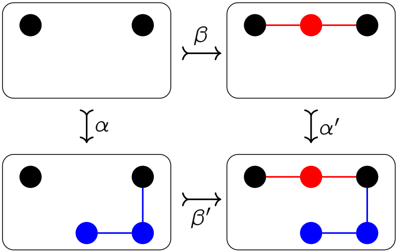
$\square$ABDC: pushout square

D: pushout object

Pushouts: Gluing Graphs Along an Interface [9, Pierce]

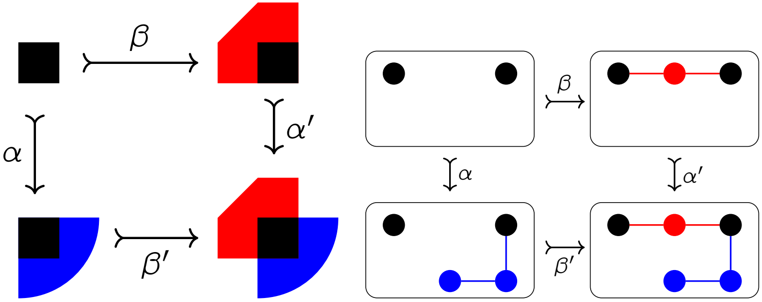
$$\begin{tikzcd} A \arrow[r, "\beta"] \arrow[d, "\alpha"'] & C \arrow[d, "\alpha'"] \\ B \arrow[r, "\beta'"] & D \end{tikzcd}$$





Pushouts: Gluing Graphs Along an Interface [9, Pierce]

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



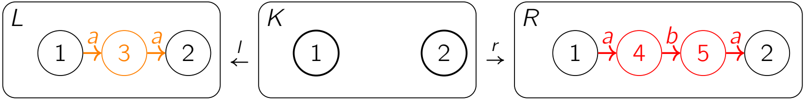
Graph Rewriting with Double-Pushout (DPO) [5, Ehrig et al.]

First algebraic approach

One of the most studied approaches

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

Rewriting rule with interface K



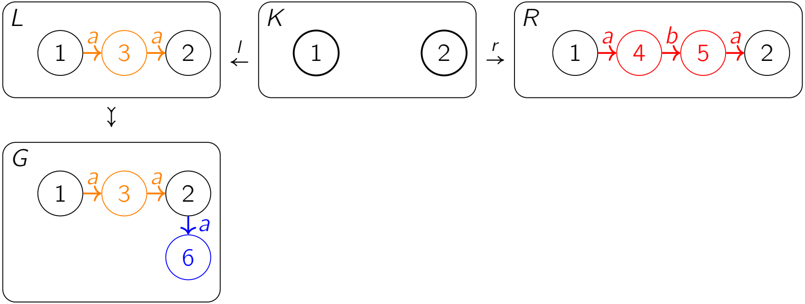
Graph Rewriting with Double-Pushout (DPO) [5, Ehrig et al.]

First algebraic approach

One of the most studied approaches

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

Rewriting rule with interface K



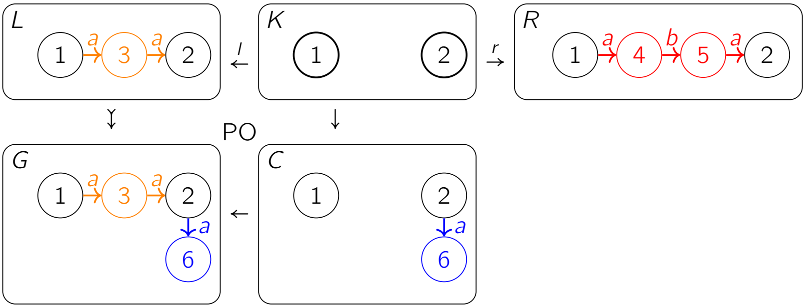
Graph Rewriting with Double-Pushout (DPO) [5, Ehrig et al.]

First algebraic approach

One of the most studied approaches

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

Rewriting rule with interface K



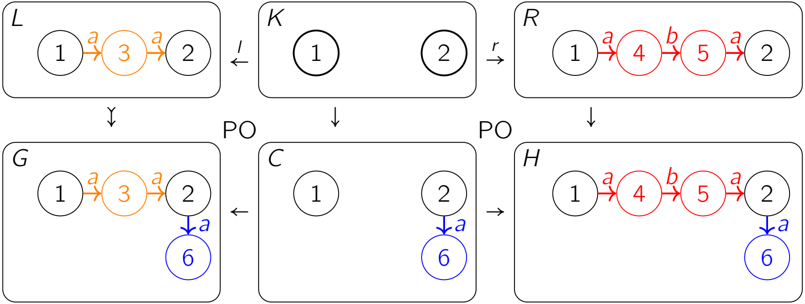
Graph Rewriting with Double-Pushout (DPO) [5, Ehrig et al.]

- First algebraic approach
- One of the most studied approaches

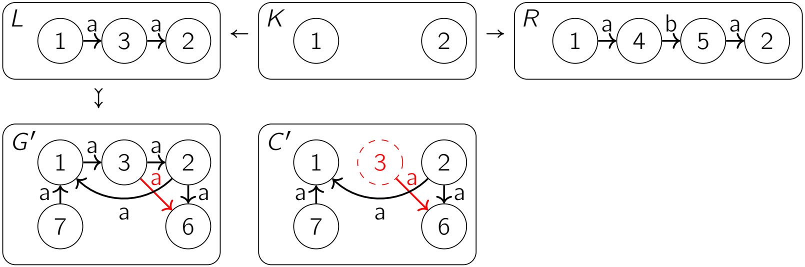
$$\begin{tikzcd} L \arrow[d, tail] & K \arrow[l, "l"'] \arrow[r, "r"] \arrow[d] & R \arrow[d] \\ G \arrow[ur, phantom, "\text{PO}"] & C \arrow[l] \arrow[r] \arrow[ur, phantom, "\text{PO}"] & H \end{tikzcd}$$

Rewriting rule with interface K

rewriting step $G \Rightarrow H$



An Invalid Rewriting Step



Weighted Type Graph Method [7, Endrullis et al.]

Termination by interpretation

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

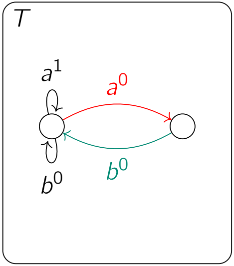
Terminology: every graph is "typed" as morphisms to T

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?

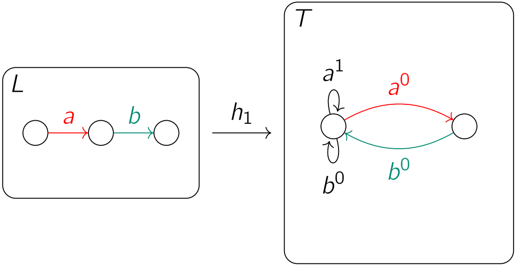
Type Graph with Weights on Edges



Morphism Weight

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

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

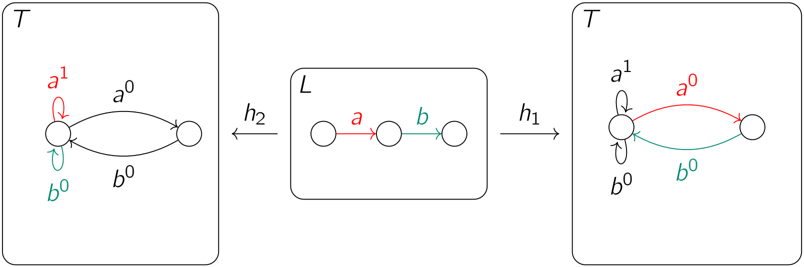


$$\mathrm{weight}_T(h_1) = 0 + 0 = 0$$

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 [4, Bruggink et al.]

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

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

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

How to find a suitable weighted type graph?

Searching for Weighted Type Graphs over $\mathbb{N}$ [4, Bruggink et al.]

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:

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

Usability Improvement

Using positive real numbers as weights [8, Lucas]

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

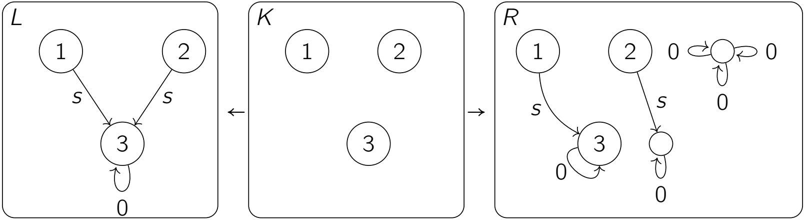
With weights in $\mathbb{R}$:

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

Implementation⇝Experiments

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

A Limitation of the Weighted Type Graph Method



Type graph fails: existence of surjections from R to L [Endrullis and Overbeek].

All existing automated methods fail.

Remark: the number of occurrences of L strictly decreases.

Capability Improvement: Morphism Counting

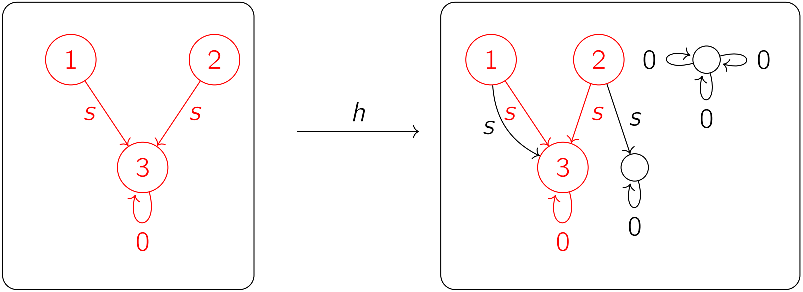
Termination by interpretation

Parameter: graph X

Interpretation:

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

Inclusions

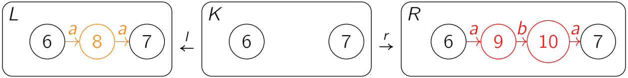


Subgraph

Graph Rewriting Systems

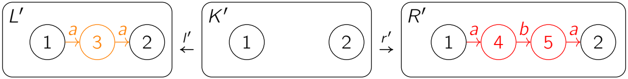
A rewriting rules consists of two inclusions.

$\varphi=$

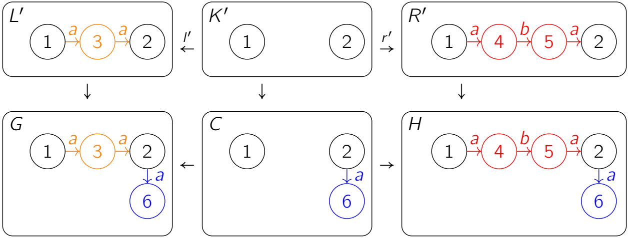


An equivalent rewriting rule expresses the same transformation.

$\varphi'=$

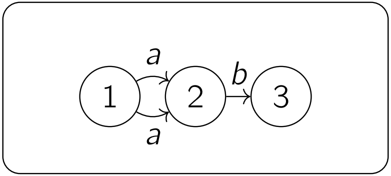


A rewriting step with φ is defined by a DPO diagram with inclusions and φ'.

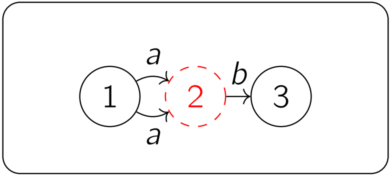


Pre-Graphs

Graph:

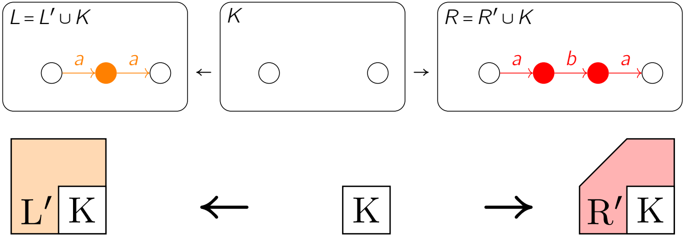


Pre-graphs obtained by removing node 2:

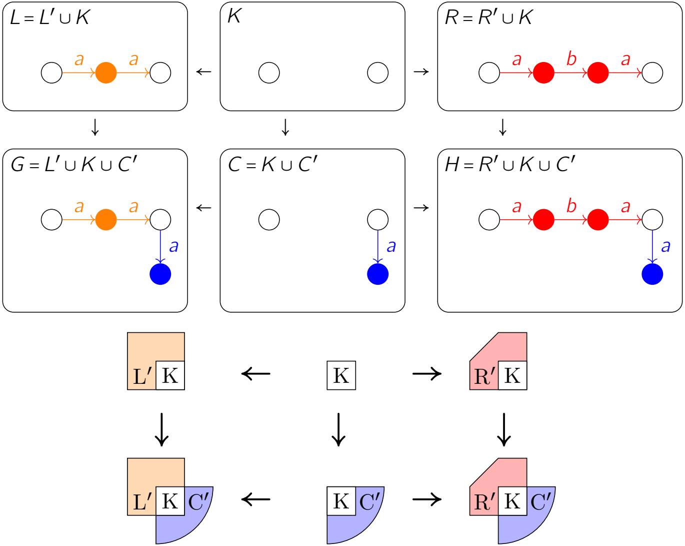


Dangling edges

Decomposition of Graphs in Rewriting Rules



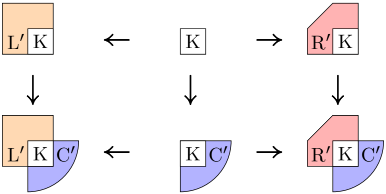
Decomposition of Graphs in Rewriting Steps



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

X-occurrences by Image Node Colors

An X-occurrence is an injective morphism from X.

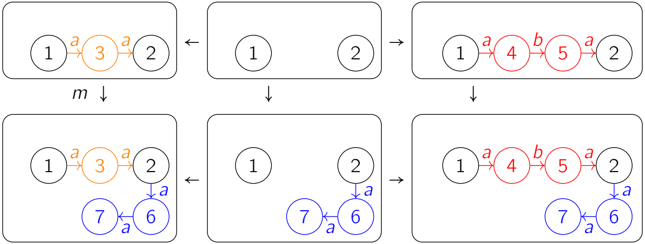


X-occurrence are classified by the colors of their image nodes:

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

Morphisms by Image Node Colors

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

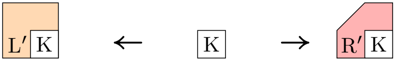


Blue X-occurrence: $2 \xrightarrow{a} 6 \xrightarrow{a} 7$

Red X-occurrences: none.

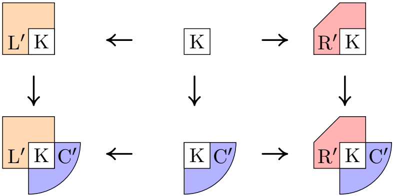
Blue-and-red X-occurrences: $5 \xrightarrow{a} 2 \xrightarrow{a} 6$

A New Sufficient Condition for Termination [12, Qiu]



terminates if there are strictly more orange X-occurrences than red X-occurrences,

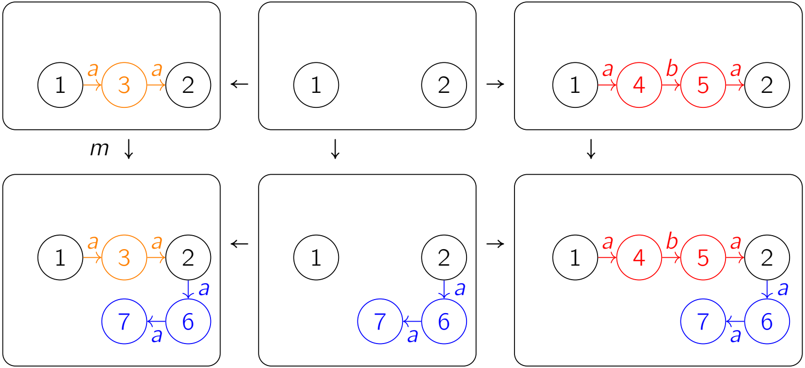
- every rewriting step:



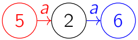
has more blue-and-orange X-occurrences than blue-and-red X-occurrences.

Challenge: check the second condition under the unknown C’

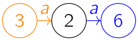
Analysis of Implicit Occurrences



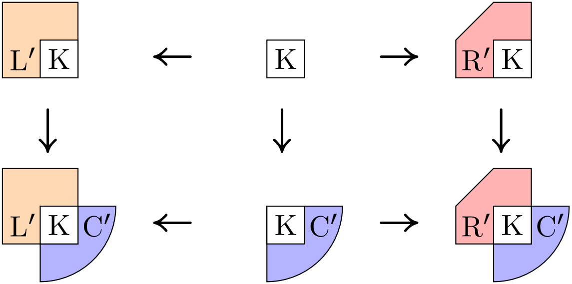
Blue-and-red X-occurrences:

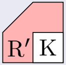
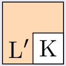


Blue-and-Orange X-occurrences:

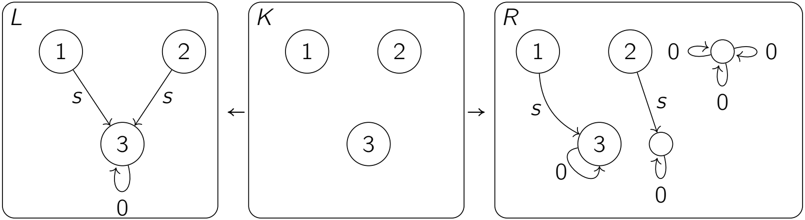


Sufficient Condition for the Second Condition [12, Qiu]



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 $\boxed{\mathrm{K}}$.

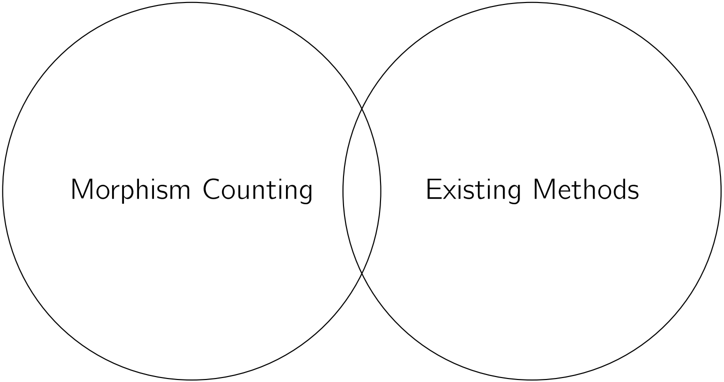
Termination of Motivating Example



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.
More power if search in parallel ✓

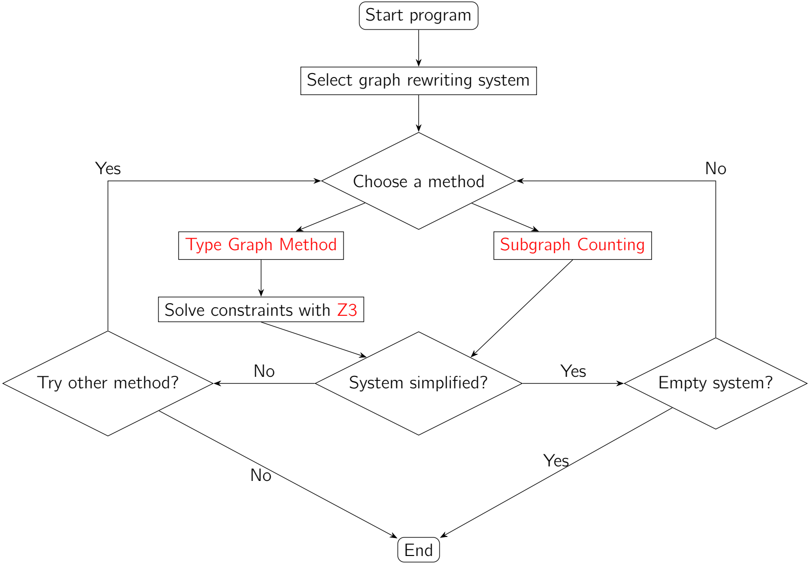
LyonParallel

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] Michael Barr and Charles Wells. *Category theory for computing science (2. ed.)* Prentice Hall international series in computer science. Prentice Hall, 1995. ISBN: 978-0-13-323809-9.

[3] 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.

[4] 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

[5] 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.

[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.

[9] Benjamin C. Pierce. *Basic category theory for computer scientists*. Foundations of computing. MIT Press, 1991. ISBN: 978-0-262-66071-6.

References IV

[10] 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.

[11] 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.

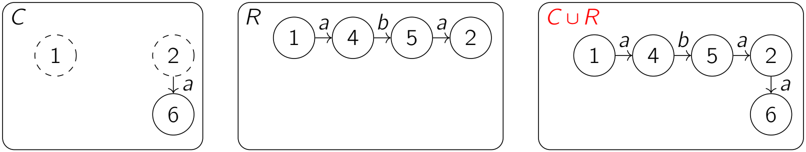
References V

[12] 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.

[13] 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 $H \setminus R$.