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

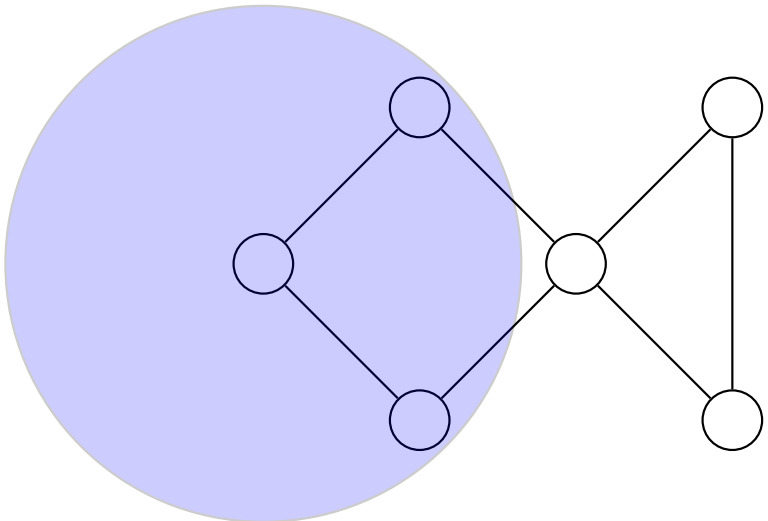
Distributed systems are everywhere.

Failures can be catastrophic.

Ensuring correctness is hard.

This thesis: automated verification.

Graph Transformation

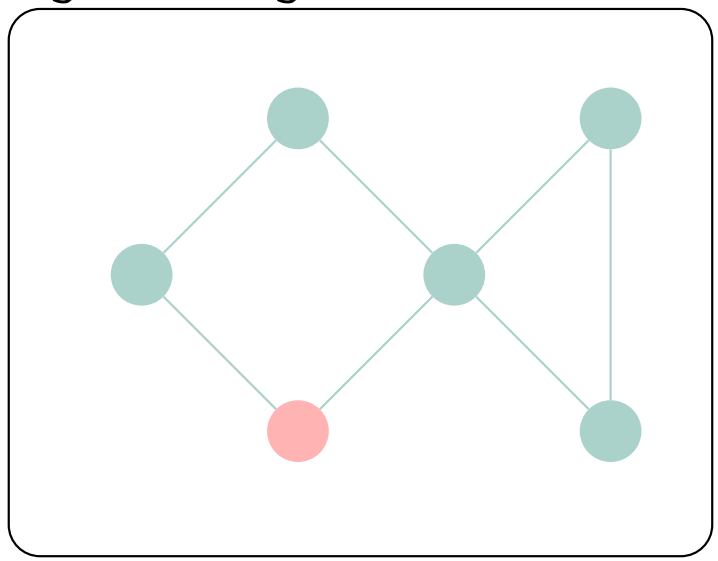


Graph rewriting:

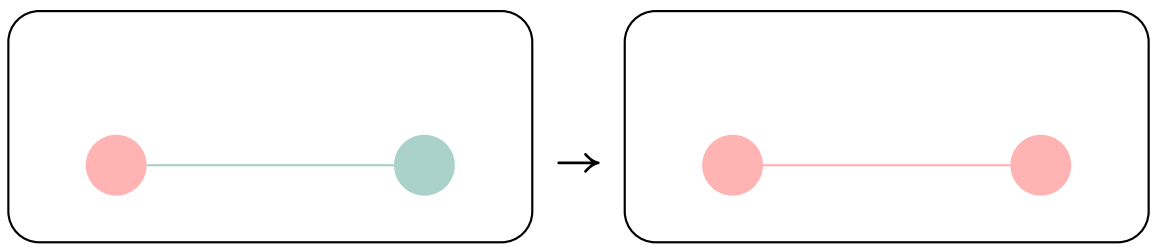
- computational units → nodes
- communication channels → edges
- system states → graphs
- algorithm behaviors
  → graph transformation rules according to local knowledge

Graph Transformation

Graph representing a configuration of a distributed system:

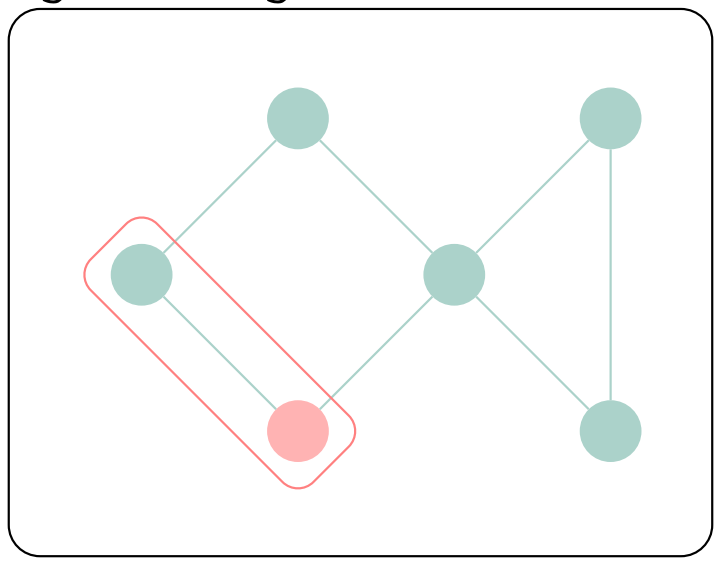


Graph transformation rule:

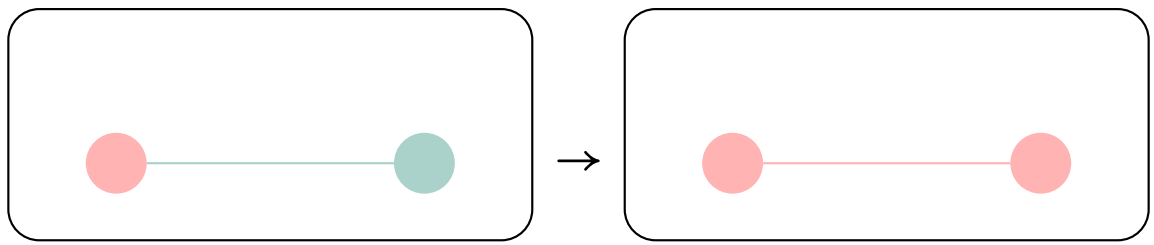


Graph Transformation

Graph representing a configuration of a distributed system:

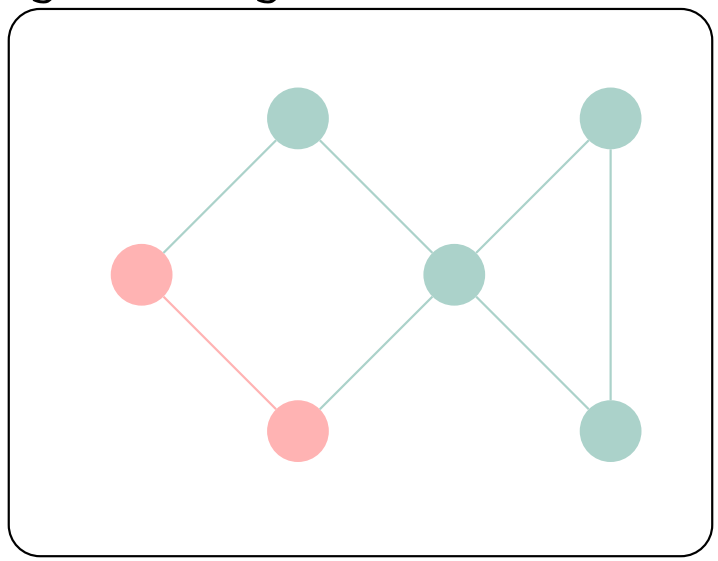


Graph transformation rule:

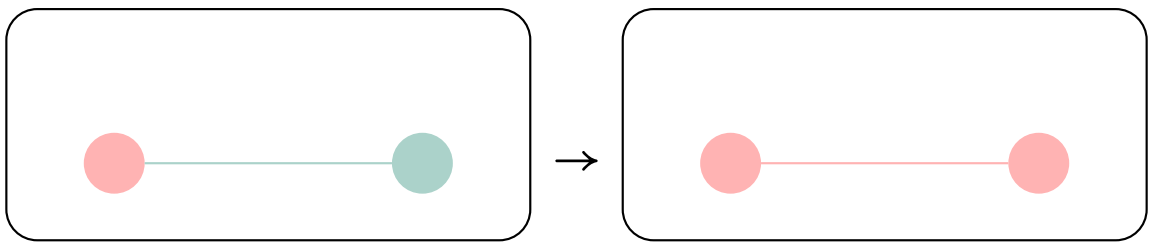


Graph Transformation

Graph representing a configuration of a distributed system:

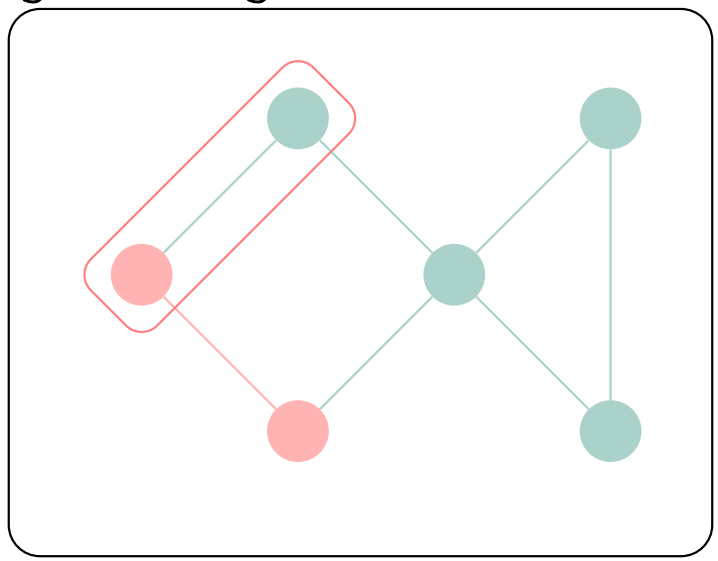


Graph transformation rule:

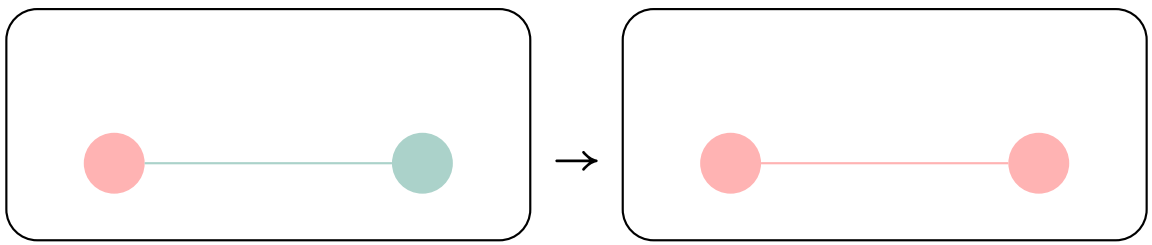


Graph Transformation

Graph representing a configuration of a distributed system:

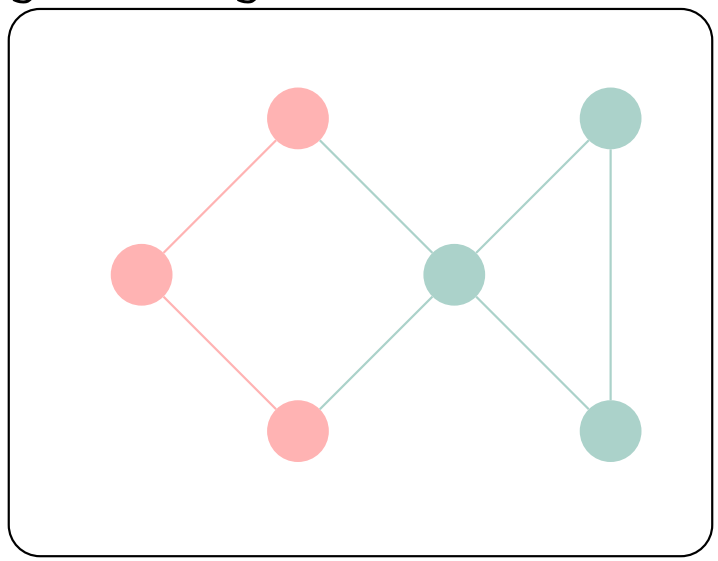


Graph transformation rule:

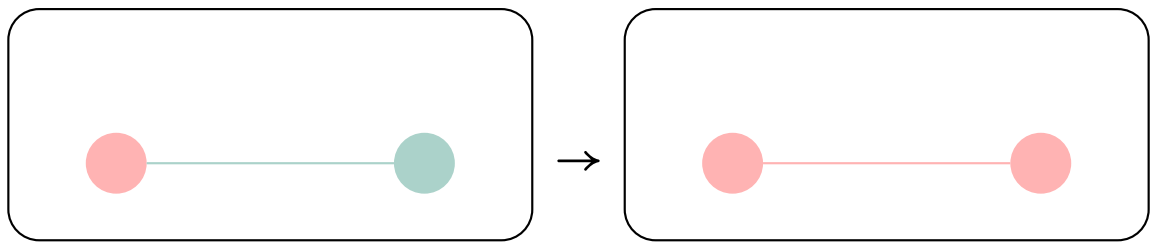


Graph Transformation

Graph representing a configuration of a distributed system:

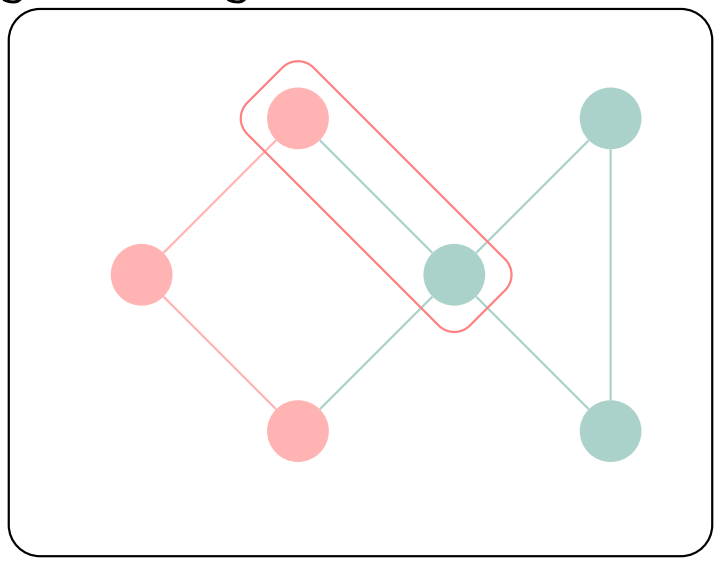


Graph transformation rule:

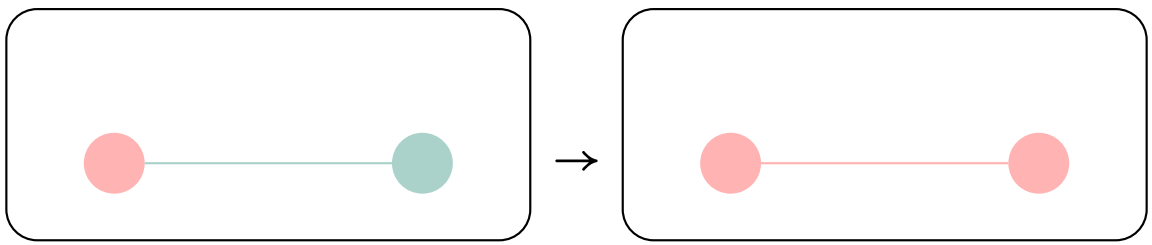


Graph Transformation

Graph representing a configuration of a distributed system:

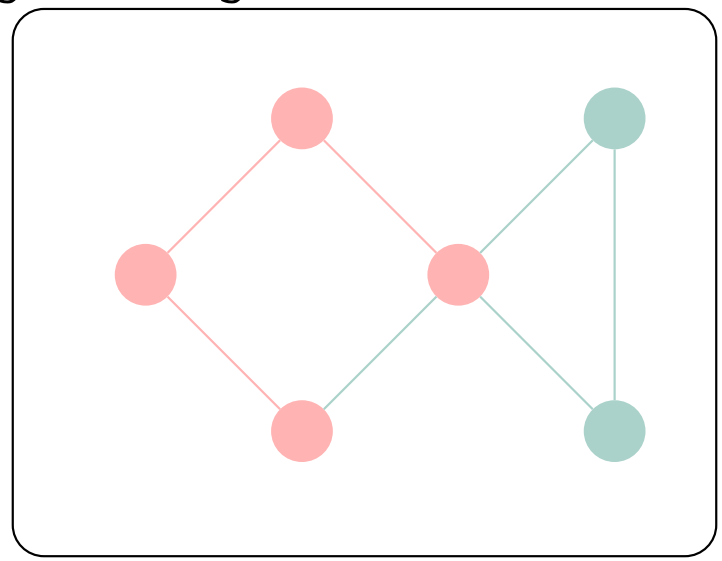


Graph transformation rule:

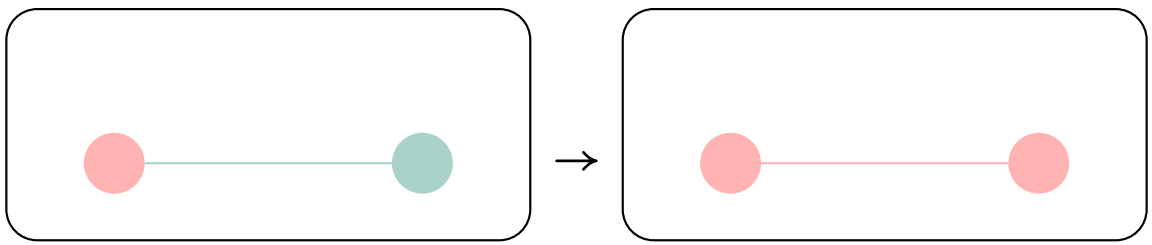


Graph Transformation

Graph representing a configuration of a distributed system:

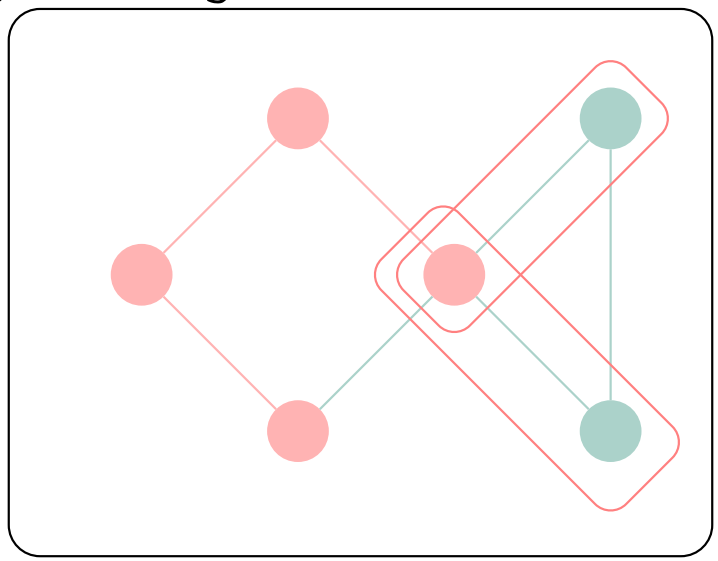


Graph transformation rule:

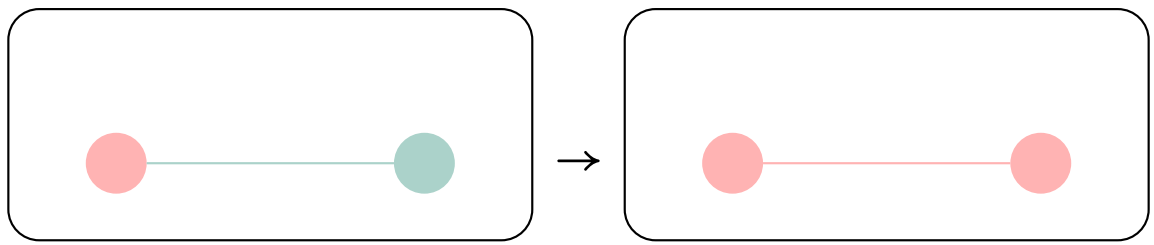


Graph Transformation

Graph representing a configuration of a distributed system:

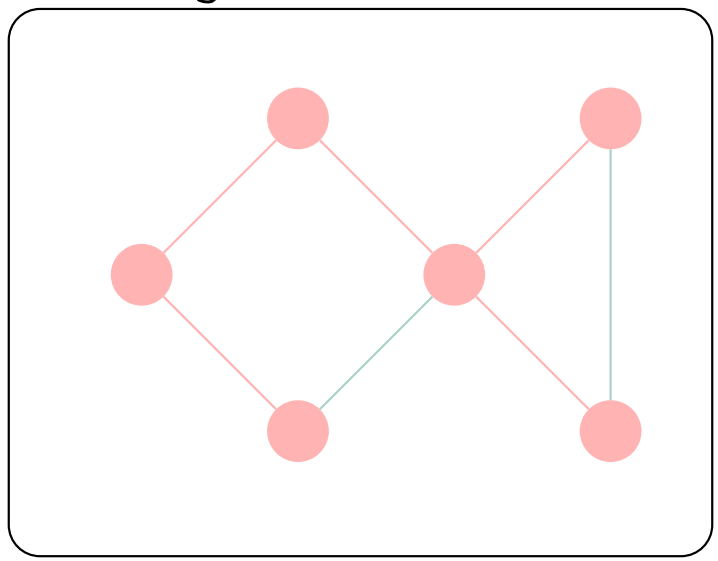


Graph transformation rule:

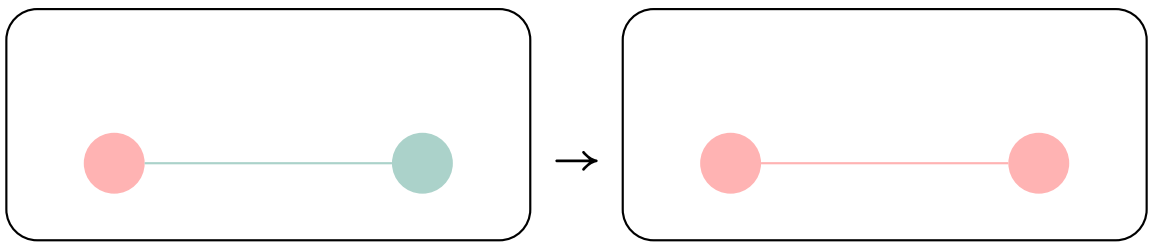


Graph Transformation

Graph representing a configuration of a distributed system:

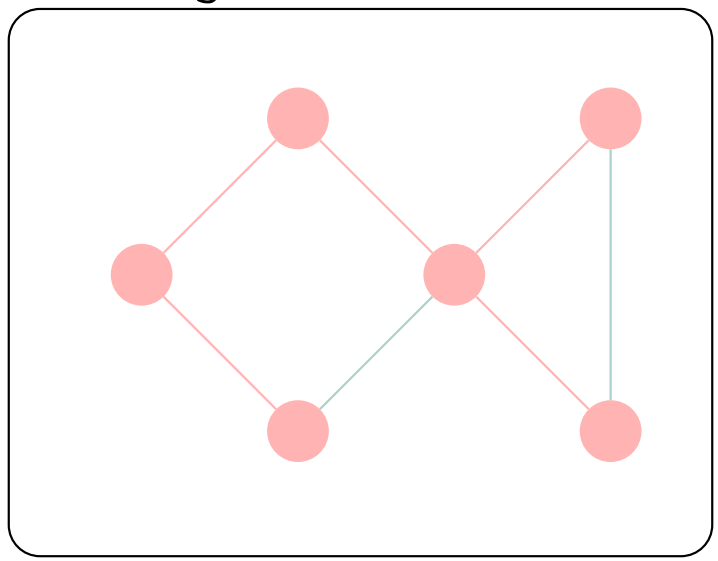


Graph transformation rule:

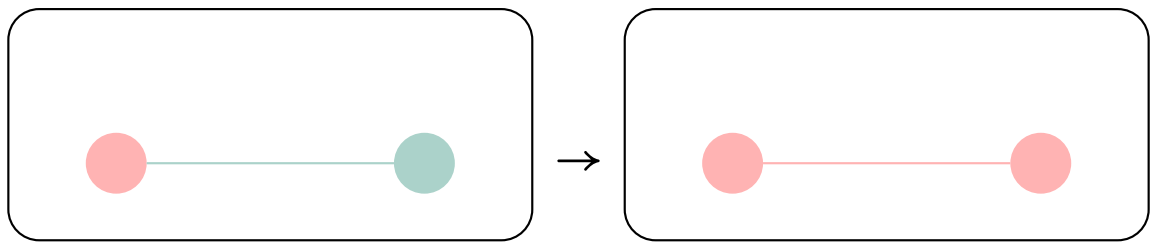


Graph Transformation

Graph representing a configuration of a distributed system:

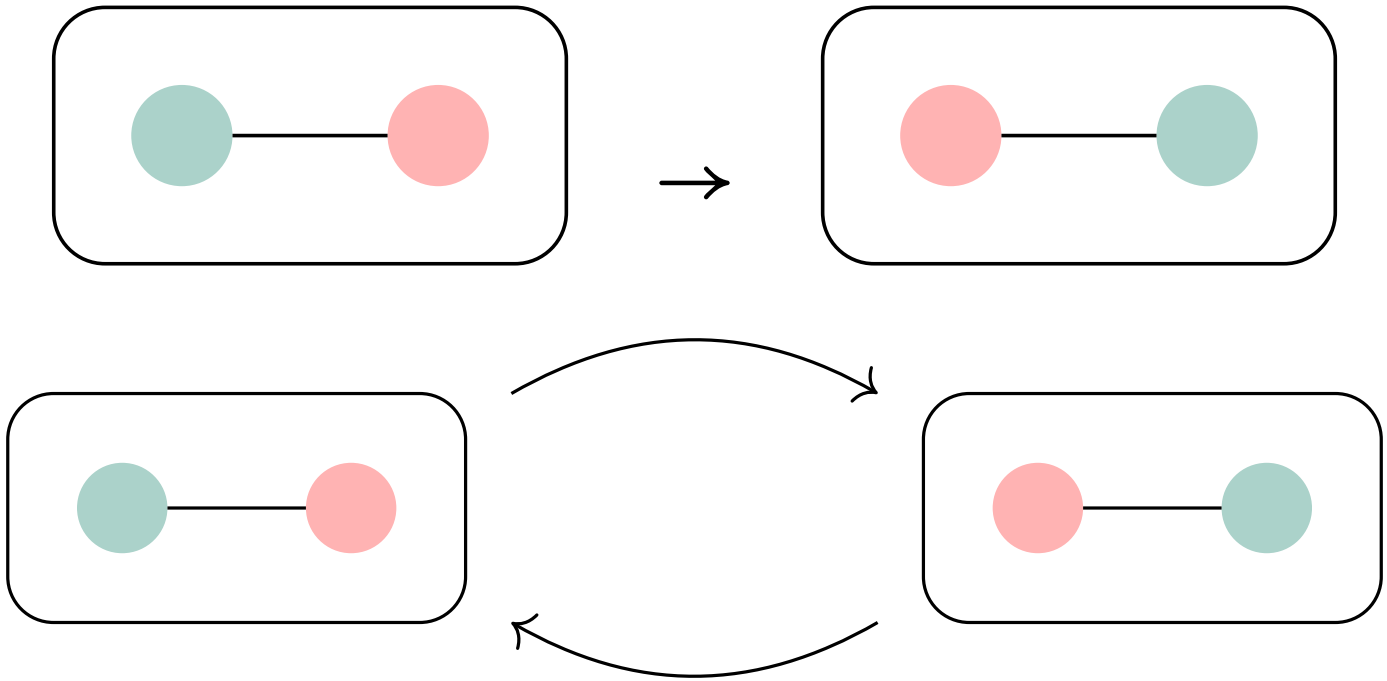


Graph transformation rule:



Does the transformation process terminate for any initial graph under the strategy of applying the rule while possible?

A Non-Termination Case



Termination

- No graph G_0 can be transformed forever

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

  under the strategy

  "apply rules as long as possible"

- Aligns with the notion of program termination:

  "every execution (on any input) halts."

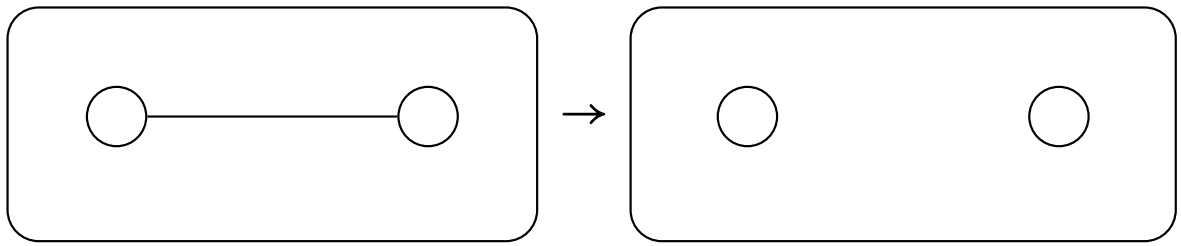
- Undecidable in general
- How to prove termination automatically?

Automated Termination Proofs

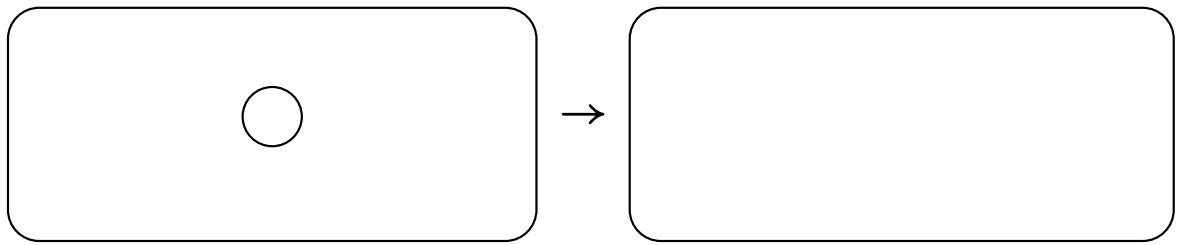
Termination by interpretations:

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

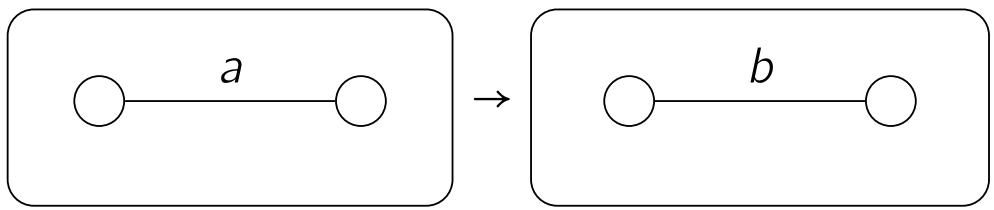
Number of edges:



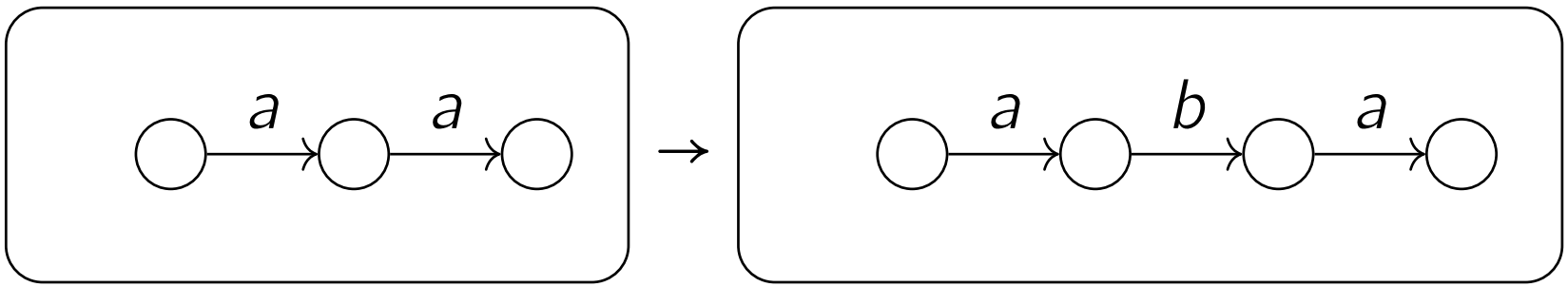
Number of nodes:



Number of edges labeled by a:



A Non-Trivial Example

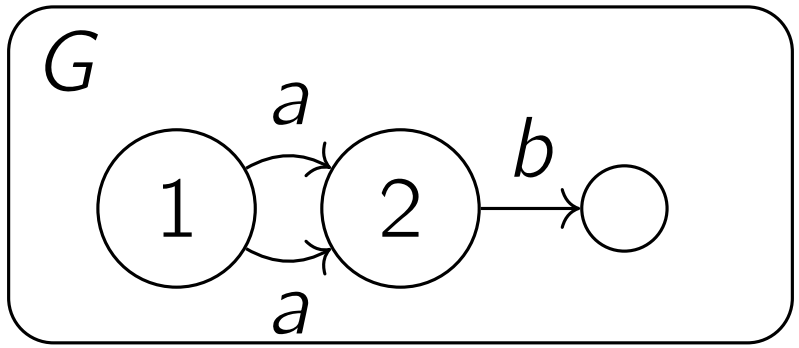


Limitation of simple interpretations

Need a formal definition of graph transformations

- edges incident to deleted nodes

Graphs: Finite, Directed, Edge-Labeled Multigraphs



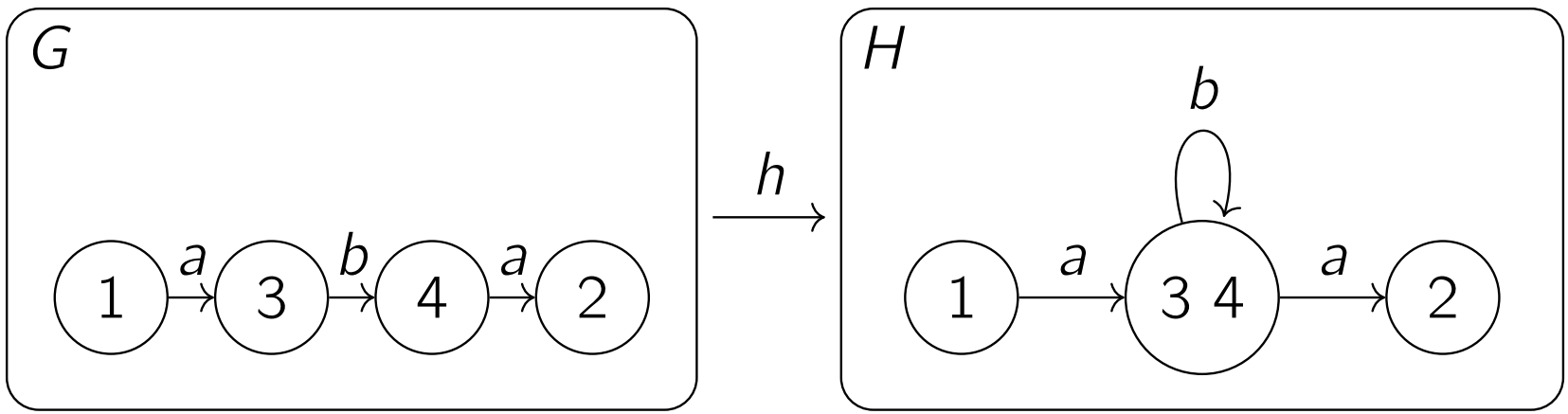
Edges with the same source, target and label are permitted.

G : graph name

Numbers inside nodes: identifiers

A non-directed edge is two directed edges with opposite directions.

Graph Morphisms: Structure-Preserving Functions



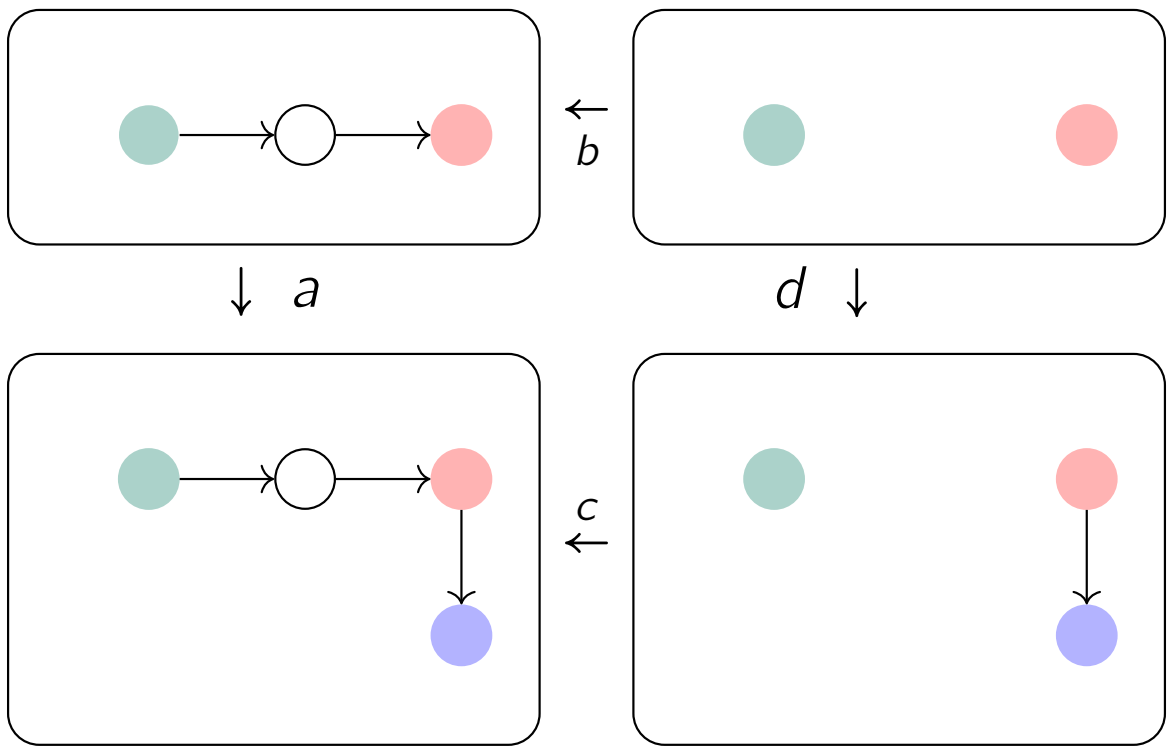
Nodes of H are labeled with the sets of identifiers of nodes of G that map to them.

$\overset{h}{\to}$ indicates $h : G \to H$.

Commutative Diagram

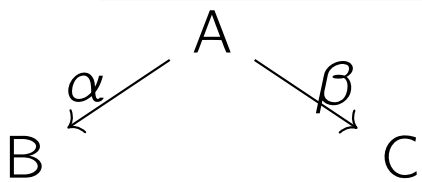
$$\begin{array}{ccc} L & \xleftarrow{b} & K \\ \downarrow{\scriptstyle a} & & \downarrow{\scriptstyle d} \\ G & \xleftarrow{c} & C \end{array}$$

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



Pushouts: Gluing Graphs Along a common part

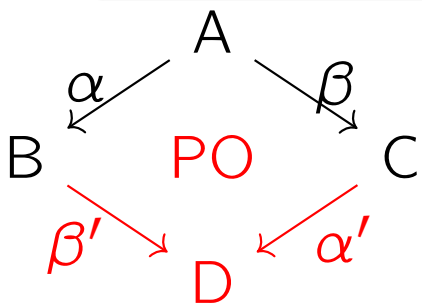
The **pushout** of (α, β) is



Pushouts: Gluing Graphs Along a common part

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

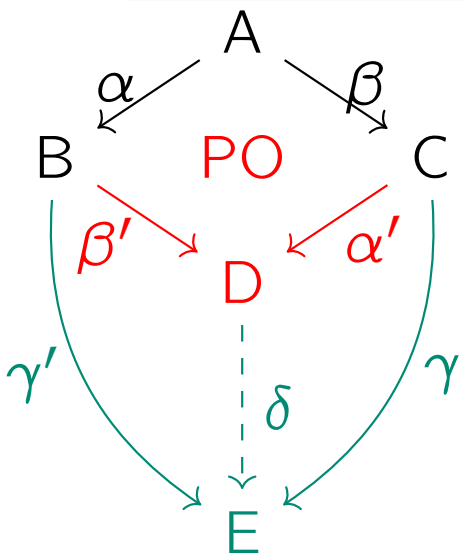
- $\square$ABDC commutative,



Pushouts: Gluing Graphs Along a common part

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

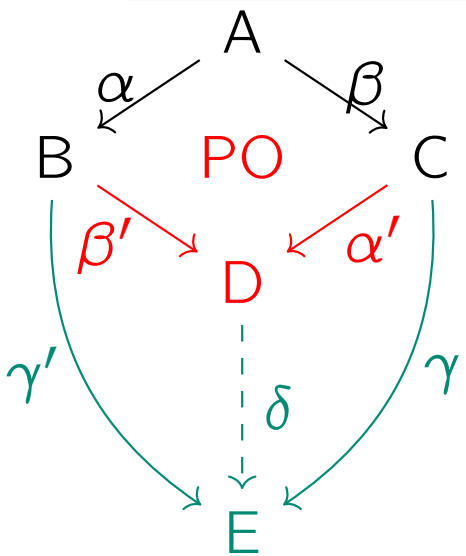
- $\square$ABDC commutative,
- universality: for all (γ, γ'),if $\square ABEC$ is commutative, then there is a unique δ such that $\triangle BDE$ and $\triangle CDE$ are commutative.

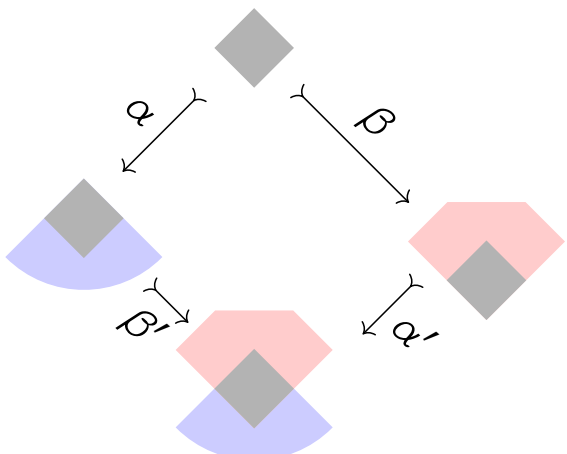


Pushouts: Gluing Graphs Along a common part

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

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

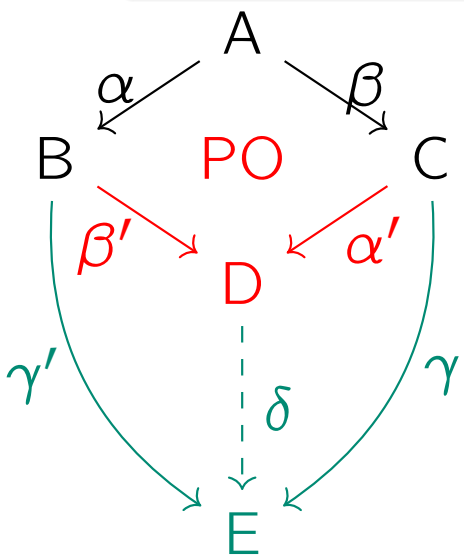


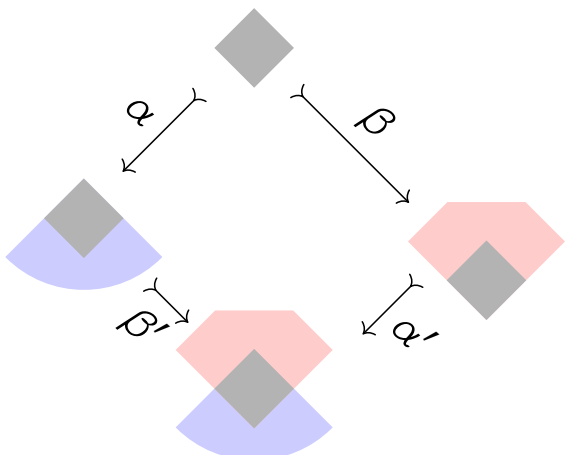


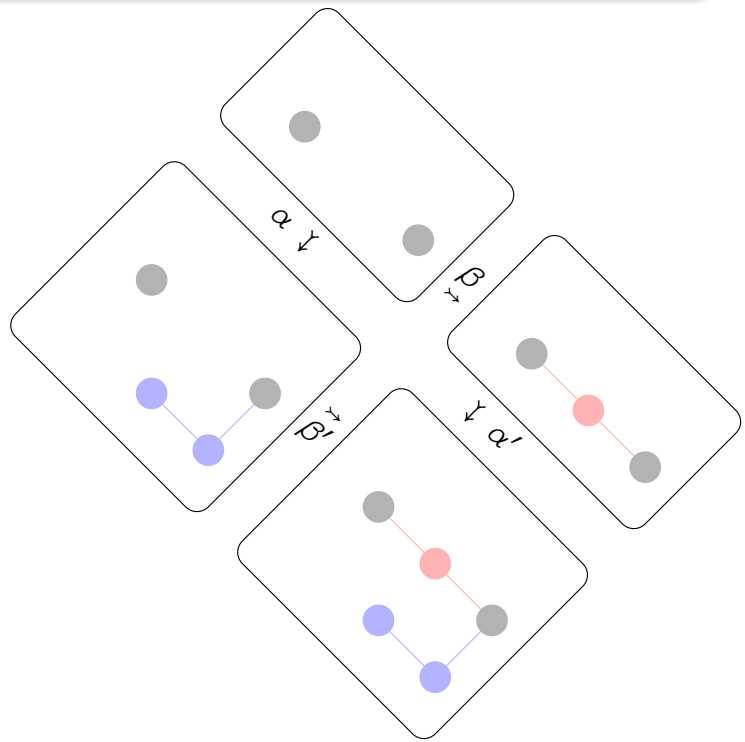
Pushouts: Gluing Graphs Along a common part

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

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



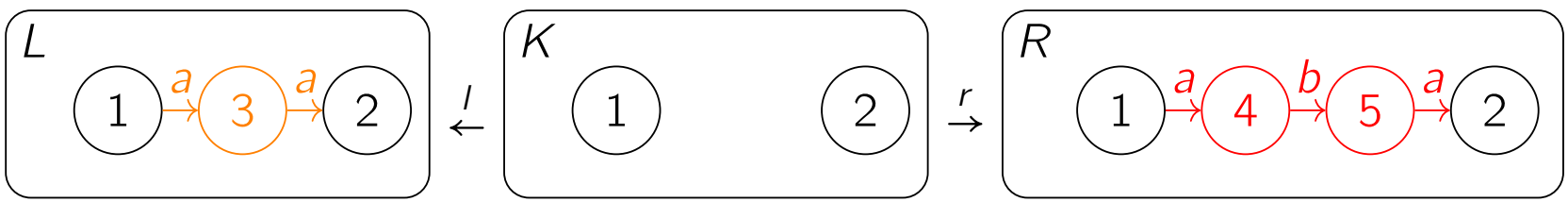




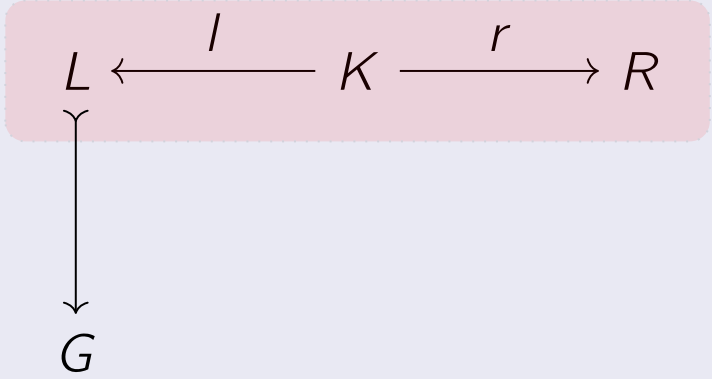
Graph Rewriting with Double-Pushout (DPO)

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

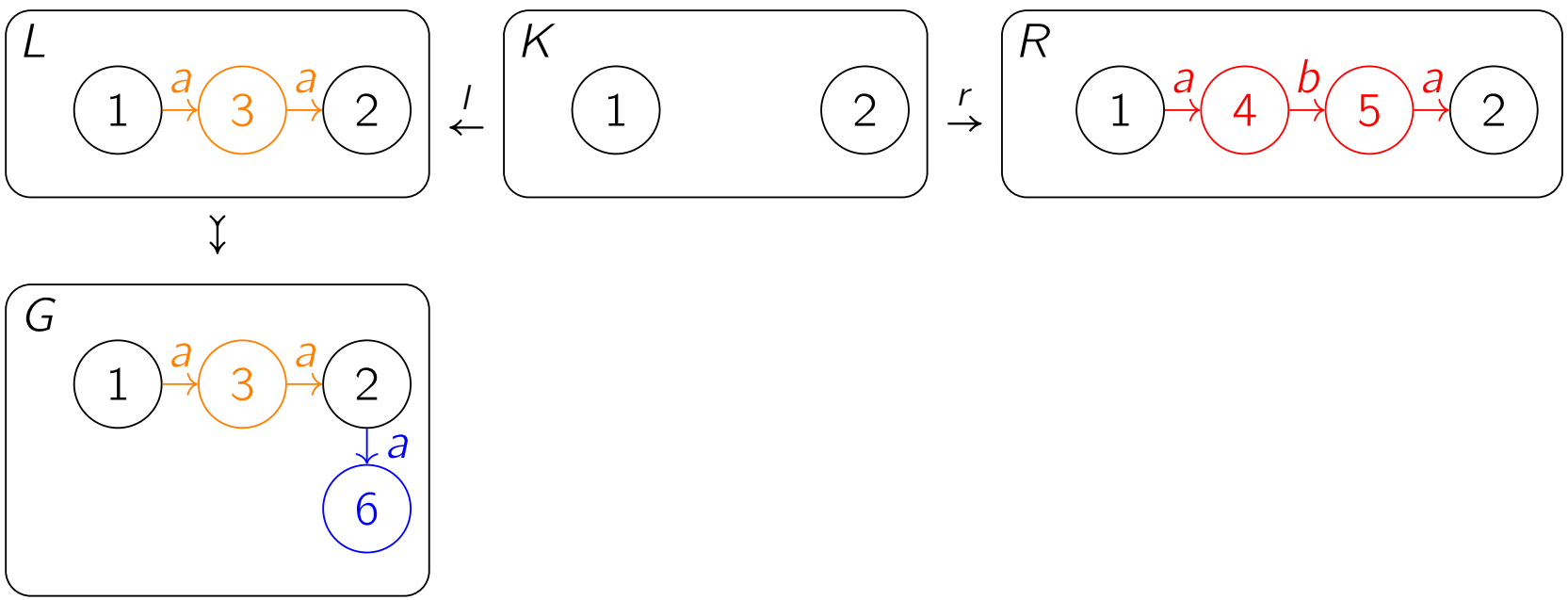
Rewriting rule with interface K



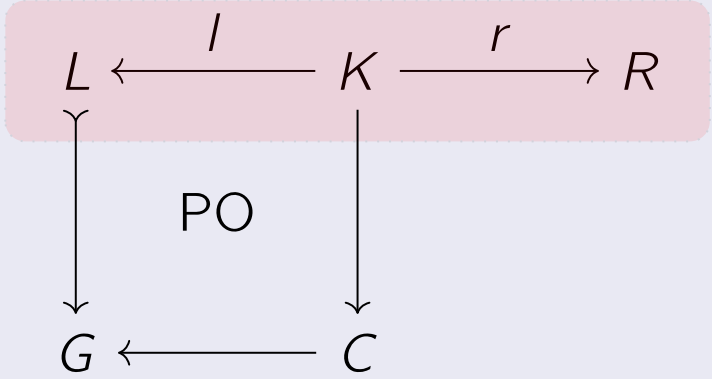
Graph Rewriting with Double-Pushout (DPO)



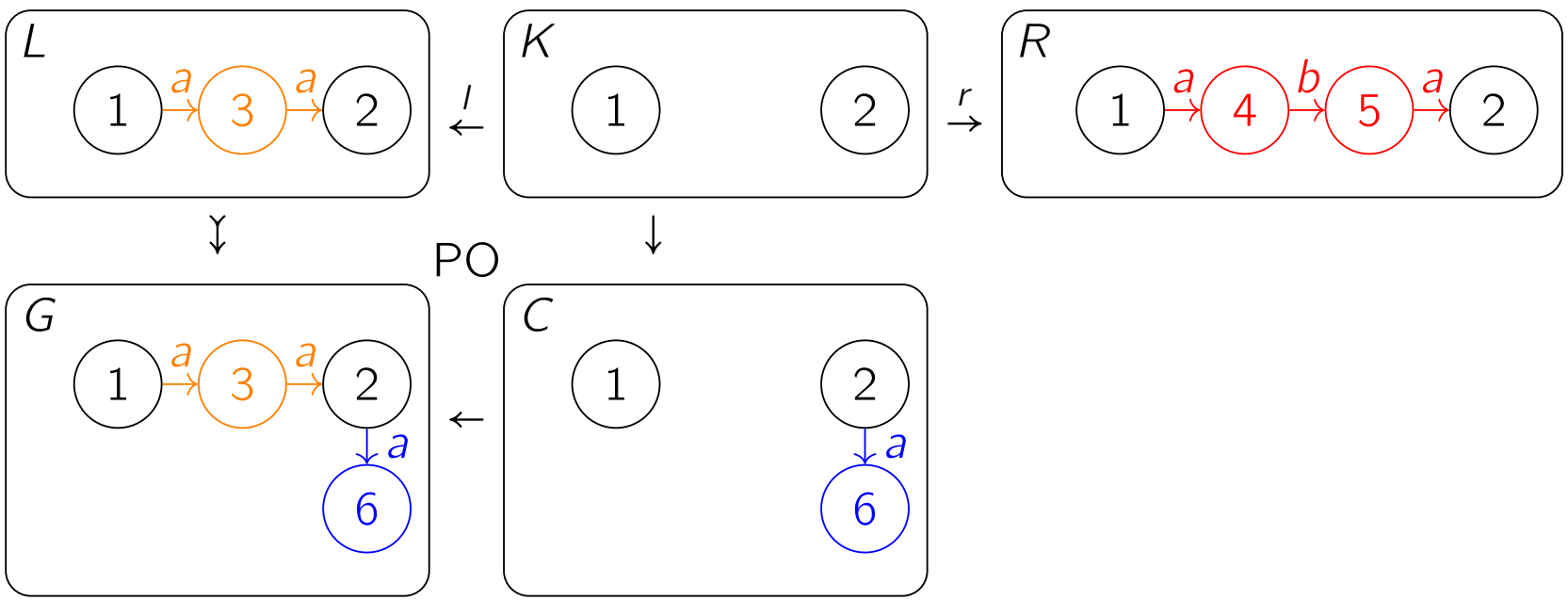
Rewriting rule with interface K



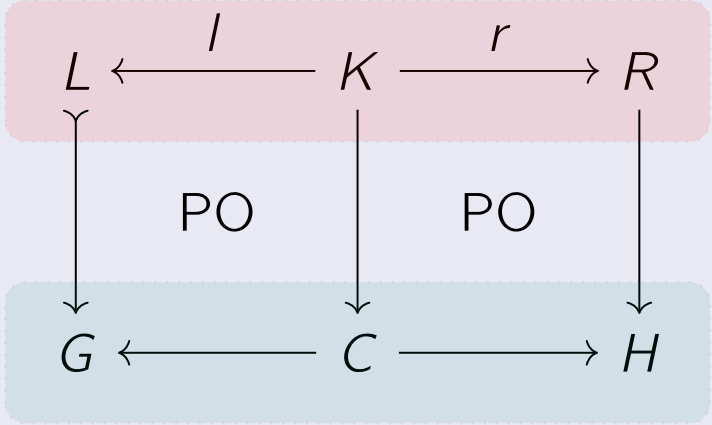
Graph Rewriting with Double-Pushout (DPO)



Rewriting rule with interface K

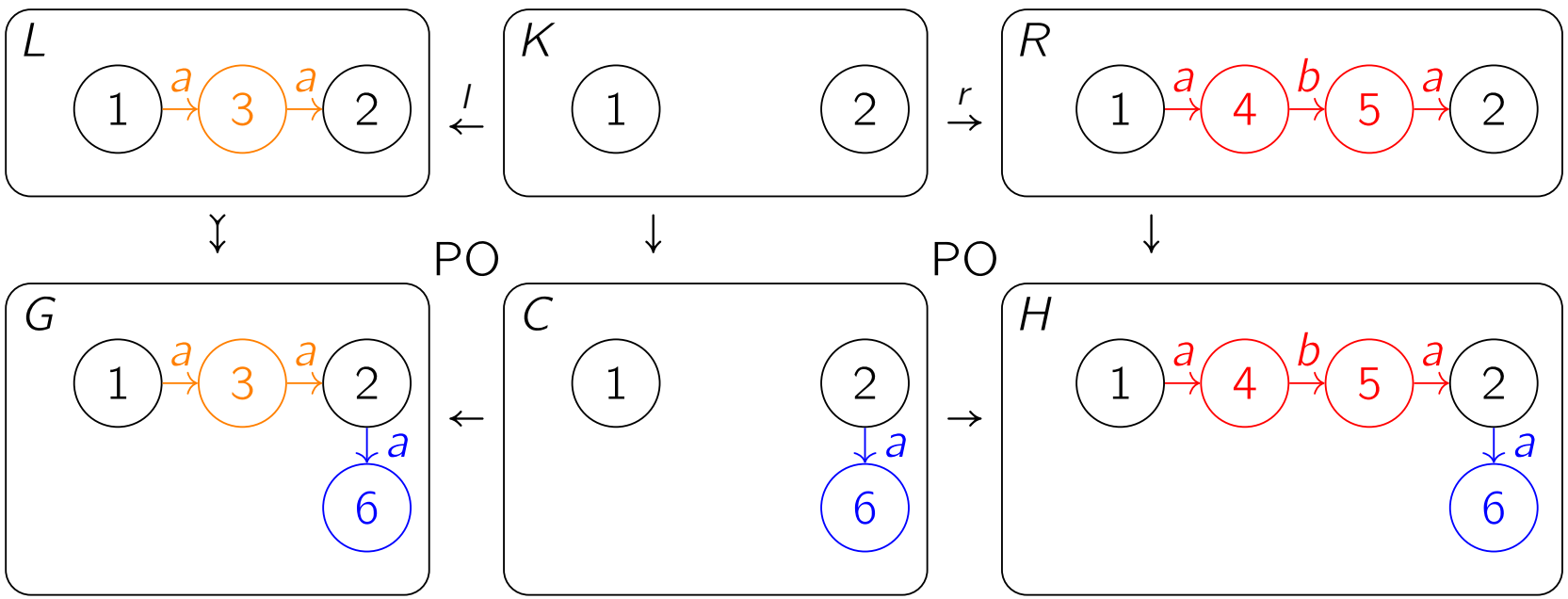


Graph Rewriting with Double-Pushout (DPO)



Rewriting rule with interface K

rewriting step $G \Rightarrow H$



Weighted Type Graph Method

Termination by interpretation

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

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

Interpretation:

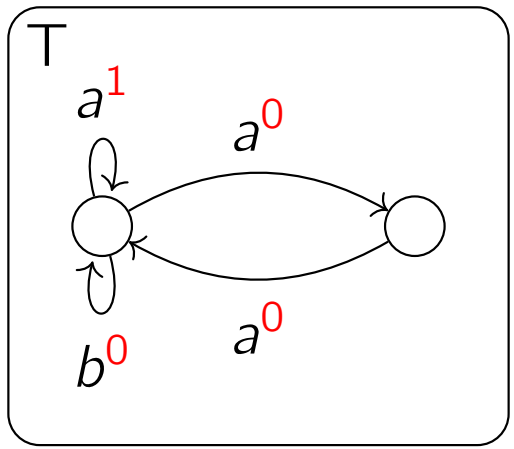
$$\begin{aligned} G &\rightsquigarrow \mathcal{F}(G, T) \\ &\rightsquigarrow \mathrm{weight}(\mathcal{F}(G, T)) \\ &\rightsquigarrow \textsf{aggregator}(\mathrm{weight}(\mathcal{F}(G, T))) \in \mathbb{N} \end{aligned}$$

What is the weight of a morphism?

What is the weight of a graph?

Weighted Type Graph

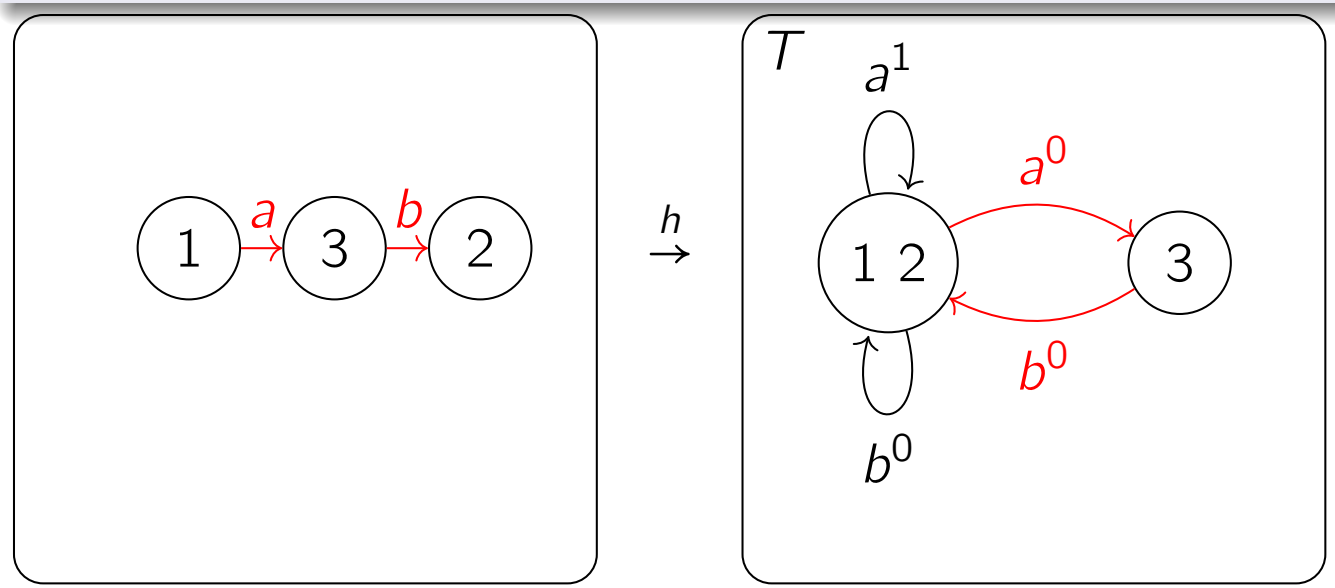
In the context of graph rewriting, a weighted type graph is a graph with weights on its edges.



Morphism Weight

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

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

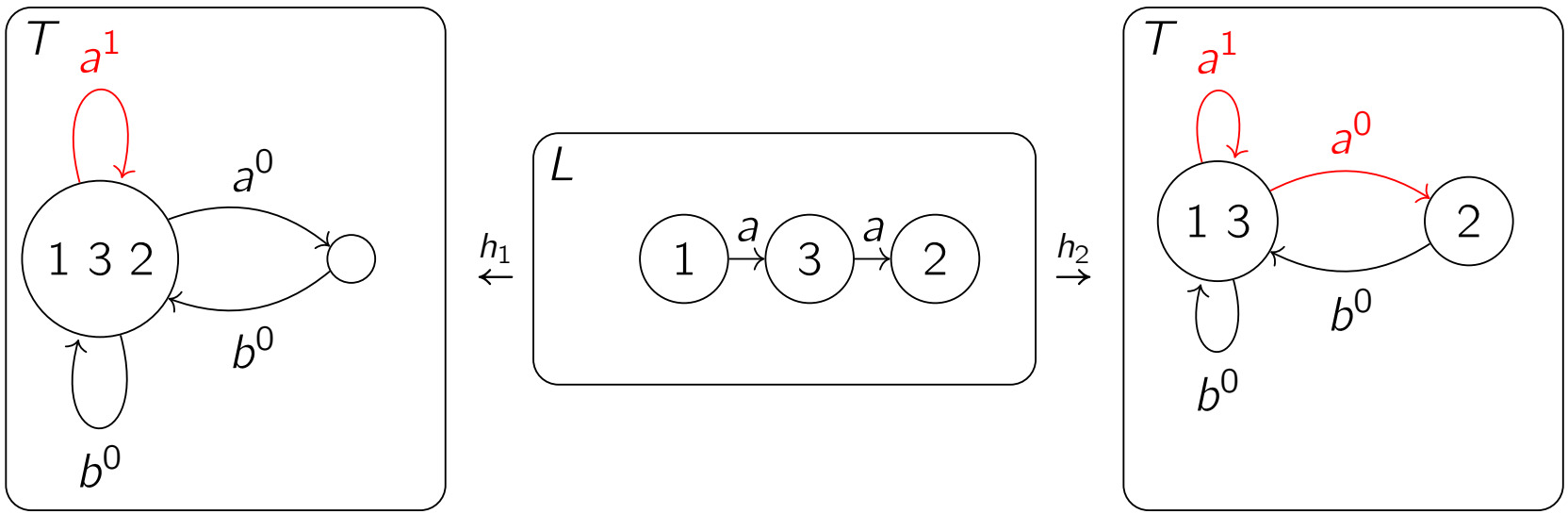


$\mathsf{weight}_T(h) = 0 + 0 = 0$

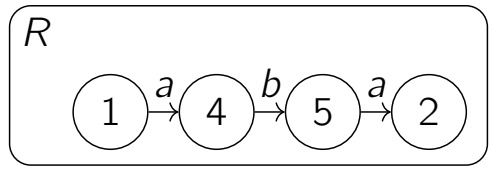
Graph Weight

The weight of a graph L is

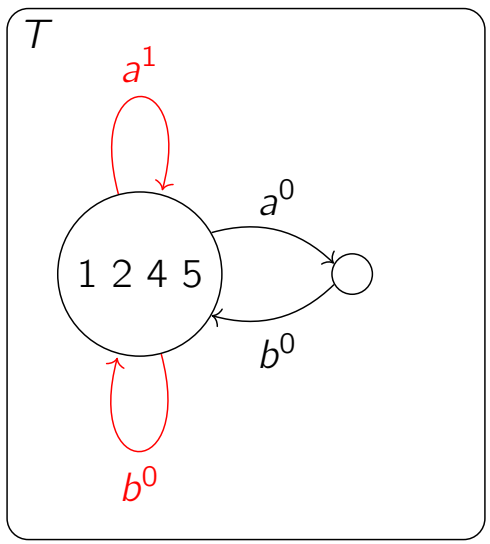
$$\min\{h : L \to T \mid \text{weight}_T(h)\}$$

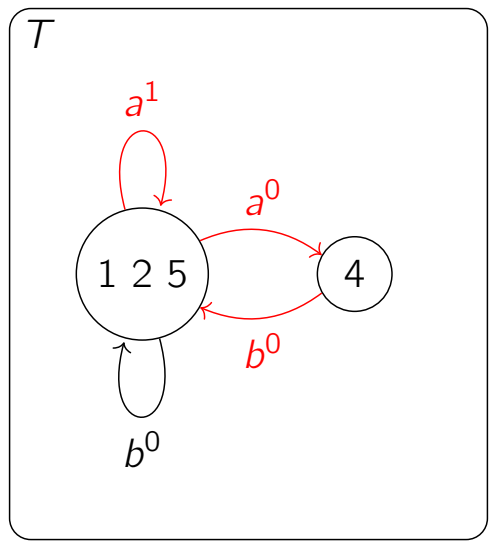


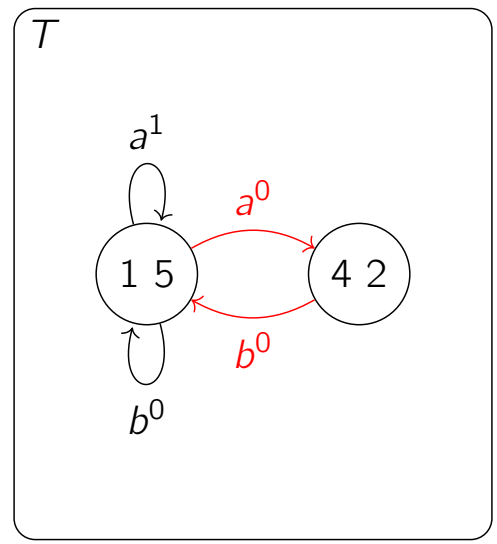
$$\text{weight}_T(L) = \min\{\text{weight}_T(h_1), \text{weight}_T(h_2)\} = \min\{1+1, 1+0\} = 1$$



has three morphisms to T:

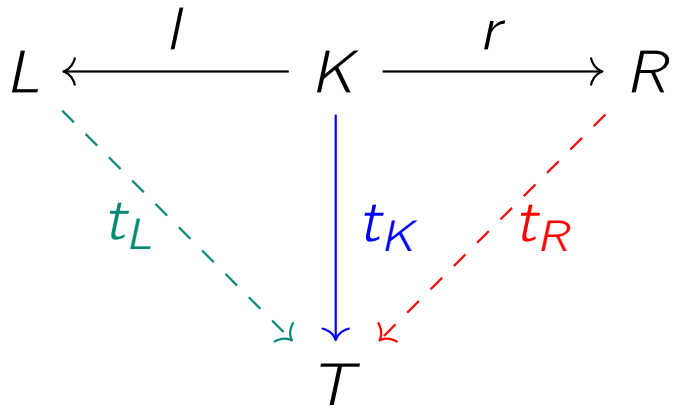






$$\mathsf{weight}_T(R) = \mathsf{min}\{1+0+1, 0+0+1, 0+0+0\} = 0.$$

Termination Criterion



Every rewriting step strictly decreases the weight if

- for all t_K, if there is t_L making $\triangle KLT$ commutative, then

$$\begin{aligned} &\mathbf{min}\{\mathbf{weight}_T(t_L) \mid \triangle KLT \,\mathrm{commutative}\} \\ >\, &\mathbf{min}\{\mathbf{weight}_T(t_R) \mid \triangle KRT \,\mathrm{commutative}\} \end{aligned}$$

How to find a suitable weighted type graph ?

Searching for Weighted Type Graphs

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 determine 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 $2k^2m$.

Solution

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 Real Numbers

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 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	$\approx 10^{181}$

With real numbers as weights:

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

Linear programs can be solved in polynomial-time on average.

Experimental Results

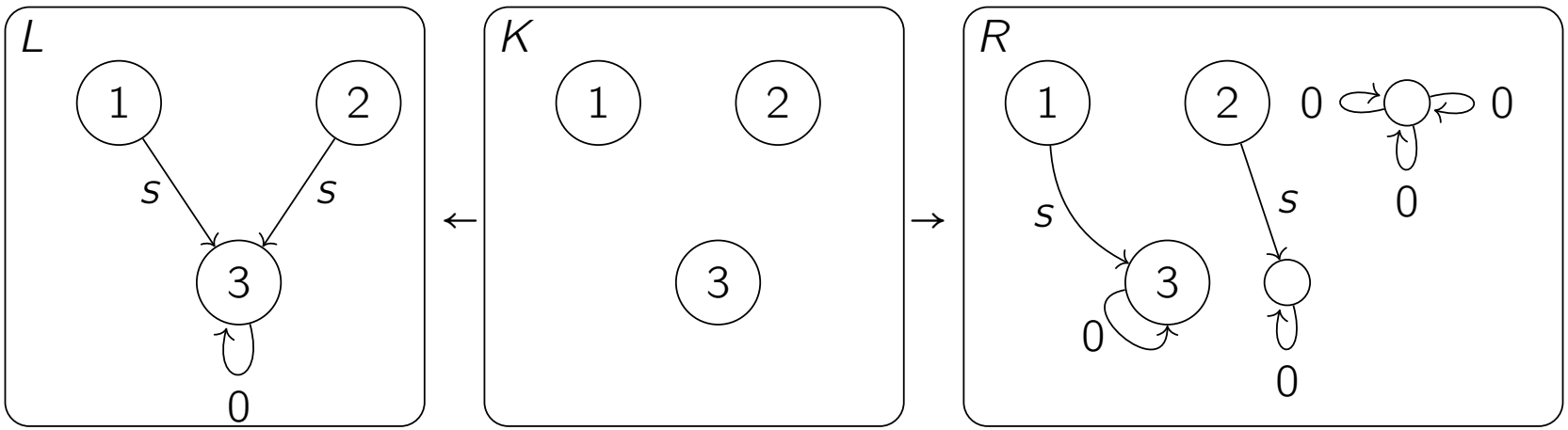
	A	a	T	t	N	n
[EO24, Example 6.3]					2.74	1.16
[EO24, Example D.3]	2.25	1.18			2.24	1.18
[Plu95, Example 3.8]	2.95	1.90	2.94	1.87	3.49	1.87
[Plu18, Example 4]	4.26	3.19	4.24	3.13	5.82	timeout
[Plu18, Example 5]	5.54	5.55	5.53	5.50	9.11	5.62
[Bru+15, Example 4]	2.44	2.46	2.47	2.54	4.58	2.46
[Bru+15, Example 5]					7.80	timeout
[Bru+15, Example 6]					9.75	timeout
[BKZ14, Example 1]	2.26	1.18			2.24	1.18
[BKZ14, Example 4]	2.25	1.22	2.24	1.18	2.25	1.19
[BKZ14, 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.

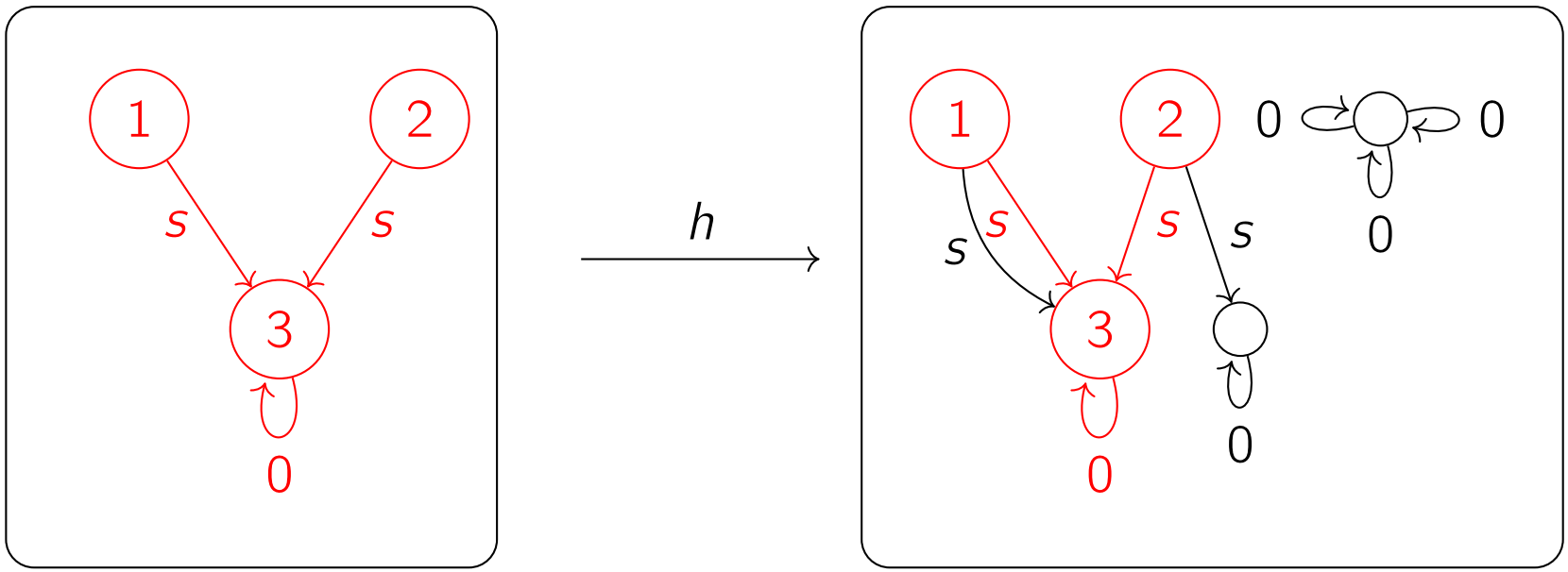
Morphism Counting

Termination by interpretation

Parameter: a graph X

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

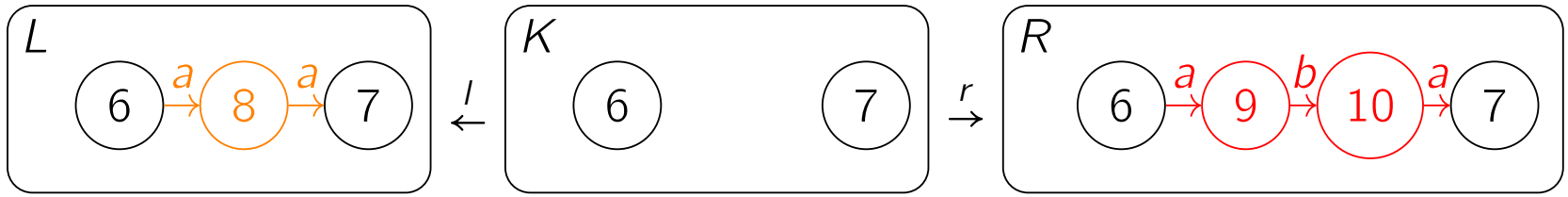
Inclusions: morphisms h with $h(x) = x$ for all x.



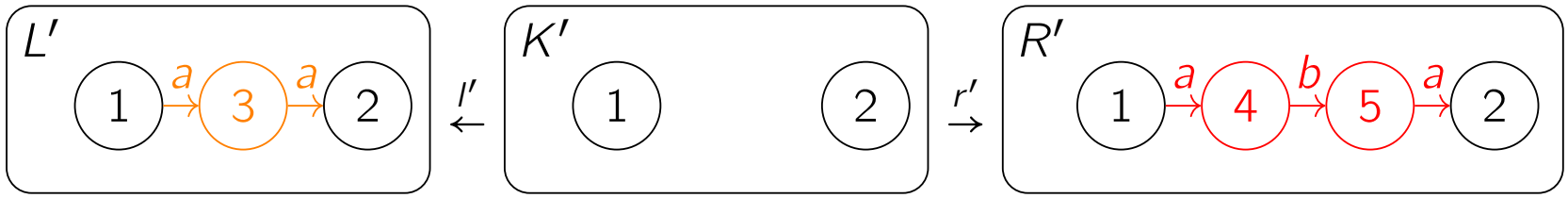
G is a subgraph of H.

Graph Rewriting with Injective Rules

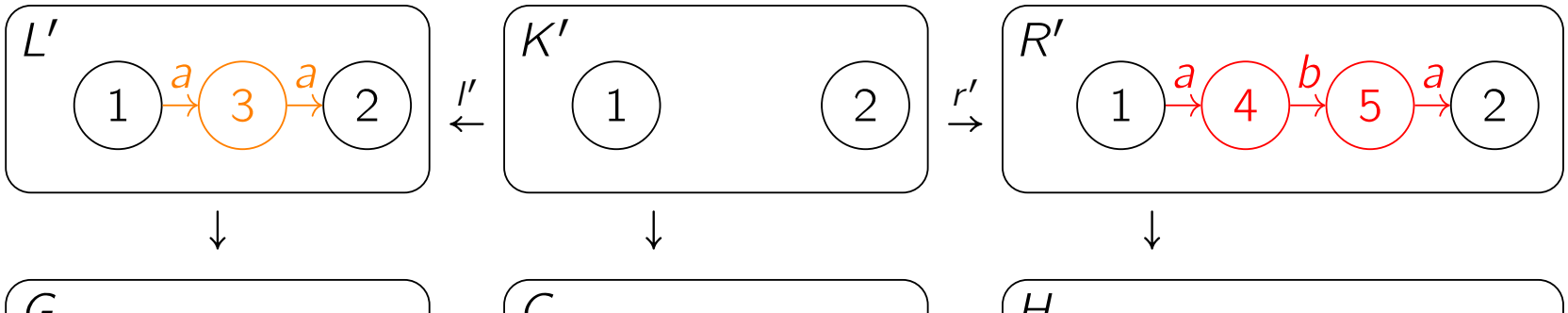
A rewriting rules consists of two inclusions.

$\varphi =$ 

An equivalent rewriting rule expresses the same transformation.

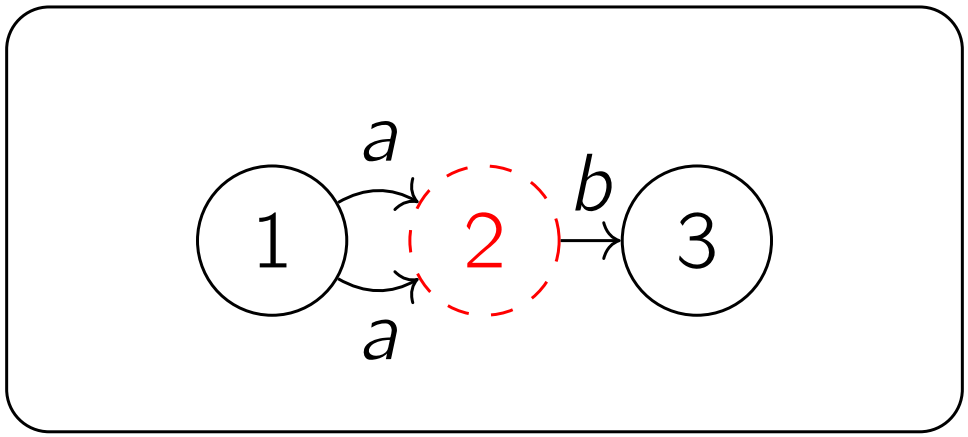
$\varphi' =$ 

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



Pre-Graphs

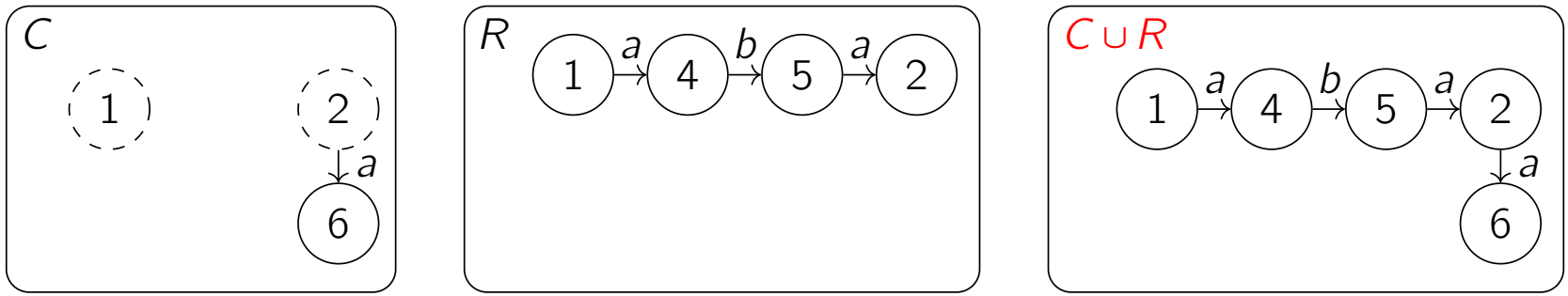
Pre-graphs are graphs with missing nodes and dangling edges.



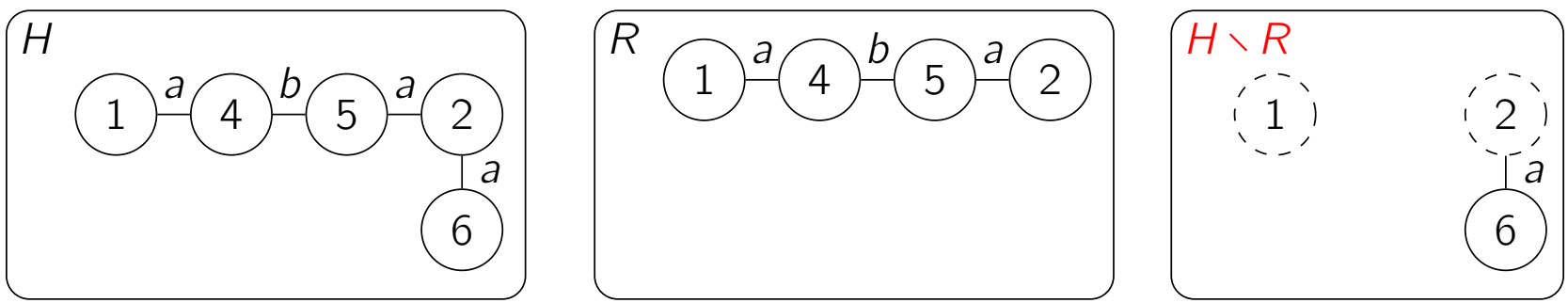
- 1 missing node in red,
- all edges are dangling,
- 2 existing nodes.

Pre-Graph Operations

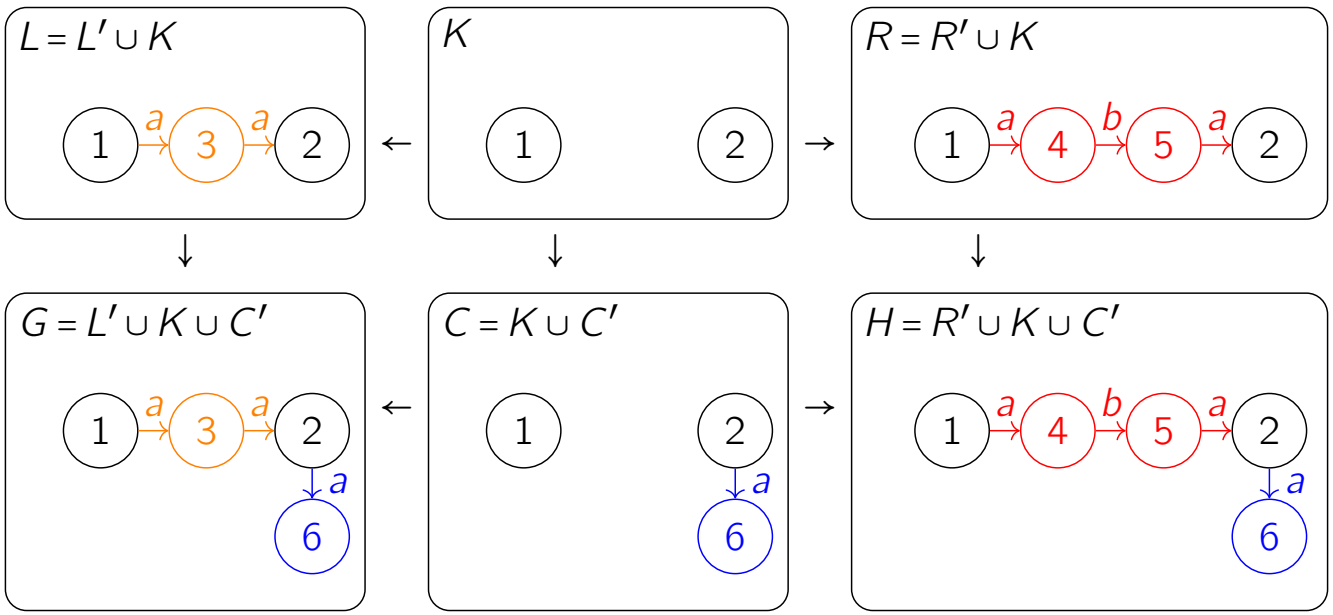
Union of two pre-graphs $C \subseteq G$ and $R \subseteq G$, denoted $C \cup R$.



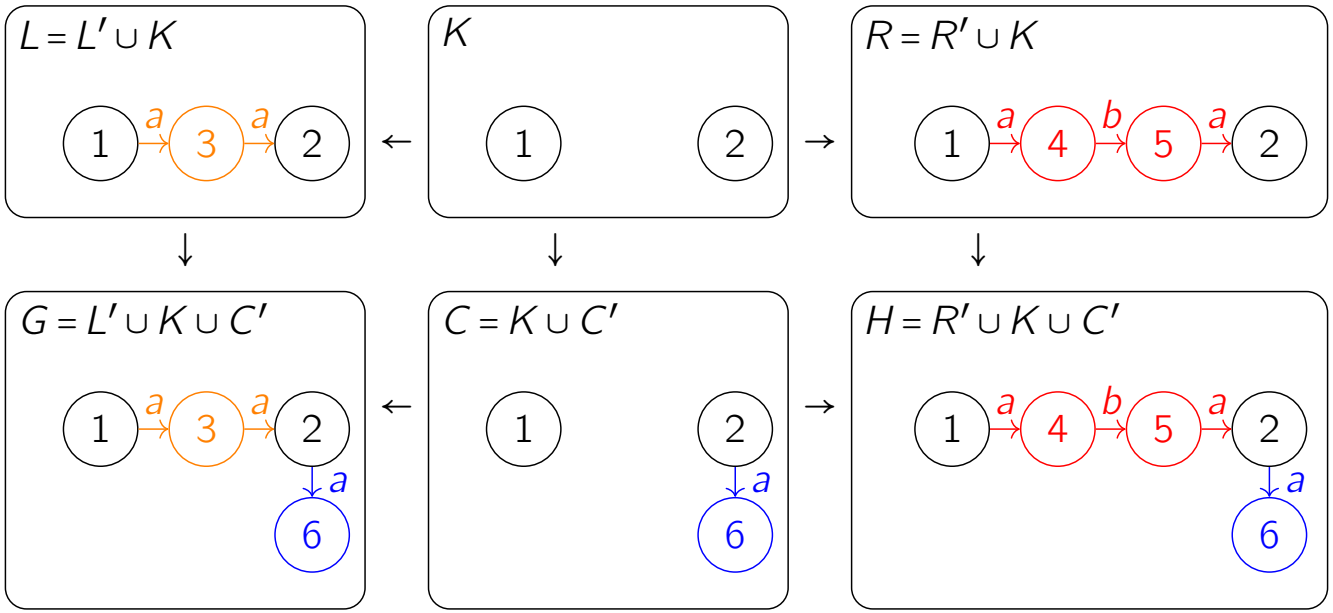
Relative complement of R in H where $R \subseteq H$, denoted $H \setminus R$.

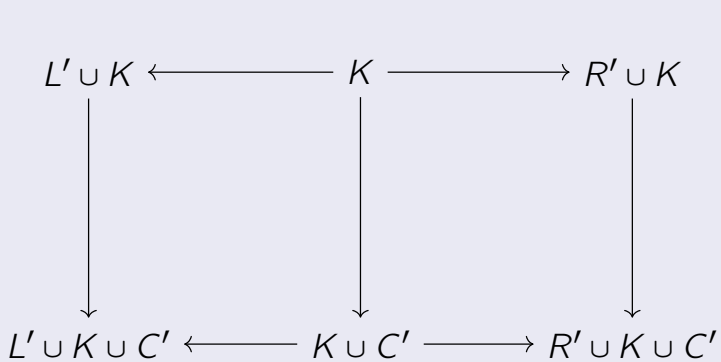


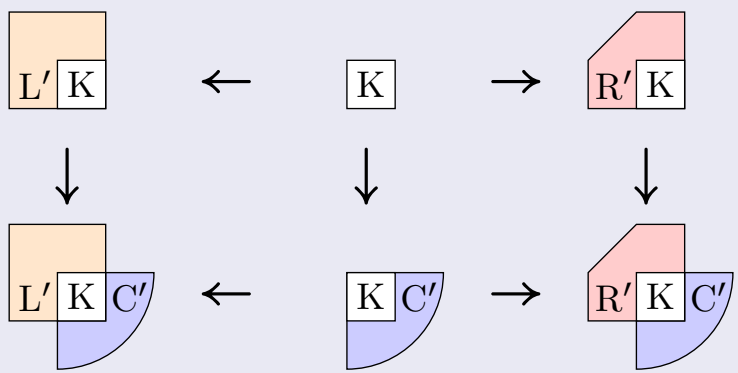
Decomposition of Graphs in Rewriting Steps



Decomposition of Graphs in Rewriting Steps

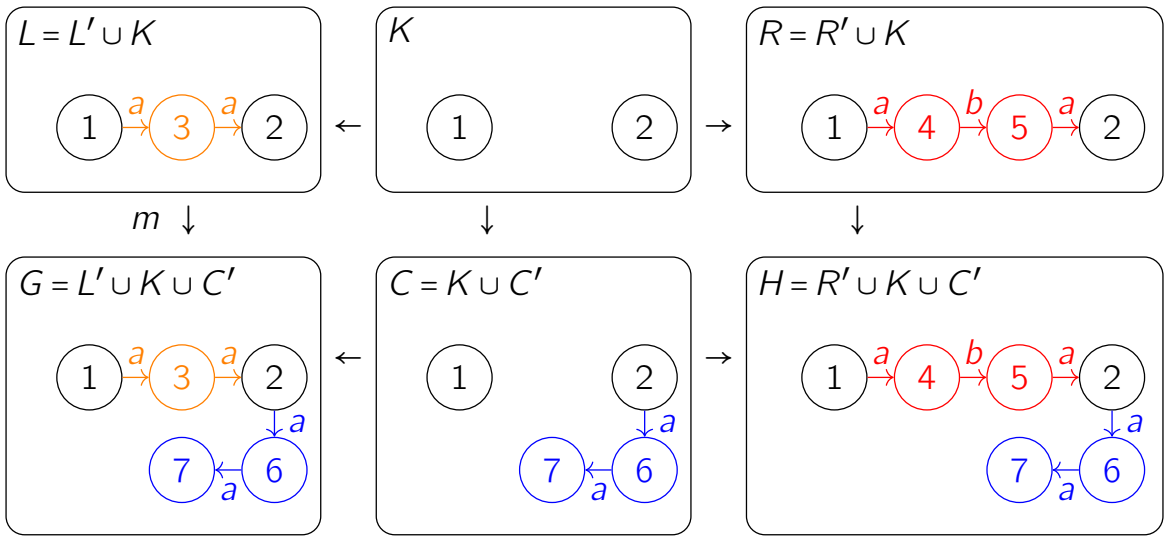




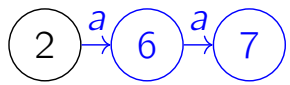
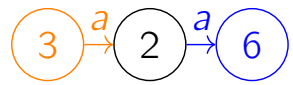
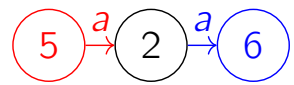


Implicit, Explicit and Shared Morphisms

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

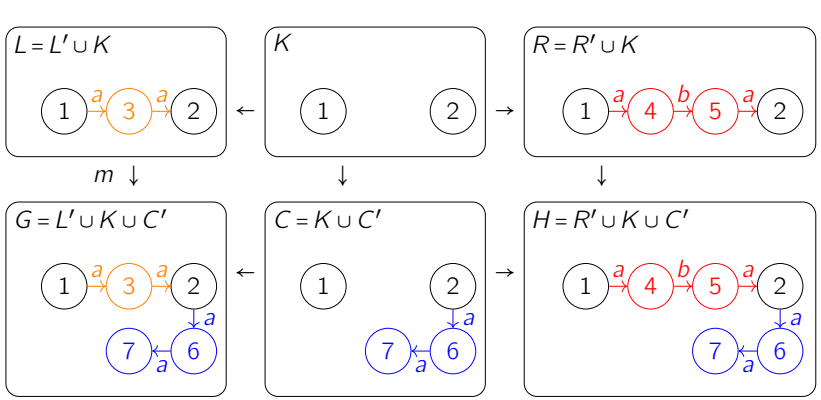


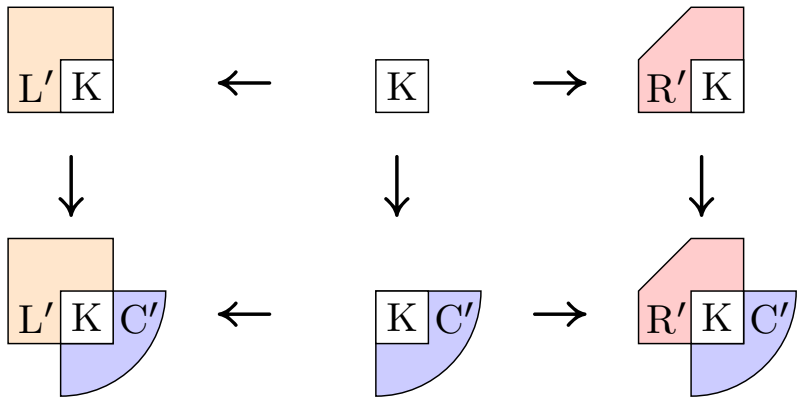
The morphisms from X has their images:

- in G and L: 
- in H and R: None
- shared by G and H: 
- formed by subgraphs of L and C: 
- formed by subgraphs of R and C: 

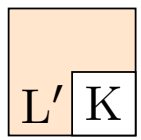
Implicit, Explicit and Shared Morphisms

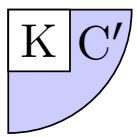
An X-occurrence is an injective morphism from X.

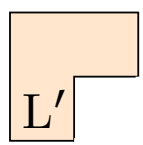


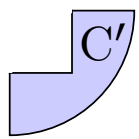


An X-occurrence is

- explicit if $\mathsf{Im}(x)$ is included in 

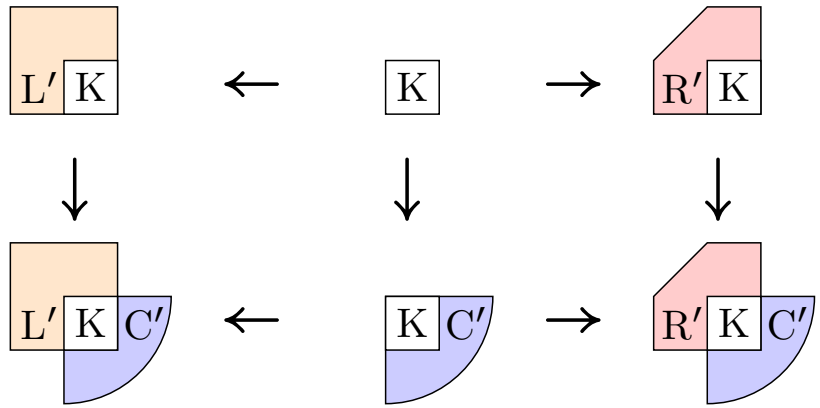
- shared if $\mathsf{Im}(x)$ is included in 

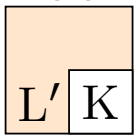
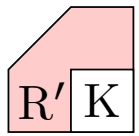
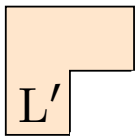
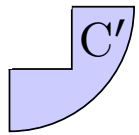
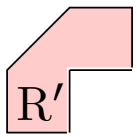
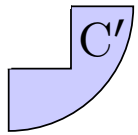
- implicit if $\mathsf{Im}(x)$ has elements in both 

and 

Similarly, in H.

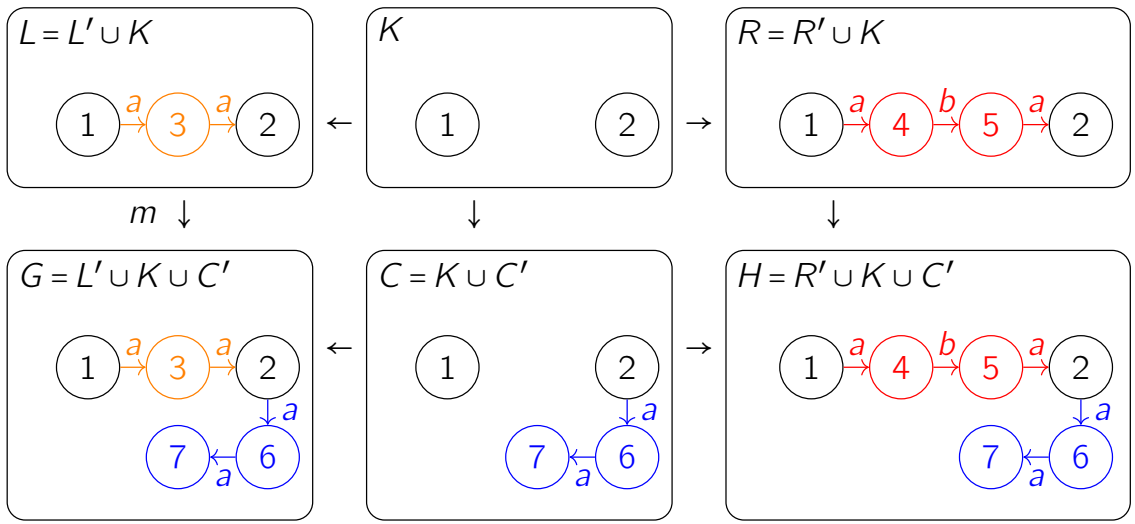
A Sufficient Condition for Termination



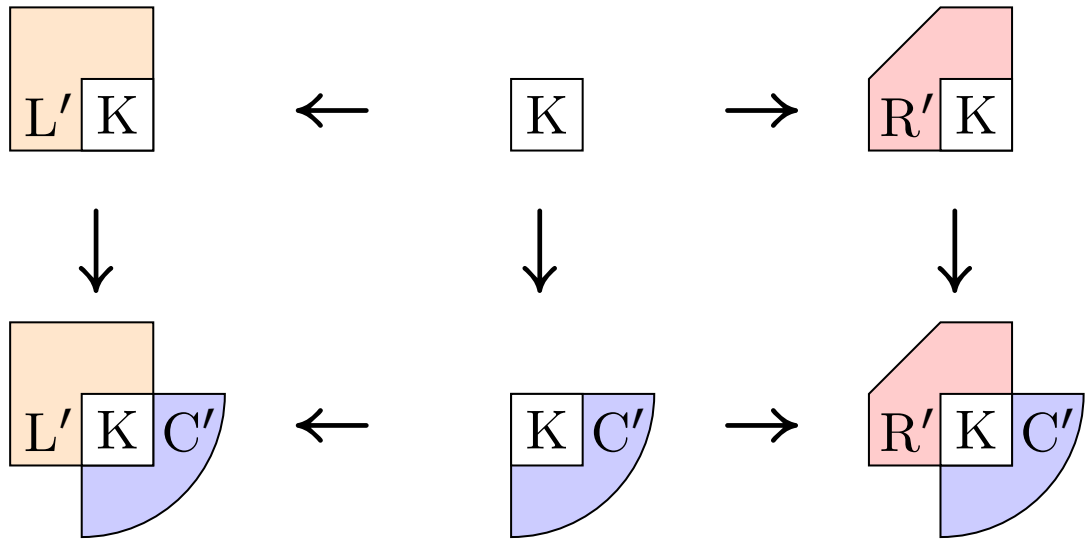
Suppose that there are strictly more X-morphisms with image in  then in  , φ terminates if, in every rewriting step, there are more X-morphisms whose image has elements in both  and  then X-morphisms whose image has elements in both  and  .

Challenge: the pregraph C' is unknown.

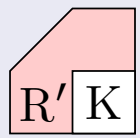
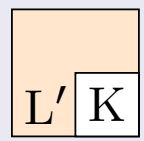
Analysis of Implicit Occurrences



Analysis of Implicit Occurrences



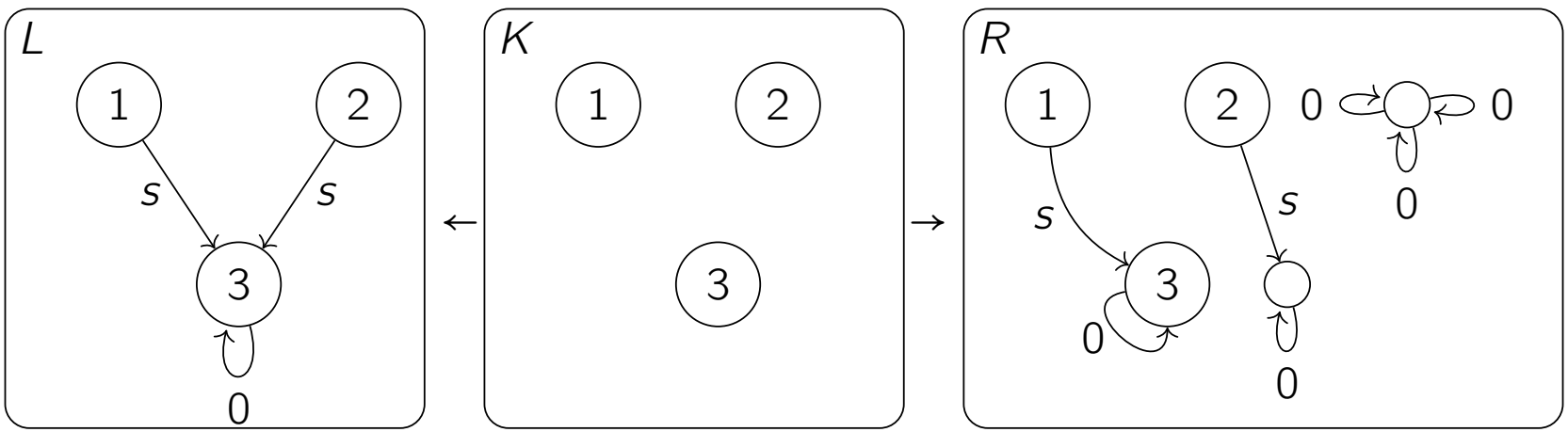
There are more implicit X-occurrences before rewriting, if

- all subgraphs of  that can form an implicit X-occurrence in some rewriting step can be mapped to distinct subgraphs in  while preserving the interface elements.

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

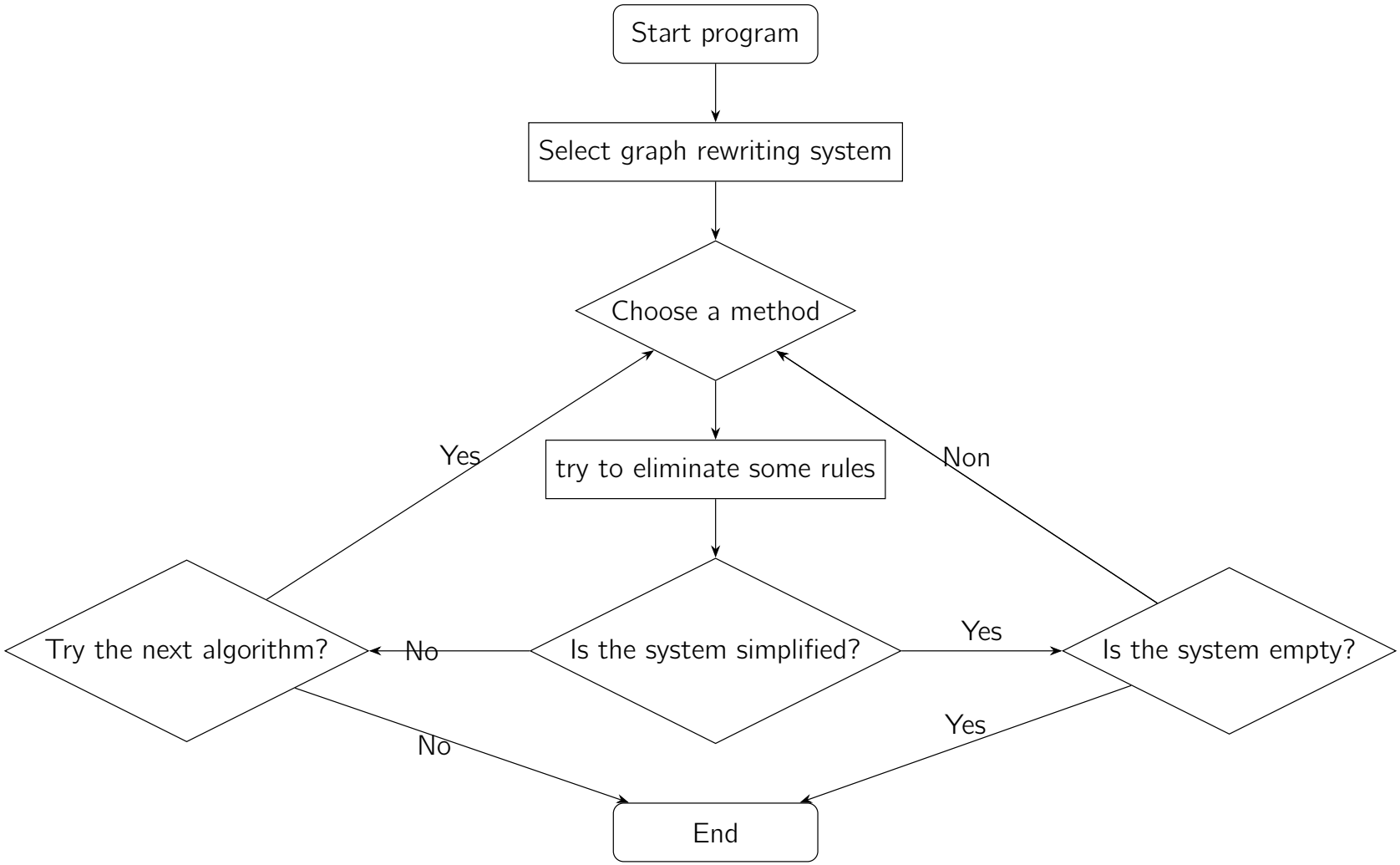
LyonParallel

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

[BKZ14] H. J. Sander Bruggink, Barbara König, and Hans Zantema. "Termination Analysis for Graph Transformation Systems". In: *Theoretical Computer Science - 8th IFIP TC 1/WG 2.2 International Conference, TCS 2014, Rome, Italy, September 1-3, 2014. Proceedings*. Ed. by Josep Diaz, Ivan Lanese, and Davide Sangiorgi. Vol. 8705. Lecture Notes in Computer Science. Springer, 2014, pp. 179–194. DOI: 10.1007/978-3-662-44602-7_15.

[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.

[EO24] J. Endrullis and R. Overbeek. *Generalized Weighted Type Graphs for Termination of Graph Transformation Systems*. 2024. arXiv: 2307.07601v2 [cs.LO].

[Plu18] Detlef Plump. "Modular Termination of Graph