Université Claude Bernard Lyon 1

THESE de DOCTORAT DE
L'UNIVERSITE CLAUDE BERNARD LYON 1

Ecole Doctorale N° 512
Informatique Mathématiques

Soutenue publiquement le 16/12/2025, par :
Qi QIU

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

---

Devant le jury composé de :

M.	**Philippe MALBOS**	**Président**
Professeur des universités, Université Claude Bernard Lyon 1		
M.	**Jean KRIVINE**	**Rapporteur**
Chargé de recherche, CNRS Paris		
Mme	**Pascale LE GALL**	**Rapporteure**
Professeure des universités, CentraleSupélec Gif-sur-Yvette		
Mme	**Évelyne CONTEJEAN**	**Examinatrice**
Directrice de recherche, CNRS Gif sur Yvette		
M.	**Frédéric PROST**	**Examinateur**
Maître de conférences, INSA Lyon		
Mme	**Clara BERTOLISSI**	**Examinatrice**
Professeure des universités, INSA Centre Val de Loire		
M.	**Pierre HYVERNAT**	**Examinateur**
Maître de conférences, Université Savoie Mont Blanc		
M.	**Xavier URBAIN**	**Directeur de thèse**
Professeur des universités, Université Claude Bernard Lyon 1		

Résumé

La réécriture de graphes est un cadre formel qui décrit comment des graphes peuvent être transformés en appliquant un ensemble de règles, appelé système de réécriture de graphes. Nous nous intéressons à la preuve automatique de la terminaison de ces systèmes, c'est-à-dire aux techniques permettant de montrer que, pour tout graphe, l'application des règles de transformation d'un système donné ne peut se répéter indéfiniment.

Nous contribuons au développement de techniques automatiques d'analyse de la terminaison des systèmes de réécriture de graphes avec double pushout (DPO), en proposant quatre apports.

Premièrement, nous étendons une technique existante, reposant sur des graphes de type. Cette méthode attribue des poids à des morphismes ciblant un graphe de type; le poids d'un graphe sujet à transformation est défini comme la somme des poids de tous les morphismes depuis ce graphe vers le graphe de type. La terminaison est alors établie en montrant que le poids du graphe diminue strictement à chaque application d'une règle de transformation, à condition que l'ensemble des poids soit bien fondé. Exiger que les poids soient bien fondés peut, cependant, se révéler limitant en pratique, car la recherche d'un graphe de type adapté est extrêmement difficile. Notre extension rend cette recherche beaucoup plus facile en autorisant des poids issus d'ensembles non bien fondés.

Deuxièmement, nous développons une nouvelle condition suffisante, appelée comptage de morphismes. Cette méthode s'applique aux systèmes de réécriture de graphes avec DPO à règles injectives sur des multigraphes orientés dont les arêtes sont étiquetées. Elle repose sur l'idée que la réécriture ne peut pas durer indéfiniment si le nombre de morphismes ayant un sous-graphe donné comme domaine diminue strictement à chaque transformation.

Troisièmement, nous étendons notre méthode de comptage de morphismes pour une analyse plus fine. Plus particulièrement, nous considérons les morphismes ayant un sous-graphe spécifique comme domaine et dont l'image n'est pas incluse dans un contexte interdit.

Nous mettons enfin en œuvre ces techniques en les implantant au sein d'un nouvel outil logiciel : LyonParallel. Dédié à la transformation avec DPO des multigraphes orientés étiquetés sur les arêtes, cet outil permet d'exhiber automatiquement la terminaison d'exemples que les approches antérieures ne pouvaient pas traiter.

Abstract

Graph rewriting is a formal framework describing how graphs can be transformed by applying a set of rules, called a graph rewriting system. We focus on the automated proof of termination of these systems, that is, techniques for showing that, for any graph, the application of the transformation rules of a given system cannot repeat indefinitely.

We contribute to the development of automated techniques for the termination analysis of graph rewriting systems using the double pushout (DPO) approach, by proposing four contributions.

Firstly, we extend an existing technique based on type graphs. This method assigns weights to morphisms targeting a type graph; the weight of a graph subject to transformation is defined as the sum of the weights of all morphisms from that graph to the type graph. Termination is then established by showing that the graph's weight strictly decreases with each application of a transformation rule, provided that the set of weights is well-founded. Requiring the weights to be well-founded can, however, prove limiting in practice, because the search for a suitable type graph is extremely difficult. Our extension makes this search much easier by allowing weights drawn from non-well-founded sets.

Secondly, we develop a new sufficient condition, called morphism counting. This method applies to DPO graph rewriting systems with injective rules on edge-labeled directed multigraphs. It is based on the idea that rewriting cannot last indefinitely if the number of morphisms whose domain is a given subgraph strictly decreases with each transformation.

Thirdly, we extend our morphism counting method for a more fine-grained analysis. More specifically, we consider morphisms whose domain is a specific subgraph and whose image is not included in a forbidden context.

Finally, we implement these techniques in a new software tool, LyonParallel. Dedicated to DPO transformation of edge-labeled directed multigraphs, this tool automatically establishes termination for examples that earlier approaches could not handle.

Acknowledgements

First of all, I thank my supervisor, Xavier Urbain, for providing me with the opportunity to work on this subject and for his guidance not only in my research but, more importantly, in my personal development: his integrity is a great example for me to follow. I am particularly grateful for the academic freedom he afforded me. Midway through my PhD, we had different opinions regarding the research direction, and he generously allowed me to pursue the path which inspired me the most. This freedom has been invaluable to me, and I am grateful for his kindness. I am also grateful for his careful, tireless reading of this dissertation; when I initially disagreed with his suggestions he was patient and later restated them in a way I found considerate. His help significantly improved the quality of my writing.

I am grateful to the members of the jury: Philippe Malbos, Jean Krivine, Pascale Le Gall, Évelyne Contejean, Frédéric Prost, Clara Bertolissi, Pierre Hyvernat, for their time, careful reading and constructive comments.

I thank Evelyne Contejean and Victor Ostromoukhov for serving as members of the "Comité de Suivi de Thèse"; Pierre Courtieu for his help in the early stages of my PhD and for his extraordinary kindness; Rachid Echahed, Nicolas Peltier, Sophie Tourret, Steve Kremer, Stephan Merz and Simon Roussanaly for hosting me as a temporary teaching and research associate at the University of Grenoble Alpes and at the University of Lorraine; Tom Hirschowitz for his invitation to give a talk at the University of Chambéry, which meant a lot to me, as my admiration for Jean-Jacques Rousseau was the reason I came to France; Zhilin Wu for his constructive questions on the first draft; the anonymous reviewers of my work for their time and valuable feedback; Jean-marie Lagniez for his encouragement during this thesis.

I extend my sincere thanks to all the faculty members who contributed to my academic development during my studies in France, for their generosity in sharing knowledge, and for their kindness and patience towards their students. I am especially grateful to Benoît Mariou, who taught my first course in logic; it was in that class that I realized logic was what I had been looking for.

I thank my parents, Shuangluan Ye and Guowen Qiu, for their unconditional love and sacrifices, which have allowed me to pursue my dreams, and my wife, Lu Wang, for her patience and understanding during the challenging times of my PhD.

I thank my friend Pierre Bourquat, the most authentic person I have ever met. I still remember one day at the very beginning of my PhD, when I was working in my cubicle, he gently tapped on the glass side of my cubicle and asked me if I was new ... then introduced me to other friends Thomas Ranvier, Auday Berro, Mehdi Hennequin and Yacine Gaci. I also thank Zhongyun Yang, Yaqing Fang, Jiayi Wei, Chukun Zhang, Xin Xie, and all those whose paths crossed with mine during my years of study in France.

The author acknowledges the use of large language models, but solely for the purpose of proofreading and wording edits.

Finally, I would like to express my gratitude to the French National Research Agency (ANR) for funding the project Safe, Adaptive, and Provable Protocols for Oblivious Robots Operation (SAPPORO) under Grant 2019-CE25-0005, which has provided essential resources and support for my research endeavors.

Contents

Chapter 1

Introduction

1.1 The correctness of distributed algorithms

Distributed algorithms run on multiple processors without centralized control and are central to both everyday devices (e.g., smartphones) and safety-critical systems (e.g., medical devices, trains, aircraft, spacecraft). Errors in distributed algorithms can have severe consequences, such as the loss of critical data or even human life. Ensuring their correctness is vital but challenging due to the inherent complexity of these systems [46, 55]. The Needham-Schroeder protocol [64]—intended for two computers to verify each other's identity—exemplifies this challenge: a serious security flaw was discovered 17 years after its publication [57]. To address the challenge of uncovering subtle vulnerabilities in distributed algorithms, one natural approach is to test all possible scenarios. This is effective when the number of scenarios is small, but becomes infeasible otherwise. For example, when there are an infinite number of scenarios, program testing may reveal bugs, but cannot prove the absence of bugs—a fundamental limitation that motivates the use of formal verification techniques.

Formal verification techniques are mathematically rigorous methods for proving system correctness. This thesis focuses on automated verification techniques [83, 85, 17, 18, 39, 66] that perform proof search automatically for specific properties, and need little user intervention.

1.2 Automated verification techniques

Automated verification techniques are particularly interesting because they can be implemented as automated tools accessible to non-experts in formal verification while also accelerating verification for experts. To verify a specific property of an algorithm, the user only needs to model the algorithm in the required formalism, and the tool then automatically checks conditions that mathematically imply the target property. To further ensure the correctness of results, advanced tools generate certificates verifiable by proof assistants [24]. Although theoretically incomplete for certain properties (meaning some inputs may yield no answer), well-designed techniques can handle many practical cases and significantly reduce verification effort. To enable such methods, distributed algorithms must be modeled using a formalism that facilitates rigorous analysis

of their properties. Specifically, states and behaviors of distributed algorithms must be formally represented.

1.3 Finite edge-labeled directed multigraphs

Finite edge-labeled directed multigraphs provide an intuitive yet formal way to model states of distributed algorithms: computational units are represented by nodes; communication channels are represented by edges; states of the system are modeled by graphs whose edges have labels representing information encoded in computational units and states of communication channels.

For example, consider the graph $\mathcal{G}$ shown in Figure 1.1, enclosed in a box labeled with its name in the top-left corner. It models a distributed network of six computational units. Each node represents a computational unit; a node's state is indicated by the label of its self-loop; each edge represents a communication channel. In Figure 1.1, the node labeled by A represents an active computational unit, the nodes labeled by N represent neutral computational units, and edges labeled by 0 represent communication channels in state 0.

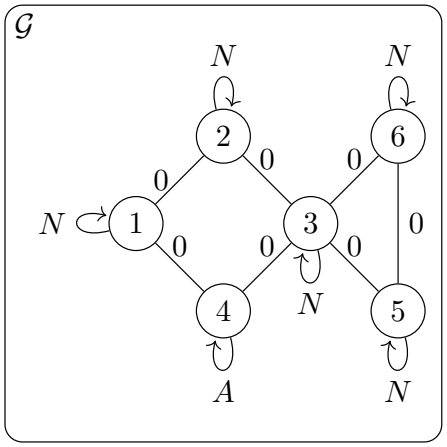


Figure 1.1: A graph modeling a network configuration. An undirected edge between nodes u and v denotes two directed edges from u to v and from v to u. The numbers inside the nodes are node identifiers.

1.4 Graph transformation

Graph transformation provides an intuitive yet formal way to model algorithm behaviors: state changes are modeled by replacing subgraphs of a graph with other subgraphs according to transformation rules.

As an example, consider a distributed spanning-tree construction algorithm on a connected network, where one computational unit is active, all other nodes are neutral, and every channel is initially in state 0. An example initial configuration is shown in Figure 1.1. The algorithm operates as follows: when an active unit detects a neutral neighbor via a channel in state 0, it activates the neighbor and updates the channel to state 1. This behavior is captured by the graph transformation rule in Figure 1.2.

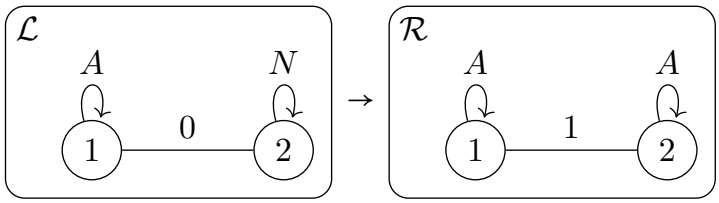


Figure 1.2: Graph rewriting rule for the spanning-tree construction of a connected network with one node labeled by A and all other nodes labeled by N and all edges labeled by 0. It relabels an occurrence of $\mathcal{L}$ to $\mathcal{R}$.

This rule applies as follows: when an active unit detects a neutral neighbor via a channel in state 0, the corresponding subgraph $\mathcal{L}$ appears in the network model and is replaced by the subgraph $\mathcal{R}$, which represents the neighbor's activation and the channel updated to state 1.

Given an initial finite edge-labeled directed multigraph in which all edges are labeled *0* and all nodes are in the neutral state N except for a single node in the active state A, the subgraph induced by edges labeled *1* constitutes a spanning tree once no further rule applications are possible. A sample execution sequence starting from the initial configuration, previously defined and illustrated in Figure 1.1, is shown in Figure 1.3, where self-loops are omitted and their labels are displayed directly on the nodes to improve readability.

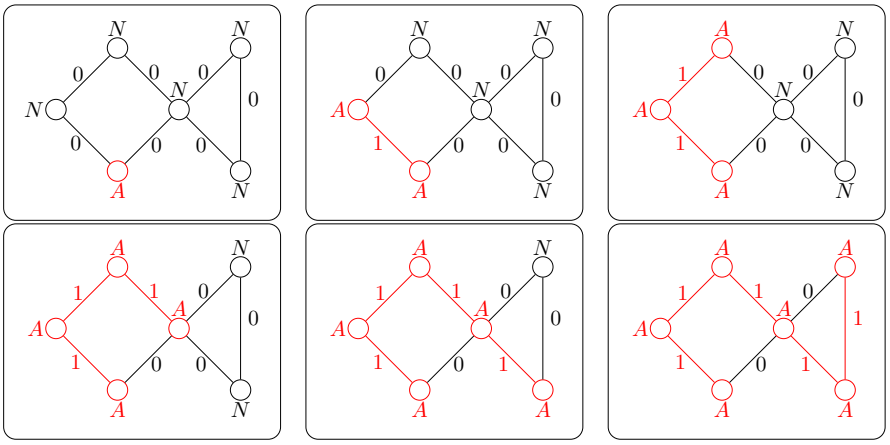


Figure 1.3: Sequence of graph transformation, to be read from left to right and from top to bottom. In each graph, the subgraph in red is to be relabeled according to the rule to obtain the next graph.

Graph transformation has many applications, such as topology-based geometric modeling [76, 75, 9, 10, 68, 2], DNA computing [44], life sciences [8], programming [74], administrative access control policies [11, 12, 52], and software engineering and security [45]. Its broad applicability stems from the fact that graphs and graph-like structures provide a natural and intuitive way to model complex systems.

A predominant school of thought in this area is called algebraic graph rewriting [79, 33, 34], which uses concepts from category theory [69, 5, 21] to define

graph transformation. There are different notions of graphs in the literature for different applications, such as edge-labeled multigraphs [53, 30], hypergraphs [70], attributed graphs [35], and polygraphs [1]. The language of category theory abstracts away from the details of the different notions of graphs, provided that the notion of graph satisfies certain properties [54, 67]. This enables a uniform and elegant definition of graph transformation.

Double-pushout (DPO) graph rewriting [30, 43] is among the most studied algebraic graph transformation formalisms. A DPO graph rewriting rule consists of two functions that preserve graph structure, namely $l : \mathcal{K} \to \mathcal{L}$ and $r : \mathcal{K} \to \mathcal{R}$, Here, $\mathcal{L}$ is the left-hand side graph encoding the rule's preconditions, $\mathcal{R}$ is the right-hand side graph describing the postconditions, and $\mathcal{K}$ is the interface graph—the part common to $\mathcal{L}$ and $\mathcal{R}$ that is preserved by the rule and serves as the interface to the surrounding context. A DPO graph rewriting rule is commonly denoted by $\mathcal{L} \overset{l}{\leftarrow} \mathcal{K} \overset{r}{\rightarrow} \mathcal{R}$ in the literature. For example, the rule in Figure 1.2, can be represented as the rule in Figure 1.4.

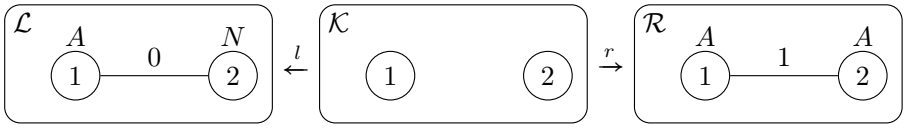


Figure 1.4: A DPO graph rewriting rule replacing an occurrence of $\mathcal{L}$ by an occurrence of $\mathcal{R}$.

To apply a DPO graph rewriting rule $\mathcal{L} \overset{l}{\leftarrow} \mathcal{K} \overset{r}{\rightarrow} \mathcal{R}$ to a host graph $\mathcal{G}$, one must first identify an occurrence of $\mathcal{L}$ in which a subgraph is designated as the interface graph. The other elements of the host graph $\mathcal{G}$, together with the interface graph, form the context graph $\mathcal{C}$ of the rewriting step. The graph $\mathcal{G}$ is thus decomposed into the occurrence of $\mathcal{L}$ and $\mathcal{C}$. Then one modifies the occurrence of $\mathcal{L}$ (by removing some nodes and edges and adding some fresh nodes and edges while keeping the interface graph unchanged, and identifying nodes in the interface graph) to obtain an occurrence of $\mathcal{R}$. Finally, the result graph is obtained by gluing the occurrence of $\mathcal{R}$ with $\mathcal{C}$ via the interface graph $\mathcal{K}$. As an example, consider the graph transformation rule in Figure 1.5.

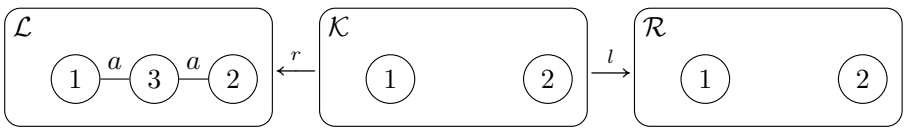
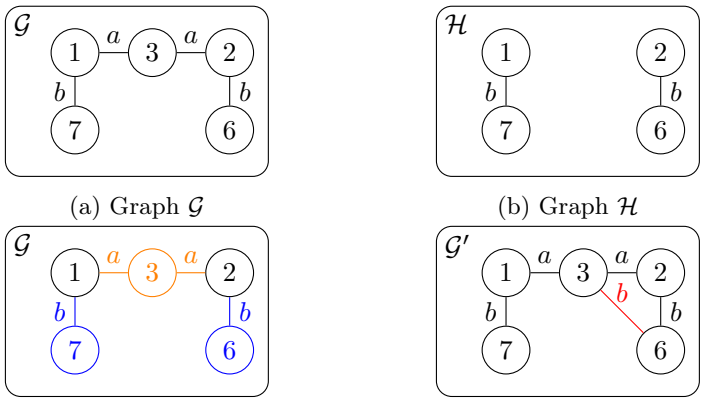


Figure 1.5: A DPO graph rewriting rule removing the middle node and edges of an occurrence of $\mathcal{L}$.

The graph $\mathcal{G}$ shown in Figure 1.6a can be rewritten to yield the graph $\mathcal{H}$ shown in Figure 1.6b by applying the rule in Figure 1.5 to the unique occurrence $1 \overset{a}{-} 3 \overset{a}{-} 2$ of $\mathcal{L}$ in $\mathcal{G}$. Specifically, in this occurrence the subgraph matched to the interface graph $\mathcal{K}$ is the subgraph shown in black in Figure 1.6c; the graph $\mathcal{G}$ is decomposed, as shown in Figure 1.6c, into the occurrence of $\mathcal{L}$ (elements shown in orange and black) and the context graph $\mathcal{C}$ (elements shown in blue and black), glued via the interface graph (elements shown in black). The rule

removes the orange part of the occurrence of $\mathcal{L}$ to obtain an occurrence of $\mathcal{R}$. The result graph $\mathcal{H}$ is obtained by gluing the occurrence of $\mathcal{R}$ and the context graph $\mathcal{C}$ via the interface graph.



(a) Graph $\mathcal{G}$ (b) Graph $\mathcal{H}$

(c) Decomposition of $\mathcal{G}$ (d) Graph $\mathcal{G}'$

Figure 1.6: Application of the rule shown in Figure 1.5: (a) host graph $\mathcal{G}$ with a unique occurrence of $\mathcal{L}$, (b) result $\mathcal{H}$, (c) decomposition of $\mathcal{G}$ into interface K (in black), occurrence of $\mathcal{L}$ (in orange and black), and context C (in blue and black), (d) a graph $\mathcal{G}'$ where the rule is not applicable because deleting the middle node would create a dangling edge (in red).

However, a DPO rewriting rule cannot always be applied to a graph with an occurrence of the left-hand side graph. For example, consider the rule, defined earlier and shown in Figure 1.5. and the graph $\mathcal{G}'$ in Figure 1.6d. The graph $\mathcal{G}'$ is obtained from the graph $\mathcal{G}$ in Figure 1.6a by adding an edge from 3 to 6 (in red). This graph has the same occurrence of $\mathcal{L}$ as the graph $\mathcal{G}$ in Figure 1.6a. However, when 3 is removed, the edge from 3 to 6 becomes dangling.

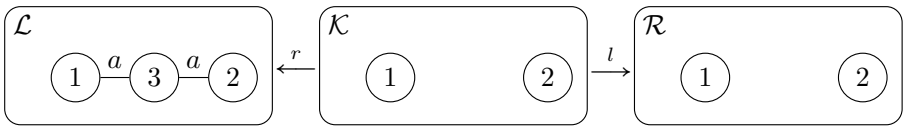
In the literature, different approaches to this problem exist, leading to different algebraic graph rewriting formalisms. Some formalisms implicitly delete dangling edges (e.g., SPO [36]); some impose conditions on contexts in the rule specification (e.g., AGREE [28], PBPO [29], PBPO+ [67]); and others, like DPO, simply forbid rule application that could create dangling edges.

This thesis considers DPO graph rewriting for three reasons. Firstly, it is among the most studied algebraic graph rewriting formalisms, so our techniques have a large potential user base. Secondly, the category-theoretic prerequisites for DPO are minimal: it depends only on pushouts. This matters because category theory's abstract nature can act as a barrier for newcomers. Finally, despite its conceptual simplicity, DPO graph rewriting is expressive enough to model many distributed algorithms [35, 30].

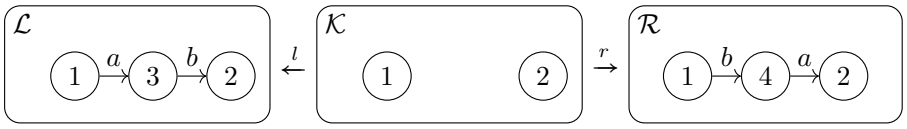
Since graph transformation rules in a system are applied repeatedly (and nondeterministically), they can lead to infinite sequences of graph transformations. However, some properties of systems depend on the absence of infinite sequences. For example, the transformation system, previously defined and illustrated in Figure 1.4, can construct a spanning tree of a connected graph with edges and nodes initially labeled as required if the transformation system terminates. Thus, ensuring termination of a rewriting system is crucial.

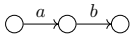
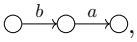
1.5 Termination

Termination is a property of algorithms that ensures they eventually halt. In the context of DPO graph rewriting systems, it ensures that no graph can be transformed indefinitely under a given set of rewriting rules. For example, the following rule terminates because each application (when applicable) reduces the number of nodes in a graph by one:



Consider the following nonterminating rule:



It replaces an occurrence of the graph  with an occurrence of the graph  , keeping the extreme nodes unchanged. A looping sequence of graphs, each obtained by applying the rule to its predecessor, is shown in the following Figure 1.7.

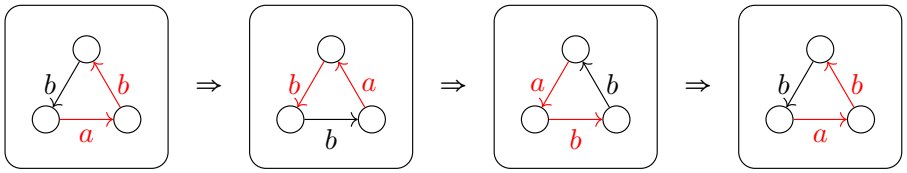


Figure 1.7: Sequence of rewriting steps, to be read left to right. In each graph, the subgraph to be replaced to obtain the next graph is highlighted in red.

Many termination techniques exist for term rewriting systems [3, 4, 24, 26, 31, 42, 62, 82, 60, 80]. Most exploit the tree structure of terms, a feature absent in general graphs, making direct adaptation impossible. Moreover, in many verification tasks, we do not need global termination; instead, we need to show that certain rules cannot occur infinitely often, even in the presence of other rules that may fire forever.

For example, consider the graph transformation system with two rules in Figure 1.8. Rule α deletes an edge, and rule β introduces a fresh isolated node. The system does not terminate because the node-adding rule β can be applied indefinitely. However, the edge-deleting rule α can be applied only a finite number of times: it deletes an edge on each application, and since no rule increases the edge count and the initial graph is finite, only finitely many deletions are possible. Thus, termination depends solely on the node-adding rule β. This observation motivates the notion of *relative termination*.

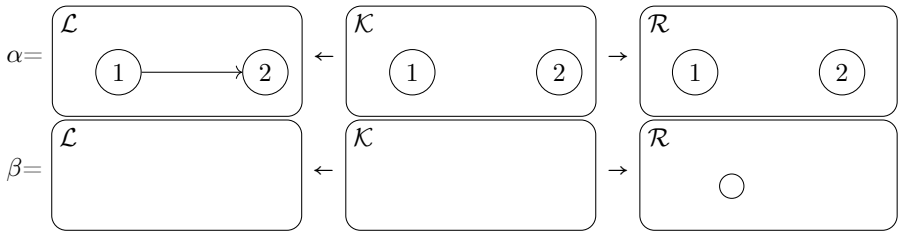


Figure 1.8: Rule α deletes an edge, and rule β adds a fresh node.

1.6 Relative termination

Relative termination was initially introduced for binary relations [51]: let $\mathcal{A}$ and $\mathcal{B}$ be two binary relations, $\mathcal{A}$ is said to be terminating relative to $\mathcal{B}$ if, for any infinite sequence of elements $x_0, x_1, \ldots$ with $(x_i, x_{i+1}) \in \mathcal{A}$ or $(x_i, x_{i+1}) \in \mathcal{B}$ for all $i \in \mathbb{N}$, there are only a finite number of $i \in \mathbb{N}$ such that $(x_i, x_{i+1}) \in \mathcal{A}$.

Since a set of rewriting rules defines a binary relation "object X can be rewritten to object Y using rules from the rule set", called a rewriting relation, the concept of relative termination carries over to rewriting systems in a straightforward way via their associated rewriting relations: a rule set $\mathcal{A}$ terminates relative to another rule set $\mathcal{B}$, if the binary relation induced by $\mathcal{A}$ terminates relative to the binary relation induced by $\mathcal{B}$. As an example, consider the edge-deleting rule α and the node-adding rule β, defined earlier and shown in Figure 1.8. The rule set $\{\alpha\}$ terminates relative to the rule set $\{\beta\}$, because any infinite sequence of graph transformations can apply the edge-deleting rule only a finite number of times.

Relative termination generalizes termination in the context of rewriting, because a set of rules terminating relative to an empty set is terminating. In practice, to prove termination of a rewriting system $\mathcal{R}$, one partitions the set of rules into two disjoint subsets $\mathcal{B}$ and $\mathcal{A}$ with non-empty $\mathcal{A}$ such that $\mathcal{A}$ terminates relative to $\mathcal{B}$, if $\mathcal{B} = \varnothing$ then the termination of $\mathcal{R}$ is established, otherwise, a new iteration starts with the strictly smaller rule set $\mathcal{B}$.

Relative termination has several advantages. Firstly, it supports iterative analysis: once a subset of rules is proved terminating relative to the rest, those rules can be removed from further consideration, allowing us to focus on the remaining rules. Secondly, when combined with Plump's modular termination technique [71]—which proves termination by establishing termination of two rule sets that partition the system's rule set—termination analysis becomes easier.

1.7 Termination of DPO graph rewriting systems on edge-labeled directed multigraphs

We consider DPO graph rewriting systems on edge-labeled directed multigraphs in this thesis, even though, using the language of category theory, it is possible to develop termination techniques for DPO rewriting systems on different graph notions that satisfy certain properties. There are several reasons for this choice.

First of all, edge-labeled directed multigraphs are simple and intuitive: they are just directed graphs with multiple edges between two nodes allowed, and

each edge is labeled by a symbol from a finite alphabet. This notion of graph is widely used in the literature, such as in the work introducing the DPO graph rewriting [37], in previous work [19, 17, 85] on the method that we extend in Chapter 3, and in illustrative examples of works [66, 39] which are closely related to our work in Chapter 4 and Chapter 5.

Secondly, while many constructions can be phrased abstractly (e.g., for adhesive categories [54]), we intentionally work with concrete edge-labeled directed multigraphs to keep the presentation accessible and to exploit graph-specific measures. As a consequence, the correctness of our technique in Chapter 4 and Chapter 5 can be checked easily by users with basic knowledge in graph theory and very basic knowledge in category theory.

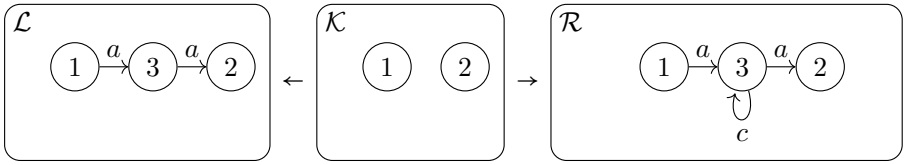
Finally, specializing to edge-labeled directed multigraphs enables us to leverage graph-specific properties to develop stronger termination criteria. Because termination is undecidable for general DPO graph rewriting systems [73], this specialization is essential for extending the boundaries of verifiable termination.

1.8 Contributions

Chapter 3 extends an existing method for termination of DPO graph rewriting systems, called type graph method [85, 19, 17, 39]. The method assigns weights to morphisms targeting a weighted type graph over a well-founded semiring. The weight of a graph is defined as the sum of the weights of all morphisms from that graph to the type graph. Relative termination of $\mathcal{A}$ with respect to $\mathcal{B}$ is proven by ensuring that rewriting steps using rules in $\mathcal{A}$ strictly decrease the weights of the host graphs, while rewriting steps using rules in $\mathcal{B}$ do not increase them. However, searching for a suitable type graph over a well-founded semiring is challenging. We propose to consider type graphs over non-well-founded semirings, which can be easier to find in some scenarios.

Chapter 4 introduces a new automated technique for relative termination of DPO graph rewriting with injective rules on edge-labeled multigraphs. The method is based on counting injective graph homomorphisms: let $\mathcal{A}$ and $\mathcal{B}$ be two sets of rewriting rules, if the number of injective homomorphisms from a chosen set of pattern graphs into the host graph strictly decreases when a rule from $\mathcal{A}$ is applied, and the number does not increase when any rule from $\mathcal{B}$ is applied, then rules in $\mathcal{A}$ can only be applied a finite number of times. This technique can handle some cases that prior interpretation-based approaches cannot. Proofs of some propositions, lemmas, and theorems of this chapter have been moved to Appendix 4.10 to improve readability.

Chapter 5 extends the technique introduced in Chapter 4 to handle rewriting systems with rules like the following one:



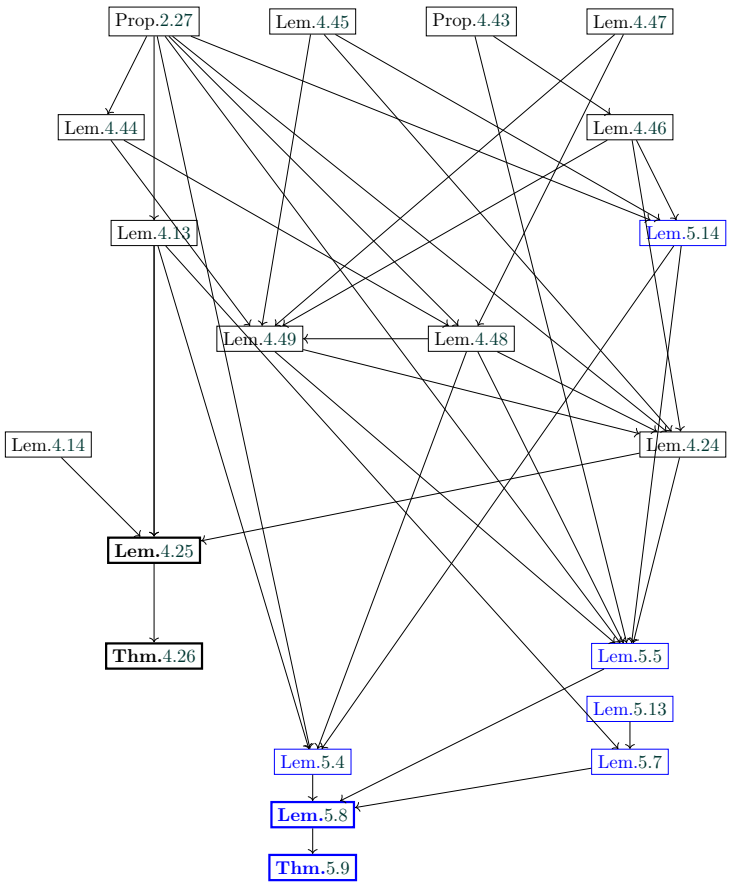
For any application of this rule that rewrites a host graph G to a result graph H, and for any choice of pattern graph X, the number of injective graph homo-

morphisms from X to G is less than or equal to the number of injective graph homomorphisms from X to H. Therefore, the method introduced in Chapter 4 cannot be applied. However, if we restrict our attention to occurrences of pattern graphs that are not part of a larger occurrence of a specific graph, the number of such occurrences decreases with each application of the rewriting rule. Proofs of some propositions, lemmas, and theorems of this chapter have been moved to Appendix 5.5 to improve readability.

Since a purely theoretical contribution would be of limited value without an accompanying software tool that implements it, Chapter 6 introduces the termination tool **LyonParallel** for DPO edge-labeled multigraph rewriting, which integrates the existing type graph method, our extension, and our morphism counting technique and its extension.

1.9 Dependency Graph

In the appendices of Chapter 4 and Chapter 5, and § 5.1, we present some propositions, lemmas, and theorems. The dependency structure among these results is clarified in the figure shown below, in which black nodes correspond to results in Chapter 4 while blue nodes correspond to results in Chapter 5. The four key results, Lemma 4.25, Theorem 4.26, Lemma 5.8 and Theorem 5.9, are highlighted in bold.



Chapter 2

Preliminaries

We recall definitions related to the double-pushout approach to edge-labeled multigraph rewriting and to relative termination.

2.1 Directed Edge-labeled Multigraph

We work with directed, edge-labeled graphs admitting parallel edges. This notion differs from that of Barr and Wells [5], whose directed multigraphs with loops lack edge labels; from that of König et al. [53], since we permit the collections of nodes and edges to be classes; and from that of Ehrig et al. [79], where both nodes and edges are labeled, whereas here only edges are labeled.

Other notions of graphs appear in the literature for distinct purposes, including hypergraphs [70], attributed graphs [35], and polygraphs [1]. Our choice reflects that a category in the sense of category theory (discussed in the next section) can be seen, in the style of Barr and Wells [5], as a directed edge-labeled multigraph endowed with extra structure, with object and morphism collections allowed to be classes.

Definition 2.1. *Let Σ be a finite set of labels. A **directed edge-labeled multigraph** is a structure $(V, E, \mathrm{dom}, \mathrm{cod}, \lambda)$ where*

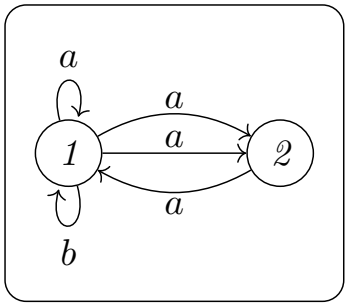
- *V is a collection of distinct elements called **nodes** (or **objects**),*
- *E is a collection of distinct elements, disjoint from V, called **edges** (or **arrows**),*
- *$\mathrm{dom} : E \to V$ is the **domain** (or **source**) function assigning to each edge its source node,*
- *$\mathrm{cod} : E \to V$ is the **codomain** (or **target**) function assigning to each edge its target node,*
- *$\lambda : E \to \Sigma$ is the **labeling** function assigning to each edge a label from Σ.*

*A directed edge-labeled multigraph is said to be **finite** if V and E are finite, and **discrete** if its edge set E is empty. For a directed edge-labeled multigraph G, we write $V(G)$ for its set of nodes, $E(G)$ for its set of edges, dom_G for its domain function, cod_G for its codomain function, and λ_G for its labeling function. $a : s \xrightarrow{l} t$ denotes an arrow a labeled by l from s to t.*

A directed edge-labeled multigraph may have multiple edges from one node to another, as well as loops (edges from a node to itself), because different edges u, v can have the same source and target nodes, i.e. $\mathrm{dom}(u) = \mathrm{dom}(v)$ and $\mathrm{cod}(u) = \mathrm{cod}(v)$.

Hereafter, a directed edge-labeled multigraph will be simply referred to as a "**labeled graph**" or "**graph**" when the context makes it clear, and we fix a set Σ of labels.

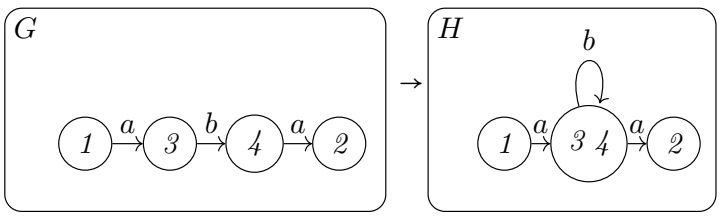
Example 2.2. *Consider the graph shown below. It has two nodes which are marked with numbers to facilitate discussion, but in general, nodes are not labeled. There are five edges: two loops on node* 1 *labeled* a *and* b*, two edges from node* 1 *to node* 2 *both labeled* a*, and one edge from node* 2 *to node* 1 *labeled* a*. Note that the two edges from node* 1 *to node* 2 *are distinct edges, even though they share the same source, target, and label.*



A homomorphism of labeled graphs is a homomorphism of unlabeled graphs that preserves the labels assigned to the edges.

Definition 2.3. *Let* G *and* H *be labeled graphs. A* ***homomorphism of labeled graphs*** $h : G \to H$ *is a pair of functions* $(h_V : V(G) \to V(H), h_E : E(G) \to E(H))$ *such that for every edge* $a : s \to t$ *in* G*, the edge* $h_E(a)$ *in* H *is from node* $h_V(s)$ *to node* $h_V(t)$*, and* $\lambda_H(h_E(a)) = \lambda_G(a)$*.*

Notation 2.4. *We use the notation from Overbeek and Endrullis [65, Notation 1] to visualize edge-labeled graph homomorphisms. Labeled graphs are enclosed in boxes with their names displayed in the top-left corner. Nodes and edges are assigned subsets of* $\mathbb{N}$ *as identifiers, and these identifiers are chosen such that: (i) Each node* y *(resp. edge* y*) in the codomain graph is assigned the union of the identifiers of all nodes (resp. edges) in the domain graph that are mapped to* y*; (ii) The graph homomorphism is uniquely determined by this assignment. To improve readability, we represent sets by listing their elements. Additionally, we omit identifiers when doing so does not cause confusion. An example of a homomorphism of labeled graphs is shown below. In this example, the sets* $\{1\}$*,* $\{2\}$*,* $\{3\}$*,* $\{4\}$*, and* $\{3, 4\}$ *are represented as* 1*,* 2*,* 3*,* 4*, and* 3 4*, respectively. Edge identifiers are omitted.*



Definition 2.5. *Let G, G' and G'' be labeled graphs, and $f : G \to G'$ and $g : G' \to G''$ homomorphisms of labeled graphs. Their composition, denoted by $g \circ f$, is defined as the component-wise composition of their node and edge mapping functions $(g_V \circ f_V, g_E \circ f_E)$.*

Definition 2.6. *An **unlabeled multigraph** G is a labeled graph such that the labeling function is constant, i.e., there is a label $l \in \Sigma$ such that $\lambda_G(e) = l$ for all edges e in G.*

Hereafter, we omit the labeling function when referring to an unlabeled graph, and $a : s \to t$ denotes an arrow a from s to t.

2.2 Pushout and pullback

Pushouts play a central role in the double-pushout approach to graph rewriting considered in this thesis. The concepts and notation in this section follow the treatments of Pierce [69] and Barr and Wells [5].

Definition 2.7. *A **category** is an unlabeled graph C together with a total function $u : V(C) \to E(C)$ and a partial function $\star : E(C) \times E(C) \to E(C)$ such that*

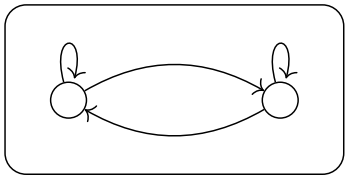
- *for all edges $f : X \to Y$ and $g : Y \to Z$, the edge $f \star g : X \to Z$ is defined;*
- *for every node X, $u(X)$ is an edge from X to X;*
- *for every $f : X \to Y$, we have $u(X) \star f = f = f \star u(Y)$;*
- *for all edges f, g and h, we have $(f \star g) \star h = f \star (g \star h)$ whenever either side is defined.*

*Edges are called **morphisms**. The function $\star$ is called **composition**. For all $X \in V(C)$, the edge $u(X)$ is denoted by id_X and is called the **identity** of the object X.*

Definition 2.8. *A category $\mathcal{C}$ is said to be **locally small** if for all objects X, Y in $\mathcal{C}$, the collection $\mathrm{Hom}(X, Y)$ of morphisms from X to Y is a set (called a **hom-set**) . For a locally small category, $\mathrm{Mono}(X, Y)$ denotes the set of all monomorphisms from X to Y.*

Throughout this section, fix a locally small category $\mathcal{C}$.

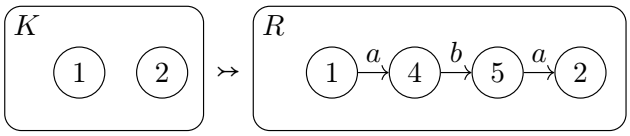
Example 2.9. *Consider the unlabeled graph shown below. It can be considered as a category where the objects are the nodes and the morphisms are the paths between nodes; composition is path concatenation. The identity of a node is the self-loop of the node. There are at least three morphisms from the left node to itself: the identity morphism (the self-loop), the path that traverses the self-loop twice, and the path that goes to the right node and back.*



Notation 2.10. *The composition of morphisms $f : X \to Y$ and $g : Y \to Z$ is written in diagrammatic order as $f \star g$, rather than in functional order $g \circ f$ as is common in litterature. The advantage is that, when reading from left to right, the morphisms appear in the same order as in the corresponding diagram, making the reasoning accompanying diagrams more intuitive.*

Definition 2.11. *A morphism $f : X \to Y$ is said to be a **monomorphism** (is **monic**) if given any morphisms $g, h : Z \to X$, $g \star f = h \star f$ implies $g = h$. In this case, we write $f : X \rightarrowtail Y$ to indicate that f is a monomorphism.*

When visualizing a monomorphism, we often use $\rightarrowtail$ instead of $\to$ to emphasize that it is monic. For example, a monomorphism of labeled graphs can be represented as follows:



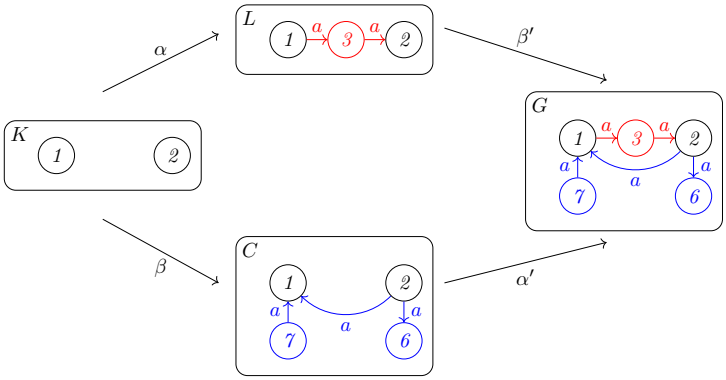
Example 2.12. *The category* **Set** *has sets as objects and total functions between them as morphisms. For $f : A \to B$ and $g : B \to C$, composition is given by $g \circ f$, and the identity morphism on a set A is the identity function id_A.*

Example 2.13. *Finite labeled graphs and their homomorphisms form a category, hereafter denoted by **Graph**. Its objects are labeled graphs, its morphisms are graph homomorphisms, and the monomorphisms are homomorphisms. **Graph** is locally small.*

A span (resp. cospan) is a couple of morphisms with a common domain (resp. codomain).

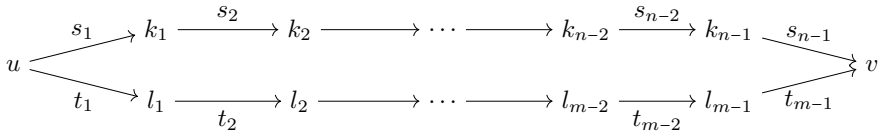
Definition 2.14. *An ordered pair $(\alpha : A \to B, \beta : A \to C)$ of morphisms with a common domain is called a **span** [58], denoted by $B \overset{\alpha}{\leftarrow} A \overset{\beta}{\to} C$. Likewise, an ordered pair $(\beta' : B \to D, \alpha' : C \to D)$ of morphisms with a common codomain is called a **cospan**, denoted by $B \overset{\beta'}{\to} D \overset{\alpha'}{\leftarrow} C$.*

Example 2.15. *Consider the diagram below in the category **Graph**, where the numbers inside nodes and the subgraphs in different colors illustrate how the morphisms map nodes and edges.*



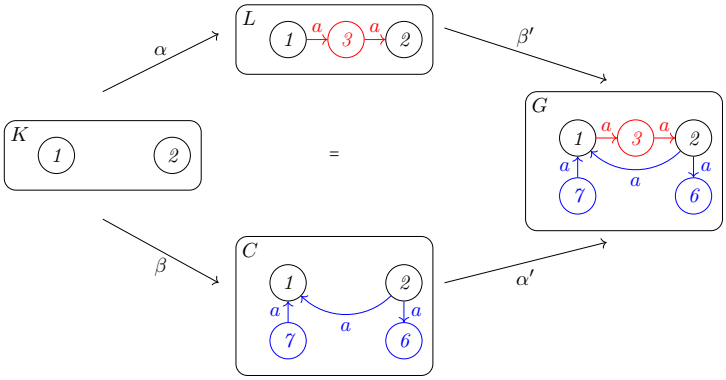
(α, β) *is a span, and* (β', α') *is a cospan.*

Definition 2.16 ([5])**.** *Let* $\mathcal{C}$ *be a category, and* G *an unlabeled graph. A* ***diagram*** *(of shape* G*) is a homomorphism of unlabeled graphs* $h : G \to \mathcal{C}$ *where* $\mathcal{C}$ *is considered as an unlabeled graph. A diagram is said to be* ***commutative*** *if, for all nodes* u*,* v*, and any two paths from* u *to* v *in the unlabeled graph* G*:*



the equality $h(s_1) \star h(s_2) \star \ldots \star h(s_{n-1}) = h(t_1) \star h(t_2) \star \ldots \star h(t_{m-1})$ *holds.*

Example 2.17. *A commutative diagram in the category* ***Graph*** *of finite, directed, edge-labeled multigraphs is illustrated below. The numbers inside nodes and the subgraphs in different colors illustrate how the morphisms map nodes and edges. The symbol = in the center of the diagram is used to indicate that the diagram is commutative, i.e. the two paths from node 1 to node 2 are equal.*

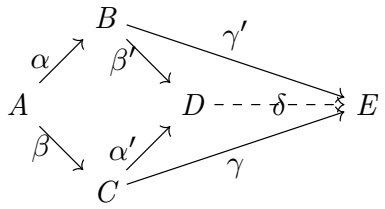


Notation 2.18. *When the context makes it clear,* h_{AB} *denotes a morphism* $h : A \to B$*, and we refer to diagrams by listing their nodes, as is standard in geometry.*

The pushout is a construction in category theory that can often be thought of as the construction of a new structure from two given structures by gluing them along a common interface structure.

Definition 2.19. *A* ***pushout*** *of a span* $B \overset{\alpha}{\leftarrow} A \overset{\beta}{\rightarrow} C$*, shown in the following diagram, is defined as a cospan* $B \overset{\beta'}{\rightarrow} D \overset{\alpha'}{\leftarrow} C$ *such that the following conditions hold:*

- $\alpha \star \beta' = \beta \star \alpha'$,
- *for every cospan* $B \overset{\gamma'}{\rightarrow} E \overset{\gamma}{\leftarrow} C$*, if* $\alpha \star \gamma' = \beta \star \gamma$ *holds, then there is a unique morphism* $\delta : D \to E$ *such that* $\gamma' = \beta' \star \delta$ *and* $\gamma = \alpha' \star \delta$.



The diagram involving $(\alpha, \beta, \alpha', \beta')$ *is called the* ***pushout square****, with* D *as the* ***pushout object****. The existence of a unique morphism is known as the* ***Universal mapping property of the pushout****.*

Example 2.20. *Pushouts of two morphisms always exist in* **Set***, and (up to isomorphism) can be described as follows. Let* $B \xleftarrow{\alpha} A \xrightarrow{\beta} C$ *be a span. Its pushout is the cospan* $B \xrightarrow{\beta'} D \xleftarrow{\alpha'} C$ *where the pushout object of* (α, β) *is the set*

$$D = (B + C)/\sim$$

where $B + C$ *denotes the disjoint union of* B *and* C *and* $\sim$ *is the smallest equivalence relation that includes* $\{(\alpha(a), \beta(b)) \mid a \in A\}$*. The maps* $\beta' : B \to D$ *and* $\alpha' : C \to D$ *send each element to its equivalence class.*

For example, consider the functions α *and* β *in the category* **Set** *in the diagram, illustrated in Figure 2.1. In this diagram, each set is drawn as a box and its elements are represented by circles. The numbers inside circles indicate how the functions map those elements.*

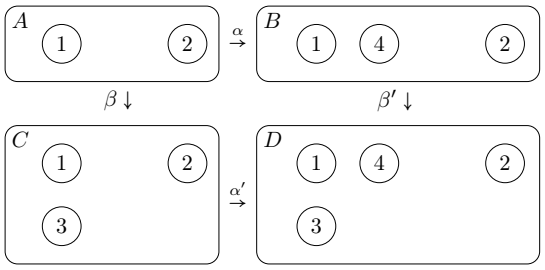


Figure 2.1

Throughout this example, an element labeled n *in a set* X *is denoted by* n_X *to avoid ambiguity. The binary relation* $\sim$ *is the reflexive, symmetric and transitive closure of the binary relation* $\{(1_B, 1_C), (2_B, 2_C)\}$*.*

The disjoint union of B *and* C *is*

$$D' = \{1_B, 2_B, 4_B, 1_C, 2_C, 3_C\},$$

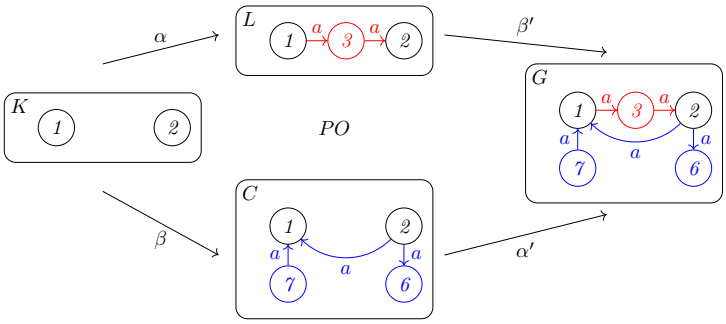
and the quotient $D'/\sim$ *is*

$$D'/\sim = \{[1_B], [2_B], [3_C], [4_B]\},$$

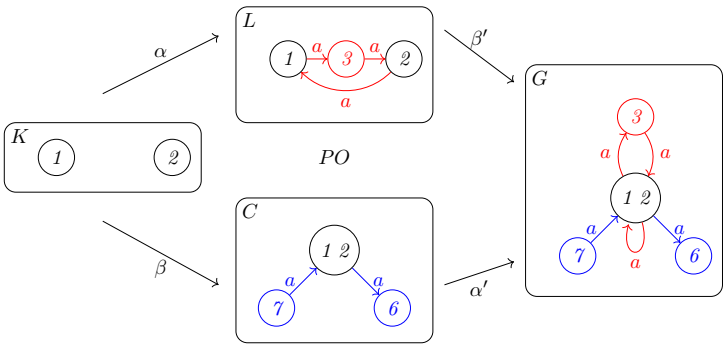
We have $[1_C] = [1_B]$ *and* $[2_C] = [2_B]$*, and the maps* $\beta'' : B \to D'$ *and* $\alpha'' : C \to D'$ *send each element to its equivalence class. Note that* $\{[1_B], [2_B], [3_C], [4_B]\}$ *is isomorphic to* D*, shown in Figure 2.1, which is expected because the pushout of a span is unique up to isomorphism.*

Proposition 2.21. *[30, p.188] In category **Graph**, the pushout of two arrows always exists: It can be computed componentwise (as a pushout in **Set**) for the nodes and for the edges, and the source, target, and labeling mappings are uniquely determined.*

Example 2.22. *The diagram in the category **Graph** shown below is a pushout square. The numbers inside nodes and the subgraphs in different colors illustrate how the morphisms map nodes and edges. In this example, both α and β are injective morphisms. Therefore, the pushout object G can be constructed easily by taking the interface graph K and adding elements from L and C which are not present in K.*



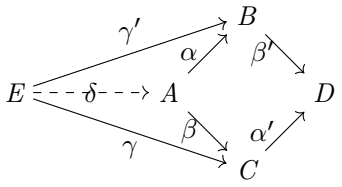
Example 2.23. *Consider the diagram in the category **Graph** shown below, where the numbers inside nodes and the subgraphs in different colors illustrate how the morphisms map nodes and edges. In this example, β is not injective. Therefore, some elements are merged in the pushout object G.*



The *pullback* is the dual construction of the pushout, and it can be thought of as construction of the interface structure along which two structures are glued together.

Definition 2.24. *A **pullback** of a cospan $B \xrightarrow{\beta'} D \xleftarrow{\alpha'} C$ is a span $B \xleftarrow{\alpha} A \xrightarrow{\beta} C$ such that the following conditions hold:*

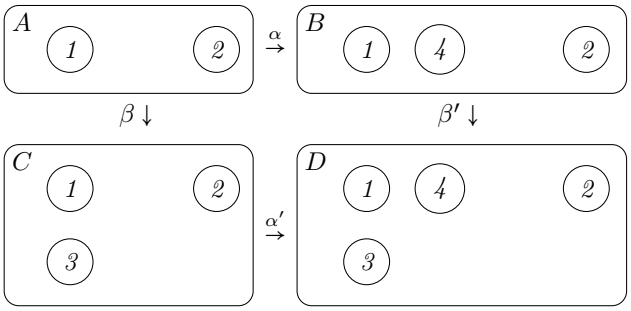
- $\alpha \star \beta' = \beta \star \alpha'$,
- *for every span $B \xleftarrow{\gamma'} E \xrightarrow{\gamma} C$ if $\gamma' \star \beta' = \gamma \star \alpha'$ holds, then there is a unique morphism $\delta : E \to A$ such that $\gamma' = \delta \star \alpha$ and $\gamma = \delta \star \beta$.*



The diagram involving $(\alpha, \beta, \alpha', \beta')$ *is called the* ***pullback square****, with* A *as the* ***pullback object****. The existence of a unique morphism is known as the* ***universal mapping property of the pullback****.*

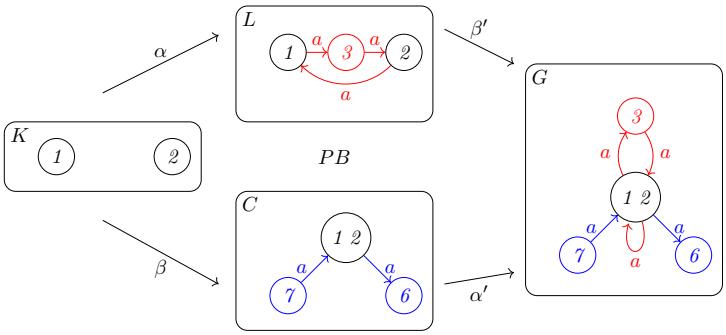
In the category **Graph** pullbacks always exist and are computed pointwise: take the pullback in **Set** of the node sets and of the edge sets, and equip the resulting graph with source, target and labeling maps induced componentwise; the projection graph homomorphisms are the corresponding componentwise projections and they satisfy the universal property.

Example 2.25. *In the category* ***Set****, the pullback of a cospan* $B \overset{\beta'}{\to} D \overset{\alpha'}{\leftarrow} C$ *is the span* $B \overset{\alpha}{\leftarrow} A \overset{\beta}{\to} C$ *where the pullback object* A *is the set* $A = \{(b, c) \mid \beta(c) = \alpha(b)\} \subseteq B \times C$*;* α *and* β *are defined as the corresponding projections, e.g.,* $\alpha((b, c)) = b$ *and* $\beta((b, c)) = c$*. Consider the following diagram in the category* ***Set****, where sets are drawn as boxes, circles represent elements of sets, and numbers inside circles indicate how the functions map those elements.*



The span $B \overset{\alpha}{\leftarrow} A \overset{\beta}{\to} C$ *is the pullback of the cospan* $B \overset{\beta'}{\to} D \overset{\alpha'}{\leftarrow} C$*. Indeed, the pullback object can be taken as the set* $A' = \{(1_B, 1_C), (2_B, 2_C)\} \subseteq B \times C$*, and the morphisms* $\alpha'' : A' \to B$ *and* $\beta'' : A' \to C$ *are defined as the corresponding projections, e.g.,* $\alpha''((x, y)) = x$ *and* $\beta''((x, y)) = y$*. Note that* A *is isomorphic to* A'*, which is expected because the pullback of a cospan is unique up to isomorphism.*

Example 2.26. *Consider the diagram in the category* ***Graph****, shown below. The numbers inside nodes and the subgraphs in different colors illustrate how the morphisms map nodes and edges. The span* $L \overset{\alpha}{\leftarrow} K \overset{\beta}{\to} C$ *is a pullback of the cospan* $L \overset{\beta'}{\to} G \overset{\alpha'}{\leftarrow} C$*.*



An interesting phenomenon about pushouts and pullbacks in the category **Graph** is stated in the following proposition.

Proposition 2.27 ([54, Lemma 13])**.** *In the category* ***Graph****, pushouts along monomorphisms are also pullbacks.*

This proposition is illustrated in Example 2.23 and Example 2.26.

2.3 Double-Pushout Rewriting

The algebraic approach of double-pushout (DPO) rewriting was introduced by Ehrig et al. [37]. It is not restricted to graphs, and can be defined in any category enjoying certain properties [35, 54]. This section recalls the definition of DPO rewriting, following the treatment of Endrullis and Overbeek [39].

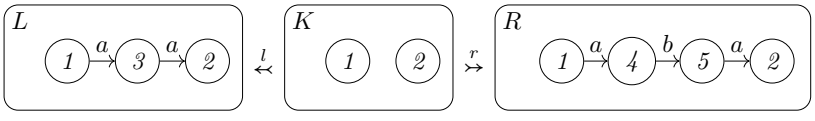
Throughout this section, a category $\mathcal{C}$. A DPO rewriting rule is given by two morphisms $l : K \to L$ and $r : K \to R$. L encodes the rule's preconditions, R specifies the postconditions, and K is the interface—the subobject common to L and R that is preserved by the rule, and links L and R to the surrounding context.

Definition 2.28. *A* ***DPO rewriting rule*** ρ *is a span* $L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R$*, where* K *is the* ***interface****,* L *is the* ***left-hand-side object****, denoted by* $\mathrm{lhs}(\rho)$*, and* R *is the* ***right-hand-side object****, denoted by* $\mathrm{rhs}(\rho)$*. The rule* $(R \overset{r}{\leftarrow} K \overset{l}{\rightarrow} L)$ *is denoted by* ρ^{-1}*.*

The rule is said to be ***left monic*** *if* l *is monic, and* ***right monic*** *if* r *is monic; it is said to be* ***monic*** *if it is both left monic and right monic. A* ***match*** *of the rule in an object* G *is a morphism* $m : L \to G$*.*

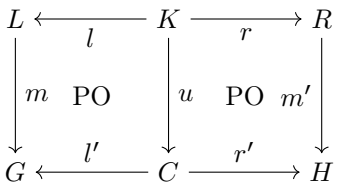
In the case of the category **Graph**, the interface K of a rule $L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R$ is said to be discrete if K is discrete. The words "injective" and "monic" are used interchangeably in this context.

Example 2.29. *Consider the following DPO rewriting rule.*



It is an injective rule, and it replaces an occurrence of the graph $\circ \xrightarrow{a} \circ \xrightarrow{a} \circ$ *with an occurrence of the graph* $\circ \xrightarrow{a} \circ \xrightarrow{b} \circ \xrightarrow{a} \circ$*, keeping the extreme nodes unchanged.*

Definition 2.30. *A diagram in a category consisting of two pushout squares arranged as shown below is called a* ***double-pushout (DPO) diagram****.*

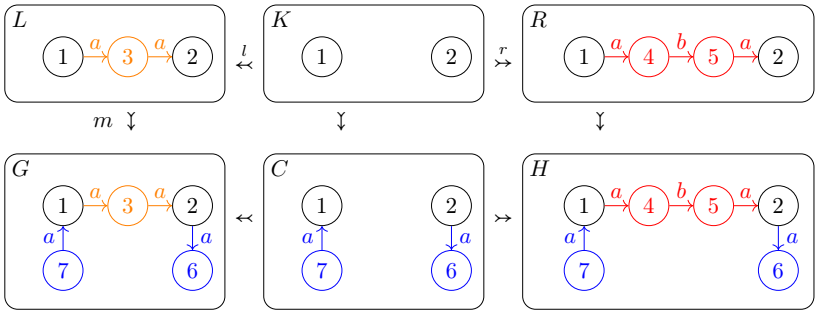


To perform a DPO rewriting step with a rule $\rho : L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R$ on an object G, choose a match $m : L \to G$. If the double-pushout diagram of Definition 2.30 exists (for l, r, and m), it yields an object H; we then say that G rewrites to H via ρ at m. The diagram with double pushout is the origin of the name "double-pushout rewriting".

The diagram with double pushout in Definition 2.30 is said to be a **witness** for the **rewriting step** from G to H using a rule $\rho = (L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$ and a **match** m, denoted $G \Rightarrow^{m}_{\rho} H$ or $G \Rightarrow^{\delta}_{\rho} H$. The pushout squares $KLGC$ and $KRHC$ are denoted by left(δ) and right(δ), respectively.

In the category **Graph**, to apply a DPO graph rewriting rule $\mathcal{L} \overset{l}{\leftarrow} \mathcal{K} \overset{r}{\rightarrow} \mathcal{R}$ to a host graph $\mathcal{G}$, one must first identify an occurrence of $\mathcal{L}$ in which a subgraph is designed as the interface graph. The other elements of the host graph $\mathcal{G}$ together with the interface graph form the context graph $\mathcal{C}$ of the rewriting step. The graph $\mathcal{G}$ is thus decomposed into the occurrence of $\mathcal{L}$ and $\mathcal{C}$. Then, one modifies the occurrence of $\mathcal{L}$ (by removing some nodes and edges and adding some fresh nodes and edges while keeping the interface graph unchanged, and identifying nodes in the interface graph) to obtain an occurrence of $\mathcal{R}$. Finally, the result graph is obtained by gluing the occurrence of $\mathcal{R}$ with $\mathcal{C}$ via the interface graph $\mathcal{K}$.

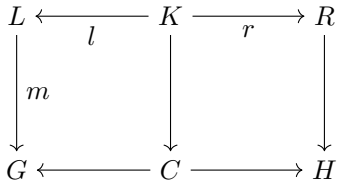
As an example, consider the DPO rewriting rule in Example 2.29, and the DPO diagram in the following figure. The numbers inside nodes and the subgraphs in different colors illustrate how the morphisms map nodes and edges.



The graph G can be rewritten to yield the graph H by applying the rule to the occurrence $(1) \xrightarrow{a} (3) \xrightarrow{a} (2)$ of L in G. Specifically, G can be decomposed into an occurrence of L (elements shown in orange and black in G) and the context C (elements shown in blue and black in G). The interface is the subgraph shown in black. The graph R can be decomposed into an occurrence of K (elements shown in black) and the others (elements shown in red). The rule removes the

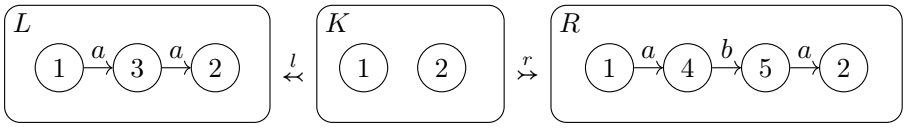
orange part of the occurrence of L in G to obtain the context C. The result graph H is obtained by gluing C and R via the interface graph.

In category **Graph**, the pushout of two arrows always exists by Proposition 2.21. Therefore, consider the DPO diagram shown below:

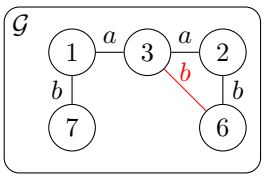


Once the left pushout square is constructed, the right pushout square can always be constructed and is unique up to isomorphism because of the universal property.

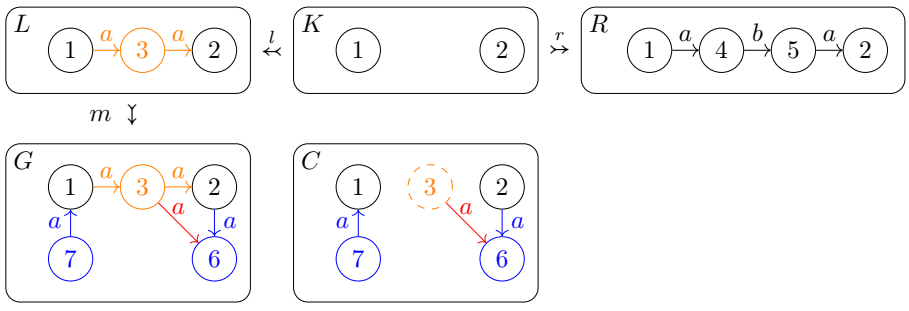
However, given a match $L \overset{m}{\to} G$, the left pushout square cannot always be constructed. For example, consider the same rewriting rule:



and the graph G shown below:



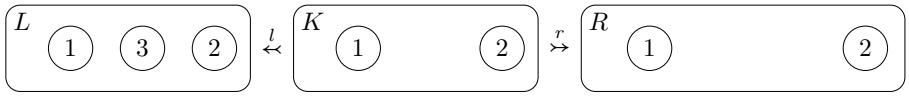
The left pushout square cannot be constructed as visualized by the following figure, where the numbers inside nodes and the subgraphs in different colors illustrate how the morphisms map nodes and edges; the dashed circle in the box C indicates that the node 3 is missing.



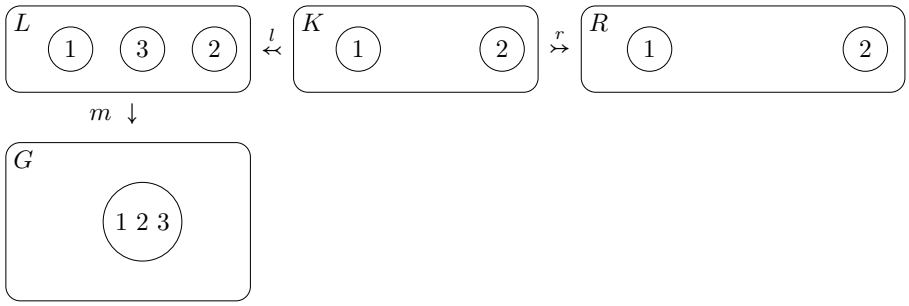
Specifically, the graph G has the occurrence $1 \overset{a}{\to} 3 \overset{a}{\to} 2$ of L. Nevertheless, since node 3 does not exist in C, the edge from node 3 to node 6 becomes dangling, making the construction of C impossible and preventing the rule application on that occurrence of L. Concretely, nodes in $m(L) \setminus (l \star m)(K)$ are removed when the rule is applied, while edges in $G \setminus m(L)$ remain, which can

produce dangling edges. Therefore, no node of G outside the image of m should be incident to a node of $m(L)$ outside the image of $l \star m$. This requirement is precisely the dangling-edge condition in the following Proposition 2.31.

There is another requirement for the existence of a pushout complement. To give an intuition, consider the rewriting rule in **Graph**, shown below.



This rule deletes a node of the graph, and the match identifies the three nodes of L with a single node of G. Consider the match $L \overset{m}{\to} G$ shown in the following diagram:



Since there are no edges, we may view these graphs as sets of nodes and compute the left pushout square explicitly, as in Example 2.20. Suppose that the left pushout square exists—i.e., there is a graph C and morphisms $u : K \to C$ and $l' : C \to G$ such that the following diagram is a pushout square:

$$\begin{array}{ccc} L & \xleftarrow{\quad l \quad} & K \\ \downarrow{\scriptstyle m} & & \downarrow{\scriptstyle u} \\ G & \xleftarrow{\quad l' \quad} & C \end{array}$$

Then the set of nodes of G is isomorphic to the quotient $D/\sim$, where D is the disjoint union of the set $V(L)$ of nodes of L and the set $V(C)$ of nodes in C and $\sim$ is the equivalence relation generated by $\{(l(v), u(v)) \mid v \in V(K)\}$.

In this example, there are two cases, depending on how the nodes of K map into C. Throughout this paragraph, a node v of a graph X is denoted by v_X to avoid ambiguity. If nodes in K are mapped to different nodes 1_C and 2_C in C, then $\sim$ is the smallest equivalence relation generated by $\{(1_L, 1_C), (2_L, 2_C)\}$ and $D = \{1_L, 2_L, 3_L, 1_C, 2_C, \cdots\}$. If nodes in K are mapped to the same node 1_C in C, then the equivalence relation $\sim$ is the smallest equivalence relation including $\{(1_L, 1_C), (2_L, 1_C)\}$ and $D = \{1_L, 2_L, 3_L, 1_C, \cdots\}$. In both cases $3_L \notin [1_L]_\sim$, and G cannot be isomorphic to the quotient set $D/\sim$, which contains at least two distinct elements. The condition that prevents this situation is the identification condition in Proposition 2.31.

This leads to the following proposition summarizing the conditions under which pushout complements exist.

Proposition 2.31 ([30]). *Let $K \xrightarrow{l} L$ and $L \xrightarrow{m} G$ be two morphisms in the category* ***Graph****. There are two morphisms $K \xrightarrow{u} C \xrightarrow{l'} G$ such that $K \xrightarrow{l} L \xrightarrow{m} G$ and $K \xrightarrow{u} L \xrightarrow{l'} G$ form a pushout square, as shown below, if and only if the following conditions are satisfied:*

- *Dangling edge condition: No edge in $G \setminus m(L)$ is incident to any node in $m(L) \setminus (l \star m)(K)$;*
- *Identification condition: There is no $x, y \in L_V \cup L_E$ such that $x \neq y$, $m(x) = m(y)$ and $y \notin l(K_V \cup K_E)$.*

$$\begin{array}{ccc} L & \xleftarrow{\quad l \quad} & K \\ {\scriptstyle m}\downarrow & \text{PO} & \downarrow{\scriptstyle u} \\ G & \xleftarrow{\quad l' \quad} & C \end{array}$$

There are several variants of DPO rewriting. Typically both the match m and the morphism l are required to be monomorphisms, and some variants impose additional constraints on the left pushout—for example, that the pushout complement be minimal or initial [16, 6, 7]. Because these choices affect which rewriting steps are permitted and hence properties such as termination, following Endrullis and Overbeek [39], we use the definition which is parametric in the variation of DPO under consideration.

Definition 2.32 ([39]). *A* ***DPO rewriting framework*** *$\mathfrak{F}$ is a mapping of DPO rewriting rules to classes of DPO diagrams. Specifically, for every rule $\rho = (L \xleftarrow{l} K \xrightarrow{r} R)$, class $\mathfrak{F}(\rho)$ consists of DPO diagrams shown below, which are witnesses of rewriting steps using the rule ρ.*

$$\begin{array}{ccccc} L & \xleftarrow{\quad l \quad} & K & \xrightarrow{\quad r \quad} & R \\ {\scriptstyle m}\downarrow & & \downarrow & & \downarrow \\ G & \xleftarrow{\qquad} & C & \xrightarrow{\qquad} & H \end{array}$$

The ***DPO rewriting relation $\Rightarrow_{\rho,\mathfrak{F}}$ induced by a DPO rewriting rule ρ in $\mathfrak{F}$*** *is defined as follows:*

$$G \underset{\rho,\mathfrak{F}}{\Rightarrow} H \text{ if and only if } G \overset{\delta}{\underset{\rho}{\Rightarrow}} H$$

for some $\delta \in \mathfrak{F}(\rho)$.

The ***DPO rewriting relation $\Rightarrow_{\mathcal{R},\mathfrak{F}}$ induced by a set $\mathcal{R}$ of DPO rewriting rules in $\mathfrak{F}$*** *is given by:*

$$G \underset{\mathcal{R},\mathfrak{F}}{\Rightarrow} H \text{ if and only if } G \underset{\rho,\mathfrak{F}}{\Rightarrow} H$$

for some $\rho \in \mathcal{R}$.

Whenever $\mathfrak{F}$ is clear from the context, we omit $\mathfrak{F}$ and write $\Rightarrow_\rho$ and $\Rightarrow_{\mathcal{R}}$.

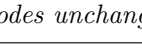
Hereafter, $\mathfrak{M}$ denotes the DPO rewriting framework that associates each rule with the class of all DPO diagrams of the form shown in Definition 2.32 with monic match m.

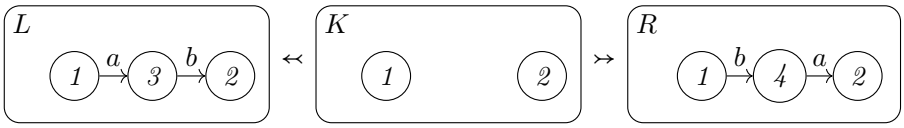
2.4 Relative Termination of DPO rewriting relation

Given a DPO rewriting framework $\mathfrak{F}$, a rule set $\mathcal{R}$ defines the binary relation "object X can be rewritten to object Y using rules from the rule set" on the objects of the category $\mathcal{C}$.

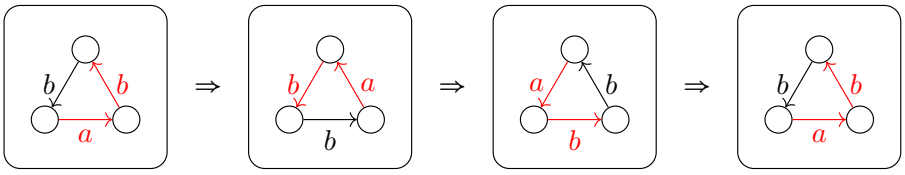
Definition 2.33. *Let $\mathcal{R}$ be a rule set and let $\mathfrak{F}$ be a DPO rewriting framework. An $(\mathcal{R}, \mathfrak{F})$-**rewriting chain** is a $\Rightarrow_{\mathcal{R},\mathfrak{F}}$-chain.*

*When the context is clear, we simply call it a **rewriting chain**.*

Example 2.34. *Consider the rewriting rule below. It replaces an occurrence of the graph*  *with an occurrence of the graph* *, keeping the extreme nodes unchanged.*

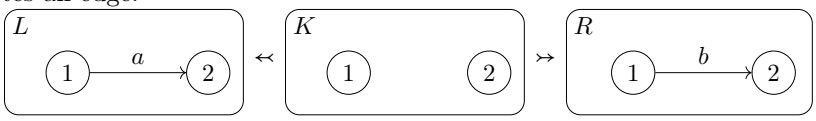


A looping rewriting chain using this rule can be the following, to be read left to right. In each graph, the subgraph to be replaced to obtain the next graph is highlighted in red.



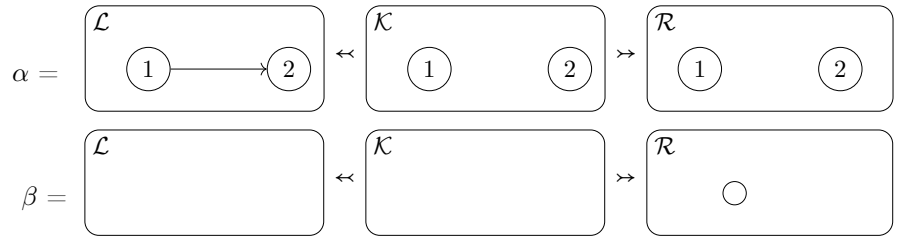
For a set of rewriting rules $\mathcal{R}$ (and a DPO rewriting framework $\mathfrak{F}$), the impossibility of transforming any object indefinitely with the non-deterministic strategy "apply rules as long as possible" using rules from $\mathcal{R}$ (in the framework $\mathfrak{F}$) is called *termination* [62]. This property corresponds to program termination on all inputs in conventional programming languages, and is undecidable in general [73].

For example of a non-terminating rule set, consider the set consisting of the rule and the infinite rewriting chain in Example 2.34. For example of a terminating rule set, consider the set with the unique rule, shown below, which deletes an edge.



Any rewriting chain using this rule is finite, because each rewriting step decreases the number of edges by one and a graph has only a finite number of edges by definition.

However, in many cases, an interesting property can be proved even if the whole rule set is not terminating. Take the system with rules α and β shown below. Rule α removes one edge per application; rule β introduces a new node.



Because β can always be re-applied to generate additional nodes, infinite rewriting sequences exist. Conversely, α is applicable only a finite number of times in any rewriting chain, since each application removes an edge and no rule adds edges. Since the initial graph is finite, the possibility of non-termination stems exclusively from β. This attribution illustrates the broader idea of *relative termination* as introduced by Klop [51] and explored in [41, 50, 39, 85, 19, 17].

Definition 2.35. *Let A be a collection of objects and let R and S be binary relations on A. We say that R is **terminating relative to** S (or that R **terminates relative to** S) if any $(R \cup S)$-chain contains only a finite number of R-steps. In particular, R is **terminating** (or terminates) if it is terminating relative to the empty relation $\varnothing$.*

For example, the relation $>$ on $\mathbb{N}$ is terminating relative to the empty relation, since any $(> \cup \varnothing)$-chain contains only a finite number of $>$-steps. The relation $>$ is also terminating relative to $\geq$ on $\mathbb{N}$, since any $(> \cup \geq)$-chain can only contain a finite number of $>$-steps.

Relative termination and termination carries over to rewriting systems in a straightforward way via their associated rewriting relations.

Definition 2.36. *Let $\mathcal{R}$ and $\mathcal{S}$ be sets of rewriting rules and let $\mathfrak{F}$ be a DPO rewriting framework. We say that $\mathcal{R}$ is **terminating relative to** $\mathcal{S}$ if $\Rightarrow_{\mathcal{R},\mathfrak{F}}$ is terminating relative to $\Rightarrow_{\mathcal{S},\mathfrak{F}}$.*

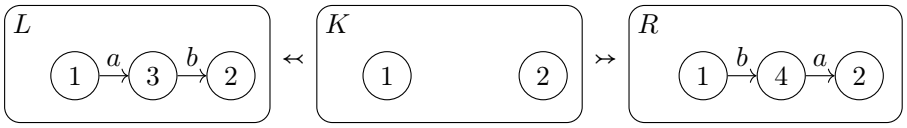
In practice, to prove termination of a rewriting system $\mathcal{R}$, one partitions the set of rules into two disjoint subsets $\mathcal{B}$ and $\mathcal{A}$ with non-empty $\mathcal{A}$ such that $\mathcal{A}$ terminates relative to $\mathcal{B}$. If $\mathcal{B}$ is empty then the termination of $\mathcal{R}$ is established, otherwise, a new iteration starts with the strictly smaller rule set $\mathcal{B}$.

Many techniques have been developed for proving termination of term rewriting systems [3, 4, 24, 26, 31, 42, 62, 82, 60, 80]. Some of them can be adapted to prove relative termination, for example the polynomial interpretation method. However, most of these techniques cannot be directly applied to graph rewriting systems.

One reason is that techniques for term rewriting systems often rely on the fact that terms are tree-like structures, while graphs are not. In the polynomial interpretation method, every n-ary function symbol f is associated with a polynomial $P_f(X_1, X_2, \ldots, X_n)$ with coefficients in $\mathbb{N}$, so that ground terms are mapped to natural numbers. The system is guaranteed to terminate if, for every rewriting rule $l \to r$, the polynomial interpretation of l is strictly greater than

that of r. For instance, consider the term rewriting rule $f(a, b) \to f(b, a)$. One could interpret the constants a and b as 2 and 1, respectively, and the function symbol f with the polynomial $P_f(X_1, X_2) = 2X_1 + X_2$. Under this interpretation, $f(a, b)$ has value 5 and $f(b, a)$ has value 4, demonstrating strict decrease under this rule.

A further complication stems from the presence of cycles in graphs, in contrast to terms, which are inherently acyclic. These cycles may allow graphs to be rewritten indefinitely, making standard termination arguments that rely on acyclicity ineffective. For example, the term rewriting rule $f(a, b) \to f(b, a)$ is analogous to the graph rewriting rule in Example 2.34:



However, the term rewriting rule terminates, while the graph rewriting rule does not as shown in Example 2.34.

Therefore, termination techniques specific to graph rewriting systems are needed. Below, we provide an overview of some existing techniques for proving (relative) termination of DPO rewriting systems.

The weighted type graph method was first introduced by Zantema, König and Bruggink (2014) [85] for proving relative termination of cycle rewriting. Their subsequent work in 2014 [19] generalized it for DPO rewriting on edge-labeled directed multigraphs with injective rules and injective matches; later, it was extended to general DPO rewriting on edge-labeled multigraphs by Bruggink et al. (2015) [17]; and it was adapted to broader categories and DPO variants by Endrullis and Overbeek (2024) [39]. It assigns a weight to each graph by aggregating the weights of all morphisms to a designated weighted type graph. The challenge consists in characterizing the type graphs and the computation of weights that guarantee a decrease in the weight of the graphs with each rule application. An extension of the weighted type graph method which facilitates searching for suitable type graphs will be presented in Chapter 3.

The subgraph-counting method was developed by Overbeek and Endrullis (2024) [66]. It is designed for proving relative termination of PBPO+ (2023) [67, 65]—a rewriting formalism capable of simulating left-injective DPO rewriting. This method assigns a weight to a graph by summing the weights of morphisms from a set of designated graphs into the target graph. This method, Additionally, this method can be applied for many different graph notions. An incomparable but related approach based on counting morphisms will be presented in Chapter 4.

Plump (1995) [72] gives a necessary and sufficient termination criterion for left-injective DPO rewriting rules in terms of forward closure which are derivations satisfying certain property. A rewriting system is terminating if and only if it does not admit an infinite forward closure and for each rule $L \overset{l}{\leftarrow} K \overset{r}{\to} R$, l is not surjective. However, absence of infinite forward closures is undecidable, since this criterion is equivalent to termination of systems with non-surjective left morphisms.

Plump (2018) [71] later proposed a modular critical pair-based strategy for left-injective DPO graph rewriting with monic matches. It decomposes a rewrit-

ing system into two subsystems, each of which can be analyzed separately using arbitrary termination techniques, and if both subsystems terminate, then the termination of the whole system is guaranteed. However, termination of the subsystems still needs to be established using some techniques.

Levendovszky et al. (2007) [56] proposed a termination criterion for DPO rewriting with monic matches, injective rules and negative application conditions on finite typed attributed graphs. It is based on the fact that if the application of every infinite sequence of rules requires the initial graph to be infinite, then the system terminates. This technique is theoretically very interesting, but it is hard to check the termination condition automatically as explained in [56, §6].

Bottoni et al. (2005) [14] present a termination criterion for DPO rewriting systems with very restrictive external control mechanisms. The method relies on a measuring function satisfying a very strong constraint, and the instance of such a measuring function proposed is node and edge counting which is subsumed by the subgraph-counting method and the weighted type graph method.

Bottoni et al. (2010) [15] present a criterion for termination of DPO rewriting with monic matches, injective rules and negative application conditions, based on the construction of a labeled transition system.

Chapter 3

Termination of Graph Rewriting using Weighted Type Graphs over Non-Well-Founded Semirings

We extend an existing method for termination of DPO graph rewriting systems, called weighted type graph method. [1]

The method employs a finite graph T, called a weighted type graph, to assign measures from a well-founded semiring to graphs subject to rewriting. Concretely, a graph G is interpreted via the set $\mathsf{Hom}(G,T)$ of all morphisms from G to T. Each morphism $h \in \mathsf{Hom}(G,T)$ is assigned a weight, and the weight of G is then obtained by aggregating the weights of all morphisms in $\mathsf{Hom}(G,T)$. Relative termination of $\mathcal{A}$ with respect to $\mathcal{B}$ is proved by ensuring that rewriting steps using rules in $\mathcal{A}$ strictly decrease the weight of the host graph, while rewriting steps using rules in $\mathcal{B}$ do not increase it. The name of the method stems from the fact that each graph can be viewed as having $\mathsf{Hom}(G,T)$ as its type.

Previous work [85, 19, 17] proposed three concrete semirings on natural numbers: the natural tropical semiring $\mathfrak{T}$, the natural arctic semiring $\mathfrak{A}$, and the natural arithmetic semiring $\mathfrak{N}$. However, constructing weighted type graphs over these concrete semirings for DPO rewriting on edge-labeled directed multigraphs is difficult in general, because it requires quantifying over all edge-labeled directed multigraphs with weighted edges over $\mathbb{N}$.

In response to this challenge, previous work restricts the search to weighted type graphs with a fixed number of nodes, where each edge carries a weight bounded by a fixed maximum value. Concretely, one fixes a number of nodes k and a maximum edge weight, and then searches for a candidate weighted type graph T within these bounds. For a fixed finite label alphabet Σ, such a candidate T is determined by choices for each potential labeled edge between pairs of nodes: whether the edge is present and, if present, which weight it carries. Despite these restrictions, searching for suitable weighted type graphs

[1] The result presented in this chapter led to a workshop paper [77].

remains challenging.

Let $n = k^2 \cdot |\Sigma|$. For a weighted type graph over the natural tropical semiring $\mathfrak{T}$ or the natural arctic semiring $\mathfrak{A}$, the problem amounts to checking the satisfiability of an existential Presburger arithmetic formula with n binary variables and n integer variables. While modern SMT solvers, such as Z3 [63] (which incorporates the dedicated CutSat solver [47]), can solve practical instances of this problem, its worst-case complexity remains exponential $O(2^{2^n})$ [13].

For a weighted type graph over the natural arithmetic semiring $\mathfrak{N}$, the task amounts to checking the satisfiability of an existential Peano arithmetic formula with addition and multiplication, involving n binary variables and n integer variables. Though modern solvers like Z3 can tackle practical instances, it is a semi-decidable problem [61].

There are automated tools that implement the weighted type graph method for proving termination of DPO rewriting systems on edge-labeled directed multigraphs: `TORPAcyc` [84] implements the weighted type graph method for cycle rewriting and uses `Yices` [32] to solve constraint systems; `Grez` [20] implements the weighted type graph method for DPO rewriting on edge-labeled directed multigraphs; `GraphTT-wtg` [2] [40] implements the weighted type graph method for DPO rewriting on many categories. `Grez` and `GraphTT-wtg` use `Z3` to solve constraint systems.

In spite of the theoretical power of the weighted type graph method, its practical applicability is limited by by the difficulty of guessing, a priori, suitable values for the number of nodes and maximum edge weight, and by the high computational complexity of searching for suitable weighted type graphs. Our experiments with the two publicly available ones `TORPAcyc` and `Grez` show that when the number of edge labels is 2, they struggle to search for weighted type graphs with 4 nodes and maximum edge weight of 2, or to search for weighted type graphs with 3 nodes when the maximum edge weight is larger than 3. [3]

To address these challenges, we extend the weighted type graph method to non-well-founded semirings and introduce three concrete semirings over $\mathbb{R}$: the real tropical semiring $\mathfrak{T}'$, the real arctic semiring $\mathfrak{A}'$, and the real arithmetic semiring $\mathfrak{N}'$. The idea of using weights from non-well-founded domains to prove termination was first used in the context of term rewriting by Lucas [59].

Working over $\mathbb{R}$ is primarily a practical device that simplifies the constraint-solving task during the search for a suitable type graph. The termination argument itself relies on a uniform strict decrease condition: we require graph weight to be elements in $\mathbb{R}^+$, and the existence of a strictly positive constant δ such that every rewrite step decreases the weight by at least δ. This assumption discretizes the real-valued measure. Consequently, the contribution of this chapter is not that termination becomes provable because the weights are real, but that using real-valued weights can make it easier to find suitable weights in practice.

Compared with the integer-weighted setting, our approach turns the edge-weight variables—accounting for half of the decision variables—into real-valued variables. This has two advantages:

1. **Theoretical complexity and practical performance.** For the arith-

[2] At the time of writing and to the best of our knowledge, it is not publicly available.

[3] Experiments were conducted on a laptop equipped with an i5-1038NG7 CPU, which features 4 cores, a base clock speed of 2 GHz, a boost speed of 3.8 GHz, and 16 GB RAM.

metic semiring, integer weights lead to constraint systems involving multiplication of variables, corresponding to fragments of Peano arithmetic that are undecidable in general, whereas using real weights yields constraints in first-order real arithmetic, which is decidable (e.g. by Tarski's decision procedure [81]). For the tropical and arctic semirings, the constraints do not involve multiplication of variables; with integer weights they fall into existential Presburger arithmetic, whose complexity is at least exponential, while with real weights they become, after fixing the discrete choices, constraints in linear real arithmetic, which can be handled efficiently (e.g. by simplex algorithm in polynomial time on average or by the interior point method [49] in polynomial time).

2. **Usability.** Integer-weighted synthesis requires the user to provide an a priori upper bound on the weights to reduce the size of the search space. Choosing such a bound is difficult in practice; working over $\mathbb{R}$ allows us to avoid this parameter and thereby makes the method more accessible.

The implementation of both our approach and the approach proposed by Endrullis and Overbeek [39], written in OCaml, will be presented in Chapter 6.

This chapter is organized as follows. § 3.1 introduces the type graph method [39]; § 3.2 presents our extension to non-well-founded semirings; and § 3.3 concludes.

3.1 Preliminaries

We introduce the type graph method from [39] in this section.

3.1.1 Well-Founded Semirings

Type graphs are weighted over semirings which are algebraic structures as defined below.

Definition 3.1. *A **monoid** $(S, \cdot, e)$ is a set S equipped with a binary operation $\cdot : S \times S \to S$ and having a special element e such that:*

- $\cdot$ *is associative:* $(x \cdot y) \cdot z = x \cdot (y \cdot z)$,
- e *is an identity element:* $e \cdot x = x = x \cdot e$.

*A **commutative monoid** is a monoid where $\cdot$ is commutative: $x \cdot y = y \cdot x$.*

Definition 3.2. *A **semiring** is a 5-tuple $(S, \oplus, \odot, 0, 1)$ satisfying the following conditions:*

- $(S, \oplus, 0)$ *is a commutative monoid,*
- $(S, \odot, 1)$ *is a monoid,*
- $\odot$ *is distributive over* $\oplus$*:*
  - $a \odot (b \oplus c) = (a \odot b) \oplus (a \odot c)$,
  - $(b \oplus c) \odot a = (b \odot a) \oplus (c \odot a)$,
- *0 is an annihilator for* $\odot$*:* $x \odot 0 = 0 = 0 \odot x$.

A semiring is said to be commutative if $(S, \odot, 1)$ *is a commutative monoid.*

The following definition of well-founded semirings is from [39].

Definition 3.3. *A* ***well-founded semiring*** $(S, \oplus, \odot, 0, 1, \prec, \preceq)$ *consists of*

- *a semiring* $(S, \oplus, \odot, 0, 1)$*, and*
- *non-empty orders* $\prec, \preceq \subseteq S \times S$ *for which* $\prec \subseteq \preceq$ *and* $\preceq$ *is reflexive,*

such that $\succ$ *is terminating relative to* $\succeq$*,* $0 \neq 1$ *and for all* $x, y, z, w \in S$ *we have*

$$x \preceq x' \wedge y \preceq y' \Rightarrow x \oplus y \preceq x' \oplus y', \tag{S1}$$
$$x \prec x' \wedge y \prec y' \Rightarrow x \oplus y \prec x' \oplus y', \tag{S2}$$
$$x \preceq x' \wedge 1 \preceq y \Rightarrow x \odot y \preceq x' \odot y \wedge y \odot x \preceq y \odot x', \tag{S3}$$
$$x \prec x' \wedge 1 \preceq y \neq 0 \Rightarrow x \odot y \prec x' \odot y \wedge y \odot x \prec y \odot x'. \tag{S4}$$

The semiring is ***strictly monotonic*** *if it additionally satisfies*

$$x \prec x' \wedge y \preceq y' \Rightarrow x \oplus y \prec x' \oplus y. \tag{S5}$$

Example 3.4. *The* ***natural tropical semiring***

$$\mathfrak{T} = (\mathbb{N} \cup \{+\infty\}, \min_{\mathbb{N} \cup \{+\infty\}}, +_{\mathbb{N} \cup \{+\infty\}}, +\infty, 0_{\mathbb{N} \cup \{+\infty\}}, <_{\mathbb{N} \cup \{+\infty\}}, \leq_{\mathbb{N} \cup \{+\infty\}})$$

is an instance of the well-founded semiring where

$$\begin{aligned} S &\longmapsto \mathbb{N} \cup \{+\infty\}, \\ \oplus &\longmapsto \min_{\mathbb{N} \cup \{+\infty\}}, \\ \odot &\longmapsto +_{\mathbb{N} \cup \{+\infty\}}, \\ 0_s &\longmapsto +\infty, \\ 1_s &\longmapsto 0_{\mathbb{N} \cup \{+\infty\}}, \\ \prec &\longmapsto <_{\mathbb{N} \cup \{+\infty\}}, \\ \preceq &\longmapsto \leq_{\mathbb{N} \cup \{+\infty\}}. \end{aligned}$$

It is a well-founded semiring but not strictly monotonic because $2 <_{\mathbb{N}} 3$ *but* $2 \oplus 2 \stackrel{\text{def}}{=} \min(2,2) = 2 \not<_{\mathbb{N}} 2 = \min(3,2) \stackrel{\text{def}}{=} 3 \oplus 2$.

Example 3.5. *The* ***natural arctic semiring***

$$\mathfrak{A} = (\mathbb{N} \cup \{-\infty\}, \max_{\mathbb{N} \cup \{-\infty\}}, +_{\mathbb{N} \cup \{-\infty\}}, -\infty, 0_{\mathbb{N} \cup \{-\infty\}}, <_{\mathbb{N} \cup \{-\infty\}}, \leq_{\mathbb{N} \cup \{-\infty\}})$$

is an instance of the well-founded semiring where

$$\begin{aligned} S &\longmapsto \mathbb{N} \cup \{-\infty\}, \\ \oplus &\longmapsto \max_{\mathbb{N} \cup \{-\infty\}}, \\ \odot &\longmapsto +_{\mathbb{N} \cup \{-\infty\}}, \\ 0_s &\longmapsto -\infty, \\ 1_s &\longmapsto 0_{\mathbb{N} \cup \{-\infty\}}, \\ \prec &\longmapsto <_{\mathbb{N} \cup \{-\infty\}}, \\ \preceq &\longmapsto \leq_{\mathbb{N} \cup \{-\infty\}}. \end{aligned}$$

It is a well-founded semiring but not strictly monotonic because $2 <_{\mathbb{N}} 3$ *but* $2 \oplus 3 \stackrel{\text{def}}{=} \max(2,3) = 3 \not<_{\mathbb{N}} 3 = \max(3,3) \stackrel{\text{def}}{=} 3 \oplus 3$.

Example 3.6. *The* ***natural arithmetic semiring*** $\mathfrak{N}=(\mathbb{N},+_{\mathbb{N}},*_{\mathbb{N}},0_{\mathbb{N}},1_{\mathbb{N}},<_{\mathbb{N}},\leq_{\mathbb{N}})$ *is an instance of the well-founded semiring where*

$$\begin{aligned} S &\longmapsto \mathbb{N},\\ \oplus &\longmapsto +_{\mathbb{N}},\\ \odot &\longmapsto *_{\mathbb{N}},\\ 0_s &\longmapsto 0_{\mathbb{N}},\\ 1_s &\longmapsto 1_{\mathbb{N}},\\ \prec &\longmapsto <_{\mathbb{N}},\\ \preceq &\longmapsto \leq_{\mathbb{N}}. \end{aligned}$$

It is strictly monotonic.

Notation 3.7. *Let* $(S,\oplus,\odot,0,1)$ *be a semiring. We extend naturally the binary operations* $\oplus$ *and* $\odot$ *to finite sets* $E\subseteq S$ *by letting*

- $\bigodot\varnothing \stackrel{\text{def}}{=} 1$ *and* $\bigodot(E\cup\{x\}) \stackrel{\text{def}}{=} (\bigodot E)\odot x$,
- $\bigoplus\varnothing \stackrel{\text{def}}{=} 0$ *and* $\bigoplus(E\cup\{x\}) \stackrel{\text{def}}{=} (\bigoplus E)\oplus x$.

We define the exponentiation operation for elements of the semiring before defining formally the morphism weight.

Notation 3.8. *Let* $(S,\oplus,\odot,0_S,1_S)$ *be a semiring. We define the* ***exponentiation operation*** *for all* $x\in S$ *and* $n\in\mathbb{N}$ *by*

- $x^0 \stackrel{\text{def}}{=} 1_S$,
- $x^{n+1} \stackrel{\text{def}}{=} x^n\odot x$.

3.1.2 Weighted Type Graph

Throughout this chapter, let $\mathcal{C}$ be a fixed locally small category. A weighted type graph is an object T in $\mathcal{C}$, called the type graph, equipped with a distinguished family of morphisms into T, each endowed with a weight. A "type graph" is an object, following the terminology from the previous work by Bruggink [19]. Given a weighted type graph and any object X in $\mathcal{C}$, we interpret X as $\mathrm{Hom}(X,T)$. The distinguished morphisms determine weight assignments on the elements of $\mathrm{Hom}(X,T)$; the weight of X is then defined as the sum of the weights of the morphisms in $\mathrm{Hom}(X,T)$. These assignments provide a basis for comparing objects of $\mathcal{C}$ according to their weights.

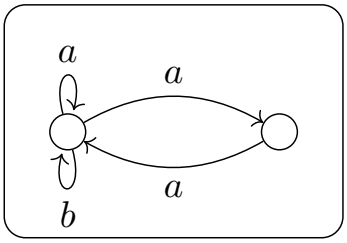
Definition 3.9 ([38])**.** *A* ***weighted type graph*** *is a tuple* $\mathcal{T}=(T,\mathbb{E},\mathcal{S},w)$ *where*

- T *is an object of* $\mathcal{C}$*, called* ***type graph***,
- $\mathbb{E}$ *is a set of morphisms in* $\mathcal{C}$ *with codomain* T*, and the elements of* $\mathbb{E}$ *are called* ***morphism-rulers***,
- $\mathcal{S}=(S,\oplus,\odot,0,1)$ *is a commutative semiring,*
- $w:\mathbb{E}\to S\setminus\{0\}$ *is a weight function such that for all* $e\in\mathbb{E}, w(e)\geq 1$,

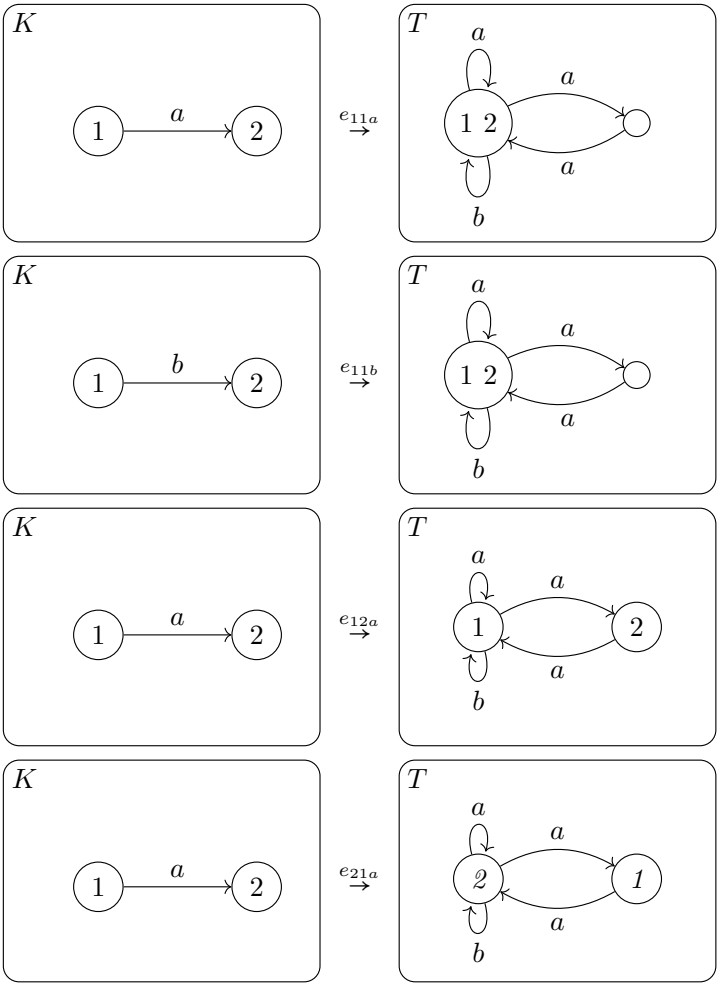
such that for every $(e : X \to T) \in \mathbb{E}$ *and every object* G, $\mathrm{Hom}(X, G)$ *and* $\mathrm{Hom}(G, T)$ *are finite sets.*

Example 3.10. *Consider the weighted type graph* $\mathcal{T} = (T, \mathbb{E}, \mathcal{S}, w)$ *where*

- T *is the labeled graph shown below:*

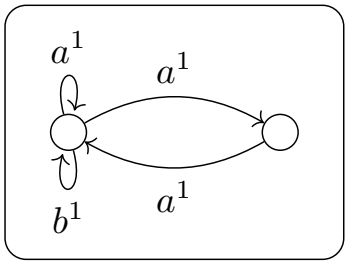


- $\mathcal{S}$ *is the natural arithmetic semiring* $(\mathbb{N}, +_{\mathbb{N}}, *_{\mathbb{N}}, 0_{\mathbb{N}}, 1_{\mathbb{N}}, <_{\mathbb{N}}, \leq_{\mathbb{N}})$,
- $\mathbb{E}$ *consists of 4 morphism-rulers* $e_{11a}, e_{12a}, e_{21a}, e_{11b}$ *shown below:*



- $w(e) = 1_{\mathbb{N}}$ *for all* $e \in \mathbb{E}$.

To improve readability, this weighted type graph will visualized as shown below, where a label l *with superscript* n *on an edge means that there is a morphism-ruler with weight* n *from* $\bigcirc \xrightarrow{l} \bigcirc$ *to this edge in* $\mathbb{E}$.



3.1.3 Measuring objects by counting morphisms

We recall the definition of morphism weight and object weight relative to a type graph [39].

Notation 3.11 ([38])**.** *Let* A, B, C *be objects. For morphism* $\alpha : A \to B$*, we introduce the notation*

$$\{\alpha \star - = \gamma\} \stackrel{\text{def}}{=} \{\beta \in \mathrm{Hom}(B, C) \mid \alpha \star \beta = \gamma\}.$$

For morphism $\beta : B \to C$*, we introduce the notation*

$$\{- \star \beta = \gamma\} \stackrel{\text{def}}{=} \{\alpha \in \mathrm{Hom}(A, B) \mid \alpha \star \beta = \gamma\}.$$

For morphism set $E \subseteq \mathrm{Hom}(A, B)$ *and morphism* $\beta : B \to C$*, we introduce the notation*

$$E \star \beta \stackrel{\text{def}}{=} \{\alpha \star \beta \mid \alpha \in E\}.$$

Analogous to measuring a physical object, a morphism ruler measures a morphism as follows.

Definition 3.12. *The* measurement *of a morphism* $h : G \to T$ *relative to a morphism-ruler* $e : X \to T$*, denoted by* $m_e(h)$*, is defined as:* $m_e(h) \stackrel{\text{def}}{=} |\{- \star h = e\}|$.

Combining the measurements of a morphism provided by all morphism-rulers with the help of a weight function w gives the weight of the morphism (Definition 3.13).

Definition 3.13. *Let* $\mathcal{T} = (T, \mathbb{E}, \mathcal{S}, w)$ *be a finitary weighted type graph. The* ***weight of a morphism*** $h : G \to T$ ***relative to a type graph*** $\mathcal{T}$ *is defined as the semiring product of* $w(e)^{m_e(h)}$ *for all* $e \in \mathbb{E}$*:*

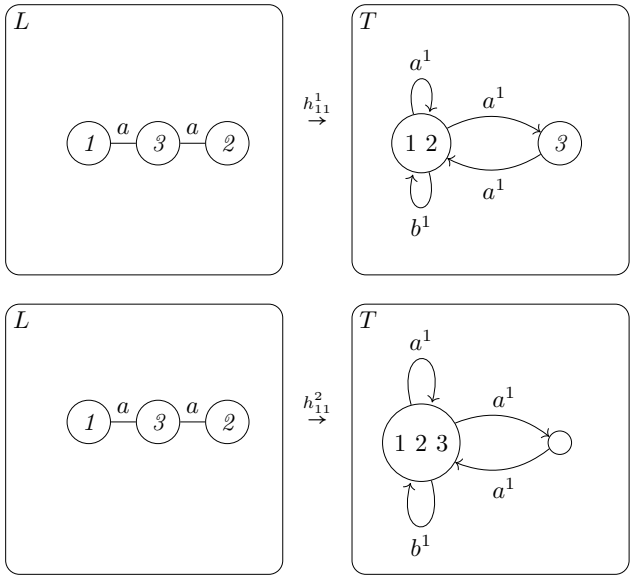
$$w_{\mathcal{T}}(h) \stackrel{\text{def}}{=} \bigodot_{e \in \mathbb{E}} w(e)^{m_e(h)}.$$

Finally, the weight of an object is defined as the semiring sum of the weights of all morphisms from the object to the underlying type graph T of the weighted type graph $\mathcal{T}$.

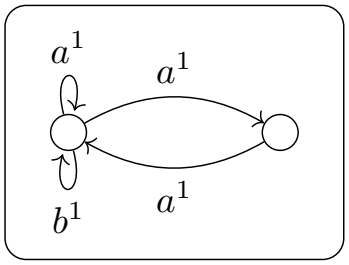
Definition 3.14 (Object weight)**.** *Let $\mathcal{T} = (T, \mathbb{E}, \mathcal{S}, w)$ be a finitary weighted type graph. The **weight of an object** G is defined as the semiring sum of $w_{\mathcal{T}}(h)$, for all $h \in \mathrm{Hom}(G, T)$:*

$$w_{\mathcal{T}}(G) \stackrel{\text{def}}{=} \bigoplus_{h \in \mathrm{Hom}(G,T)} w_{\mathcal{T}}(h).$$

Example 3.15. *Consider the two morphisms shown below*



and the weighted type graph shown below



We have

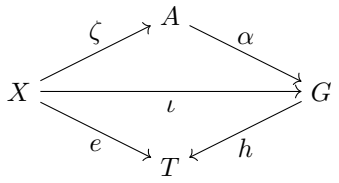
- $w_{\mathcal{T}}(h^1_{11}) = w(e_{13a})^{m_{e_{13a}}(h^1_{11})} \odot w(e_{31a})^{m_{e_{31a}}(h^1_{11})} = 1^1 * 1^1 = 1$,
- $w_{\mathcal{T}}(h^2_{11}) = 1$.

3.1.4 Precise and upperbound of weights of pushout objects

We first introduce the notions of *morphism measurement excluding morphisms in a set*(Definition 3.16) and *weight of a morphism relative to a morphism-ruler set excluding some specific morphisms* (Definition 3.17), to reduce notational overhead (following Endrullis and Overbeek [39]). We then recall the notions

of traceability of objects (Definition 3.18) and relative monicity of morphisms (Definition 3.21), which are used to define the notion of weighable pushout squares (Definition 3.23). With the notion of a weighable pushout square, we recall a result from Endrullis and Overbeek [39] which shows that the weight of the pushout object of a weighable pushout square can be bounded by the weights of the other objects in the pushout square.

For any set $\Gamma \subseteq \mathrm{Hom}(A, G)$, we define Γ' as the set consisting of all morphisms $\iota : X \to G$ admitting morphisms $\zeta : X \to A$ and $\alpha \in \Gamma$ such that $\zeta \star \alpha = \iota$ holds, i.e., such that the diagram XAG shown below is commutative. Formally, $\Gamma' \stackrel{\text{def}}{=} \{\iota \in \mathrm{Hom}(X, G) \mid \exists \alpha \in \Gamma,\ \exists \zeta : X \to A,\ \zeta \star \alpha = \iota\}$.



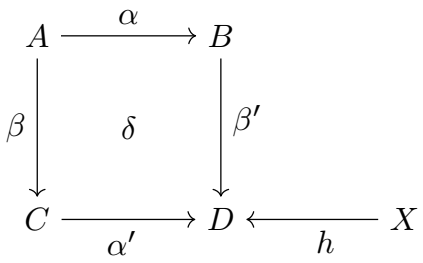
Definition 3.16. *Let $\Gamma \subseteq \mathrm{Hom}(A, G)$. The **measurement of a morphism** $h : G \to T$ **relative to a morphism-ruler** $e : X \to T$**, excluding morphisms in** Γ', denoted by $m_e(h - \Gamma)$, is defined as:*

$$m_e(h - \Gamma) \stackrel{\text{def}}{=} |\{- \star h = e\} \setminus \Gamma'|.$$

Definition 3.17. *Let $\mathcal{T} = (T, \mathbb{E}, \mathcal{S}, w)$ be a weighted type graph. The **weight of a morphism** $h : G \to T$ **relative to a set** $\mathbb{E}$ **of morphism-rulers excluding morphisms in** Γ' is defined as the semiring product of $w(e)^{w_e(h-\Gamma)}$ for $e \in \mathbb{E}$:*

$$w_{\mathcal{T}}(h - \Gamma) \stackrel{\text{def}}{=} \bigodot_{e \in \mathbb{E}} w(e)^{m_e(h-\Gamma)}.$$

Definition 3.18 ([38])**.** *Let Δ be a class of pushout squares. An object X is said to be **traceable along** Δ if for every diagram δ in Δ, as shown below:*



one of the following conditions holds:

(a) there is a morphism $f : X \to B$ such that $h = f \star \beta'$, or

(b) there is a morphism $g : X \to C$ such that $h = g \star \alpha'$.

If additionally, whenever (a) and (b) hold,

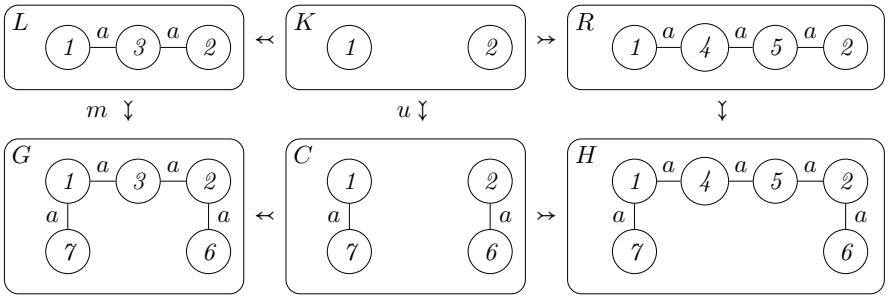
(c) there is a morphism $k : X \to A$ such that $h = k \star \alpha \star \beta'$,

*then we say that X is **strongly traceable** along Δ.*

Intuitively, in the category **Graph**, a graph X is β-strongly traceable along a pushout square, if whenever X occurs in D, it occurs either in B or in C, and if it occurs in both, then it occurs in A as well.

Remark 3.19 ([38]). *In **Graph**, the objects $\bigcirc$ and $\bigcirc \xrightarrow{x} \bigcirc$ (for edge labels x) are the only (non-initial) objects that are (strongly) traceable along all pushout squares. Other objects, such as loops $\bigcirc\!\!\circlearrowleft x$, are strongly traceable if all morphisms in the square are monomorphisms.*

Example 3.20. *Consider the following DPO diagram:*



The graph $\bigcirc \xrightarrow{a} \bigcirc \xrightarrow{a} \bigcirc$ is not traceable in both pushout squares. The graph $\bigcirc \xrightarrow{a} \bigcirc$ is traceable (and strongly traceable) along both pushout squares.

The following concepts are restricted versions of the concept of monicity.

Definition 3.21 ([38]). *Let A, B, C and X be objects, Γ a set of objects, S a set of morphisms to A and $u : C \to A$ a morphism. A morphism $f : A \to B$ is said to be*

- ***monic for** S if $g \star f = h \star f$ implies $g = h$ for all $g, h \in S$;*
- *X**-monic** if f is monic for $\mathrm{Hom}(X, A)$;*
- *X**-monic outside of** u, if f is monic for $\mathrm{Hom}(X, A) \setminus (\mathrm{Hom}(X, C) \star u)$;*
- *Γ**-monic** if f is X-monic for every $X \in \Gamma$.*

Example 3.22 (Edge-Monicity [38]). *Let $f : G \to H$ be a morphism in* **Graph** *that does not identify distinct edges, but may identify distinct nodes. Then f is not necessarily monic. Indeed, if $u \neq v$ are nodes of G with $f(u) = f(v)$, let K be the graph consisting of a single node and no edges, and let $g, h : K \to G$ send that node to u and v, respectively. Then $g \neq h$ while $f \circ g = f \circ h$, so f is not monic.*

Nevertheless f is Γ-monic, where

$$\Gamma = \left\{ \bigcirc \xrightarrow{x} \bigcirc \mid x \text{ is an edge label} \right\}.$$

To see this, let $X \in \Gamma$ and let $g, h : X \to G$ satisfy $g \star f = h \star f$. Let e denote the unique edge of X. From $f(g(e)) = f(h(e))$ and the hypothesis that f is injective on edges we obtain $g(e) = h(e)$. Since we are working with directed graphs, the images of the two nodes of X are determined by the image of e, so g and h agree on both nodes. Hence $g = h$, and f is Γ-monic.

Definition 3.23 ([38, Def. 4.9]). *Let $\mathcal{T}=(T,\mathbb{E},S,w)$ be a weighted type graph. Consider the pushout square δ shown below.*

$$\begin{array}{ccc} A & \xrightarrow{\alpha} & B \\ {\scriptstyle\beta}\downarrow & \delta & \downarrow{\scriptstyle\beta'} \\ C & \xrightarrow[\alpha']{} & D \end{array}$$

δ is said to be

(a) ***weighable*** *with $\mathcal{T}$ if the following conditions hold:*

(i) $dom(\mathbb{E})$ is strongly traceable along δ,

(ii) β' is $dom(\mathbb{E})$-monic,

(iii) α' is $dom(\mathbb{E})$-monic outside of β.

(b) ***bounded-above*** *by $\mathcal{T}$ if $dom(\mathbb{E})$ is traceable along δ.*

The following lemma shows that, under certain constraints, exact weights of pushout object can be precisely computed or upperbounded.

Lemma 3.24 ([38, Lemma 4.13]). *Let $\mathcal{T}=(T,\mathbb{E},S,w)$ be a finitary weighted type graph. Consider the pushout square δ in Example 3.23. We define*

$$k=\bigoplus_{t_A:A\to T}\left(\bigoplus_{\substack{t_C:C\to T\\ t_A=\beta\star t_C}} w_{\mathcal{T}}(t_C-\beta)\right)\odot w_{\mathcal{T}}(\{\alpha\star -=t_A\})$$

The following conditions hold:

(A) $w_{\mathcal{T}}(D)=k$ if δ is weighable with $\mathcal{T}$.

(B) $w_{\mathcal{T}}(D)\leq k$ if δ is bounded above by $\mathcal{T}$ and $w(e)\geq 1_S$ for all $e\in\mathbb{E}$.

Consider the following DPO diagram that defines a rewriting step $G\Rightarrow_{\rho,\mathfrak{F}} H$.

$$\begin{array}{ccccc} L & \xleftarrow[l]{} & K & \xrightarrow[r]{} & R \\ {\scriptstyle m}\downarrow & \text{PO} & {\scriptstyle u}\downarrow & \text{PO} & \downarrow{\scriptstyle m'} \\ G & \xleftarrow[l']{} & C & \xrightarrow[r']{} & H \end{array}$$

If the left pushout square is weighable with $\mathcal{T}$ and the right pushout square is bounded above by $\mathcal{T}$, by Lemma 3.24, we obtain:

$$w_{\mathcal{T}}(G) = \bigoplus_{t_K : K \to T} \left(\bigoplus_{\substack{t_C : C \to T \\ t_K = u \star t_C}} w_{\mathcal{T}}(t_C - u) \right) \odot \left(\bigoplus_{\substack{t_L : L \to T \\ t_K = l \star t_L}} w_{\mathcal{T}}(t_L) \right)$$

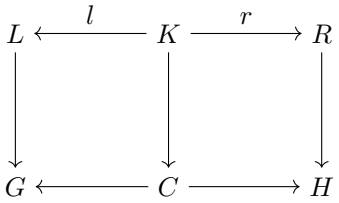
$$w_{\mathcal{T}}(H) \leq \bigoplus_{t_K : K \to T} \left(\bigoplus_{\substack{t_C : C \to T \\ t_K = u \star t_C}} w_{\mathcal{T}}(t_C - u) \right) \odot \left(\bigoplus_{\substack{t_R : R \to T \\ t_K = r \star t_R}} w_{\mathcal{T}}(t_R) \right)$$

According to these results, in §3.1.7, we establish a termination criterion by, for every $t_K : K \to T$, comparing $\bigoplus_{\substack{t_L : L \to T \\ t_K = l \star t_L}} w_{\mathcal{T}}(t_L)$ and $\bigoplus_{\substack{t_R : R \to T \\ t_K = r \star t_R}} w_{\mathcal{T}}(t_R)$. Before doing so, we introduce the concept of context closure in the following section.

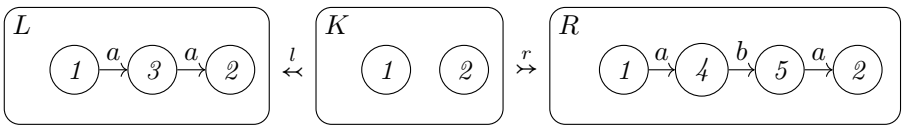
3.1.5 Context Closure

It is crucial to ensure that the weight of any object subject to rewriting is not 0_S, because 0_S behaves unpredictably in strongly monotonic measurable semirings. For instance, in the natural tropical semiring $(\mathbb{N} \cup \{+\infty\}, \min, +)$, the element 0_S is the greatest element $+\infty$, while in the natural arctic semirings $(\mathbb{N} \cup \{-\infty\}, \max, +)$, the element 0_S is the smallest element $-\infty$. This issue is addressed by the existence of a context closure defined as follows:

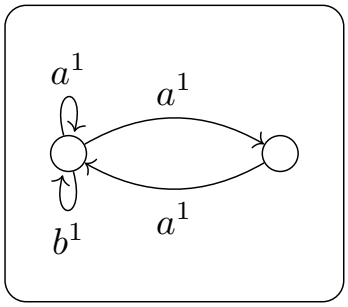
Definition 3.25 ([38])**.** *Let $\mathcal{T} = (T, \mathbb{E}, \mathcal{S}, w)$ be a finitary weighted type graph, $\rho = (L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$ a DPO rewriting rule and $\mathfrak{F}$ a rewriting framework. A **context closure** for ρ and $\mathcal{T}$ in $\mathfrak{F}$ is a morphism $c : L \to T$ such that for every DPO diagram in $\mathfrak{F}(\rho)$ shown below, there is $\alpha : G \to T$ such that $m \star \alpha = c$.*



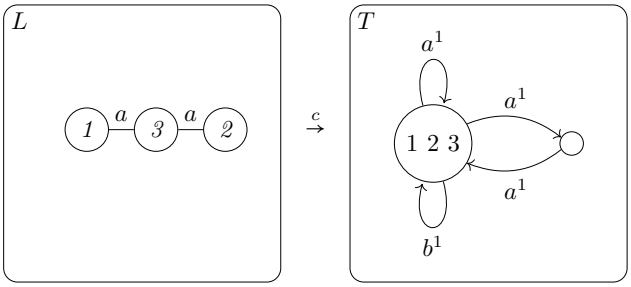
Example 3.26. *Let the set of edge labels be $\Sigma = \{a, b\}$. Consider the following DPO rule:*



and the following weighted type graph:



Let c be the morphism shown below. Since for every match $m : L \to G$, there is a morphism $\alpha : G \to T$ such that $m \star \alpha = c$, the morphism c is a context closure for the DPO rule in the type graph.



3.1.6 Decreasing rules

The following definition of decreasing rules from [38] classifies DPO rewriting rules.

Definition 3.27 ([38])**.** *Let $\mathcal{T} = (T, \mathbb{E}, (S, \oplus, \odot, 0_S, 1_S, <, \leq), w)$ be a finitary weighted type graph, $\mathfrak{F}$ a DPO rewriting framework, $\rho = (L \overset{l}{\leftarrow} K \overset{r}{\to} R)$ a DPO rewriting rule.*
*The rule ρ is said to be **weakly decreasing** with respect to $\mathcal{T}$ in $\mathfrak{F}$ if for every $t_K : K \to T$, the following inequality holds:*

$$w_{\mathcal{T}}(\{l \star - = t_K\}) \geq w_{\mathcal{T}}(\{r \star - = t_K\}).$$

*The rule ρ is said to be **uniformly decreasing** with respect to $\mathcal{T}$ in $\mathfrak{F}$ if the following conditions holds:*

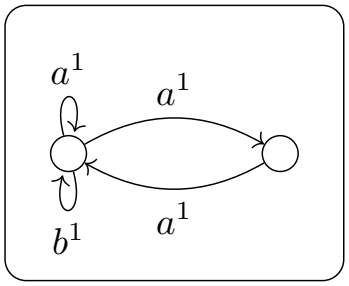
- *there is a context closure c_ρ for ρ and $\mathcal{T}$ in $\mathfrak{F}$,*
- *for every $t_K : K \to T$,*
  - $\{l \star - = t_K\} = \varnothing = \{r \star - = t_K\}$, *or*
  - $w_{\mathcal{T}}(\{l \star - = t_K\}) > w_{\mathcal{T}}(\{r \star - = t_K\})$.

*If S is moreover strictly monotonic, we say the rule ρ is **closure decreasing** with respect to $\mathcal{T}$ in $\mathfrak{F}$ if the following conditions hold:*

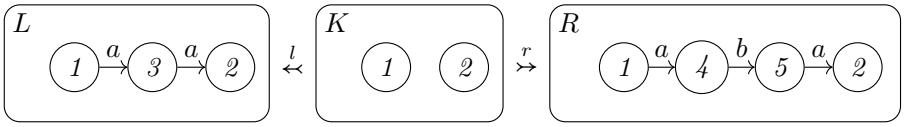
- *ρ is weakly decreasing,*
- *there is a context closure c_ρ for ρ and $\mathcal{T}$ in $\mathfrak{F}$,*

- $w_{\mathcal{T}}(\{l \star - = t_K\}) > w_{\mathcal{T}}(\{r \star - = t_K\})$ *for* $t_K = l \star c_\rho$.

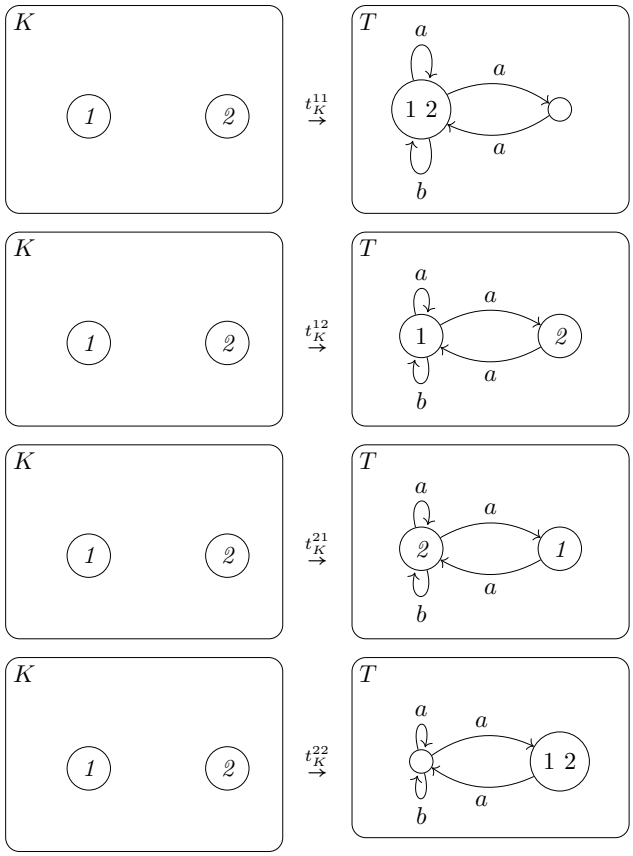
Example 3.28. *Consider the weighted type graph over the natural arithmetic semiring* $(\mathbb{N}, +, *, 0, 1, <, \leq)$*:*



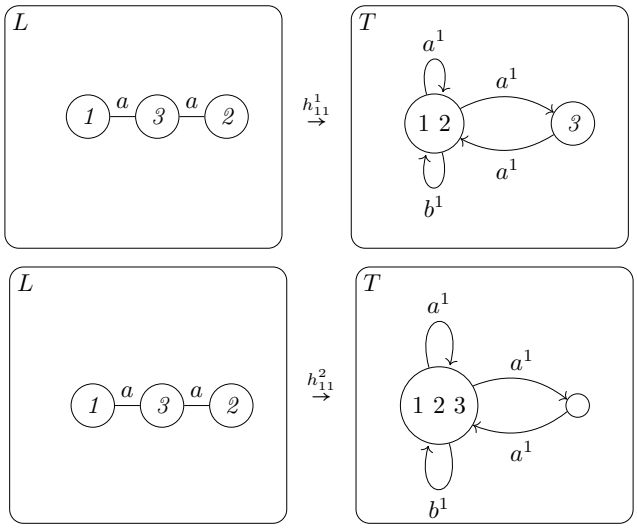
and the DPO rule:



There are 4 morphisms $t_K^{11}, t_K^{12}, t_K^{21}, t_K^{22}$ *from* K *to* T*:*



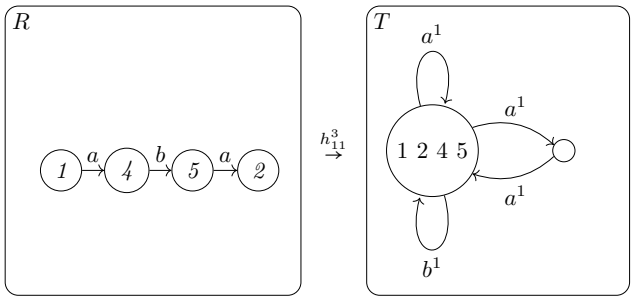
The set $\{l \star -= t_K^{11}\}$ *consists of two morphisms* h_{11}^1 *and* h_{11}^2*:*



Therefore, we have

$$\begin{aligned} w_{\mathcal{T}}(\{l \star -= t_K^{11}\}) =& w_{\mathcal{T}}(\{h_{11}^1, h_{11}^2\}) \\ =& w_{\mathcal{T}}(h_{11}^1) + w_{\mathcal{T}}(h_{11}^2) \\ =& (1^1 * 1^1) + (1^1 * 1^1) \\ =& 2. \end{aligned}$$

The set $\{r \star -= t_K^{11}\}$ *has one morphism* h_{11}^3*:*



Therefore, we have:

$$w_{\mathcal{T}}(\{r \star -= t_K^{11}\}) = w_{\mathcal{T}}(h_{11}^3) = 1^1 * 1^1 * 1^1 = 1.$$

The following inequality follows

$$w_{\mathcal{T}}(\{l \star -= t_K^{11}\}) = 2 \geq 1 = w_{\mathcal{T}}(\{r \star -= t_K^{11}\}).$$

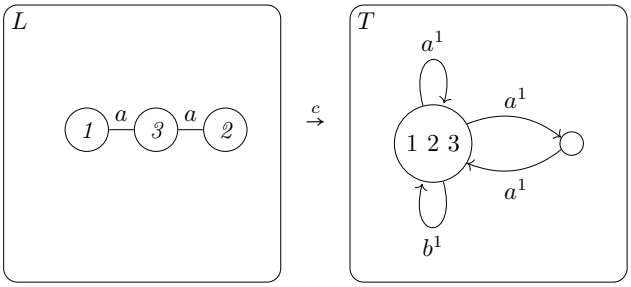
Similarly, we can check that

- $w_{\mathcal{T}}(\{l \star -= t_K^{12}\}) = 1 \geq 1 = w_{\mathcal{T}}(\{r \star -= t_K^{12}\})$*, and*
- $w_{\mathcal{T}}(\{l \star -= t_K^{21}\}) = 1 \geq 1 = w_{\mathcal{T}}(\{r \star -= t_K^{21}\})$*, and*

- $w_{\mathcal{T}}(\{l \star - = t_K^{22}\}) = 1 \geq 1 = w_{\mathcal{T}}(\{r \star - = t_K^{22}\})$.

The rule is therefore weakly decreasing.

The morphism c illustrated below is a context closure for the DPO rule and the weighted type graph shown above, as explained in Example 3.28.



Since the following conditions hold:

- $t_K^{11} = l \star c$, *and*
- $w_{\mathcal{T}}(\{l \star - = t_K^{11}\}) = 2 > 1 = w_{\mathcal{T}}(\{r \star - = t_K^{11}\})$,

we conclude that the rule is closure decreasing since the semiring is strictly monotonic.

3.1.7 Termination Criterion

Finally, we can state the main result from [39].

Theorem 3.29 (Termination of DPO rewriting system)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be sets of DPO rewriting rules, $\mathcal{T} = (T, \mathbb{E}, (S, \oplus, \odot, 0_S, 1_S, \prec, \leq), w)$ a finitary weighted type graph and $\mathfrak{F}$ a DPO rewriting framework such that*

- $\mathrm{left}(\Delta)$ *is weighable with* $\mathcal{T}$,
- $\mathrm{right}(\Delta)$ *is bounded above by* $\mathcal{T}$.

for every rule $\rho \in (\mathcal{A} \cup \mathcal{B})$ and every double pushout diagram $\Delta \in \mathfrak{F}(\rho)$. If the following conditions hold:

1. *either every $\rho \in \mathcal{A}$ is uniformly decreasing or every $\rho \in \mathcal{A}$ is closure decreasing, and*
2. *every rule $\rho \in \mathcal{B}$ is weakly decreasing,*

then $\Rightarrow_{\mathcal{A},\mathfrak{F}}$ is ***terminating*** *relative to $\Rightarrow_{\mathcal{B},\mathfrak{F}}$.*

Example 3.30. *Termination of the DPO rule in Example 2.29 can be established using Theorem 3.29 together with the weighted type graph in Example 3.10 over the natural arithmetic semiring $\mathfrak{N} = (\mathbb{N}, +, *, 0_{\mathbb{R}}, 1_{\mathbb{N}}, <, \leq)$. It is closure decreasing as explained in Example 3.28.*

3.2 Extension to Non-Well-Founded Semirings

We introduce the notion of a strongly monotonic measurable semiring. The key difference with well-founded semirings (Definition 3.3) is that the semiring is required to be equipped with a homomorphism to the extended real numbers instead of being well-founded. Hereafter, $<$ and $\leq$ denote the canonical irreflexive and reflexive orders on the set of extended real numbers $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$.

Definition 3.31. *A **strongly monotonic measurable semiring** $(S, \oplus, \odot, 0, 1, \prec, \mu)$ consists of*

- *A commutative semiring $(S, \oplus, \odot, 0, 1)$,*
- *A non-empty irreflexive order $\prec$ on S,*
- *A homomorphism $\mu : (S, \prec) \to (\overline{\mathbb{R}}, <_{\overline{\mathbb{R}}})$,*

such that $0 \neq 1$ and for all $x, y, z, w \in S$, for all $\delta \in \mathbb{R}$ and $\delta > 0$, we have

$$1 \leq x \wedge 1 \leq y \Rightarrow 1 \leq x \oplus y, \tag{S0}$$
$$x \leq x' \wedge y \leq y' \Rightarrow x \oplus y \leq x' \oplus y', \tag{S1}$$
$$x \prec x' \wedge y \prec y' \Rightarrow x \oplus y \prec x' \oplus y', \tag{S2}$$
$$\delta + \mu(x) <_{\overline{\mathbb{R}}} \mu(y) \wedge \delta + \mu(z) <_{\overline{\mathbb{R}}} \mu(w) \Rightarrow \delta + \mu(x \oplus z) <_{\overline{\mathbb{R}}} \mu(y \oplus w), \tag{S3}$$
$$x \leq x' \Rightarrow x \odot y \leq x' \odot y, \tag{S4}$$
$$x \prec x' \wedge y \neq 0 \Rightarrow x \odot y \prec x' \odot y, \tag{S5}$$
$$\delta + \mu(x) <_{\overline{\mathbb{R}}} \mu(y) \wedge 1 \leq z \wedge z \neq 0 \Rightarrow \delta + \mu(x \odot z) <_{\overline{\mathbb{R}}} \mu(y \odot z), \tag{S6}$$
$$\delta + \mu(x) <_{\overline{\mathbb{R}}} \mu(x') \wedge y \neq 0 \Rightarrow \mu(x \odot y) <_{\overline{\mathbb{R}}} \mu(x' \odot y). \tag{S7}$$

*where $\leq$ denotes the reflexive closure of $\prec$. The semiring is a **strictly monotonic measurable semiring** if it additionally satisfies*

$$x \prec x' \Rightarrow x \oplus y \prec x' \oplus y, \tag{S8}$$
$$\delta + \mu(x) <_{\overline{\mathbb{R}}} \mu(x') \Rightarrow \delta + \mu(x \oplus y) <_{\overline{\mathbb{R}}} \mu(x' \oplus y). \tag{S9}$$

Example 3.32. *The **real tropical semiring***

$$\mathfrak{T}' \stackrel{\text{def}}{=} (\mathbb{R} \cup \{+\infty\}, \min_{\mathbb{R} \cup \{+\infty\}}, +_{\mathbb{R} \cup \{+\infty\}}, +\infty, 0_{\mathbb{R} \cup \{+\infty\}}, <_{\mathbb{R} \cup \{+\infty\}}, \mathrm{id}_{\mathbb{R} \cup \{+\infty\}})$$

is an instance of the strongly monotonic measurable semiring where

$$\begin{aligned} S &\longmapsto \mathbb{R} \cup \{+\infty\}, \\ \oplus &\longmapsto \min_{\mathbb{R} \cup \{+\infty\}}, \\ \odot &\longmapsto +_{\mathbb{R} \cup \{+\infty\}}, \\ 0_s &\longmapsto +\infty, \\ 1_s &\longmapsto 0_{\mathbb{R} \cup \{+\infty\}}, \\ \prec &\longmapsto <_{\mathbb{R} \cup \{+\infty\}}, \\ \mu &\longmapsto \mathrm{id}_{\mathbb{R} \cup \{+\infty\}}. \end{aligned}$$

It is a strongly monotonic measurable semiring but not strictly monotonic measurable, because we have $2 <_{\mathbb{R} \cup \{+\infty\}} 3$ but

$$2 \oplus 2 \stackrel{\text{def}}{=} \min_{\mathbb{R} \cup \{+\infty\}}(2, 2) = 2 \not<_{\mathbb{R} \cup \{+\infty\}} 2 = \min_{\mathbb{R} \cup \{+\infty\}}(3, 2) \stackrel{\text{def}}{=} 3 \oplus 2.$$

Example 3.33. *The* ***real arctic semiring***

$$\mathfrak{A}' \stackrel{\text{def}}{=} (\mathbb{R} \cup \{-\infty\}, \max_{\mathbb{R} \cup \{-\infty\}}, +_{\mathbb{R} \cup \{-\infty\}}, -\infty, 0_{\mathbb{R} \cup \{-\infty\}}, <_{\mathbb{R} \cup \{-\infty\}}, \mathrm{id}_{\mathbb{R} \cup \{-\infty\}})$$

is an instance of the strongly monotonic measurable semiring where

$$\begin{aligned} S &\longmapsto \mathbb{R} \cup \{-\infty\}, \\ \oplus &\longmapsto \max_{\mathbb{R} \cup \{-\infty\}}, \\ \odot &\longmapsto +_{\mathbb{R} \cup \{-\infty\}}, \\ 0_s &\longmapsto -\infty, \\ 1_s &\longmapsto 0_{\mathbb{R} \cup \{-\infty\}}, \\ \prec &\longmapsto <_{\mathbb{R} \cup \{-\infty\}}, \\ \mu &\longmapsto \mathrm{id}_{\mathbb{R} \cup \{-\infty\}}. \end{aligned}$$

It is a strongly monotonic measurable semiring but not strictly monotonic measurable, because we have $2 <_{\mathbb{R} \cup \{-\infty\}} 3$ *but*

$$2 \oplus 3 \stackrel{\text{def}}{=} \max_{\mathbb{R} \cup \{-\infty\}}(2,3) = 3 \not<_{\mathbb{R} \cup \{-\infty\}} 3 = \max_{\mathbb{R} \cup \{-\infty\}}(3,3) \stackrel{\text{def}}{=} 3 \oplus 3.$$

Example 3.34. *The* ***real arithmetic semiring***

$$\mathfrak{N}' \stackrel{\text{def}}{=} (\mathbb{R}^+, +_{\mathbb{R}^+}, *_{\mathbb{R}^+}, 0_{\mathbb{R}^+}, 1_{\mathbb{R}^+}, <_{\mathbb{R}^+}, \mathrm{id}_{\mathbb{R}^+})$$

is an instance of the strongly monotonic measurable semiring where

$$\begin{aligned} S &\longmapsto \mathbb{R}^+, \\ \oplus &\longmapsto +_{\mathbb{R}^+}, \\ \odot &\longmapsto *_{\mathbb{R}^+}, \\ 0_s &\longmapsto 0_{\mathbb{R}^+}, \\ 1_s &\longmapsto 1_{\mathbb{R}^+}, \\ \prec &\longmapsto <_{\mathbb{R}^+}, \\ \mu &\longmapsto \mathrm{id}_{\mathbb{R}^+}. \end{aligned}$$

It is a strictly monotonic measurable semiring.

Example 3.35. *The* ***natural tropical semiring***

$$\mathfrak{T} \stackrel{\text{def}}{=} (\mathbb{N} \cup \{+\infty\}, \min_{\mathbb{N} \cup \{+\infty\}}, +_{\mathbb{N} \cup \{+\infty\}}, +\infty, 0_{\mathbb{N} \cup \{+\infty\}}, <_{\mathbb{N} \cup \{+\infty\}}, \mathrm{id}_{\mathbb{N} \cup \{+\infty\}})$$

is an instance of the strongly monotonic measurable semiring where

$$\begin{aligned} S &\longmapsto \mathbb{N} \cup \{+\infty\}, \\ \oplus &\longmapsto \min_{\mathbb{N} \cup \{+\infty\}}, \\ \odot &\longmapsto +_{\mathbb{N} \cup \{+\infty\}}, \\ 0_s &\longmapsto +\infty, \\ 1_s &\longmapsto 0_{\mathbb{N} \cup \{+\infty\}}, \\ \prec &\longmapsto <_{\mathbb{N} \cup \{+\infty\}}, \\ \mu &\longmapsto \mathrm{id}_{\mathbb{N} \cup \{+\infty\}}. \end{aligned}$$

*The **natural arctic semiring***

$$\mathfrak{A} \stackrel{\text{def}}{=} (\mathbb{N}\cup\{-\infty\}, \max_{\mathbb{N}\cup\{-\infty\}}, +_{\mathbb{N}\cup\{-\infty\}}, -\infty, 0_{\mathbb{N}\cup\{-\infty\}}, <_{\mathbb{N}\cup\{-\infty\}}, \mathrm{id}_{\mathbb{N}\cup\{-\infty\}})$$

is an instance of the strongly monotonic measurable semiring where

$$\begin{aligned} S &\longmapsto \mathbb{N}\cup\{-\infty\},\\ \oplus &\longmapsto \max_{\mathbb{N}\cup\{-\infty\}},\\ \odot &\longmapsto +_{\mathbb{N}\cup\{-\infty\}},\\ 0_s &\longmapsto -\infty,\\ 1_s &\longmapsto 0_{\mathbb{N}\cup\{-\infty\}},\\ \prec &\longmapsto <_{\mathbb{N}\cup\{-\infty\}},\\ \mu &\longmapsto \mathrm{id}_{\mathbb{N}\cup\{-\infty\}}\,. \end{aligned}$$

*The **Semiring!natural arithmetic***

$$\mathfrak{N} \stackrel{\text{def}}{=} (\mathbb{N}, +_{\mathbb{N}}, *_{\mathbb{N}}, +_{\mathbb{N}}, 0_{\mathbb{N}}, 1_{\mathbb{N}}, <_{\mathbb{N}}, \mathrm{id}_{\mathbb{N}})$$

is an instance of the strictly monotonic measurable semiring where

$$\begin{aligned} S &\longmapsto \mathbb{N},\\ \oplus &\longmapsto +_{\mathbb{N}},\\ \odot &\longmapsto *_{\mathbb{N}},\\ 0_s &\longmapsto 0_{\mathbb{N}},\\ 1_s &\longmapsto 1_{\mathbb{N}},\\ \prec &\longmapsto <_{\mathbb{N}},\\ \mu &\longmapsto \mathrm{id}_{\mathbb{N}}\,. \end{aligned}$$

3.2.1 Measuring objects by counting morphisms

The definitions of weighted type graph and the weight of a morphism and an object relative to a weighted type graph are the same as in § 3.1.2 and § 3.1.3.

Example 3.36. *Consider the weighted type graph $\mathcal{T}=(T,\mathbb{E},\mathcal{S},w)$ in **Graph** where*

- *$\mathcal{S}$ is the real arithmetic semiring $(\mathbb{R}^+,+,*,0_{\mathbb{R}^+},1_{\mathbb{R}^+},<,\mathrm{id}_{\mathbb{R}^+})$,*
- *T is the graph illustrated in Figure 3.1 without superscripts on the edge labels,*
- *$\mathbb{E} = \{e_{11a}, e_{12a}, e_{21a}, e_{11b}\}$ as the set of morphism-rulers where e_{uvl} has domain $\bigcirc \xrightarrow{l} \bigcirc$ and image $(\mathrm{u}) \xrightarrow{l} (\mathrm{v})$ in the graph T,*
- *$w(e) = 1.0$ for all $e \in \mathbb{E}$.*

It is visualized in Figure 3.1.

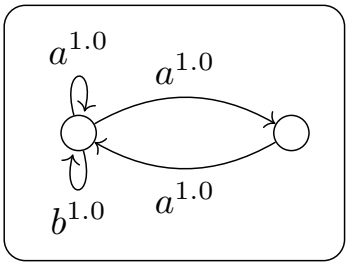
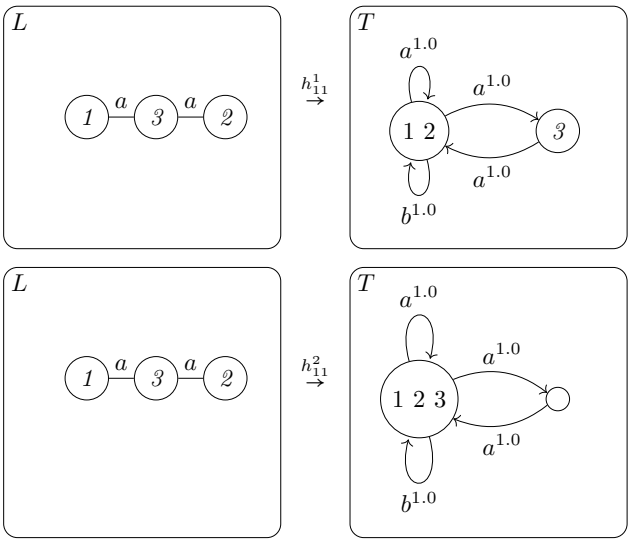


Figure 3.1

Consider the following morphisms:



We have

$$\begin{aligned} w_{\mathcal{T}}(h^1_{11}) =& w(e_{13a})^{m_{e_{13a}}(h^1_{11})} \odot w(e_{31a})^{m_{e_{31a}}(h^1_{11})} \\ =& 1.0^1 * 1.0^1 = 1.0. \end{aligned}$$

and

$$\begin{aligned} w_{\mathcal{T}}(h^2_{11}) =& 1.0^1 * 1.0^1 \\ =& 1.0. \end{aligned}$$

3.2.2 Precise and upperbound of weights of pushout objects

Let $\mathcal{R} = \mathcal{A} \cup \mathcal{B}$ be a set of DPO rewriting rules and let $\mathcal{T} = (T, \mathbb{E}, \mathcal{S}, w)$ be a weighted type graph over the strongly monotonic measurable semiring $\mathcal{S}$. The following proposition ensures that every morphism to a type graph $(T, S, \mathbb{E}, w)$ has a weight different from 0_S and greater than or equal to 1_S.

Proposition 3.37. *Let $(S, \oplus, \odot, 0_S, 1_S, \prec, \mu)$ be a strongly monotonic measurable semiring and $\preceq$ be the reflexive closure of $\prec$. We have for all $x, y \in S$:*

$$0_S \neq x \wedge 0_S \neq y \Rightarrow 0_S \neq x \odot y, \tag{S10}$$

$$1_S \preceq x \wedge 1_S \preceq y \wedge 0_S \neq x \wedge 0_S \neq y \Rightarrow 1_S \preceq x \odot y, \tag{S11}$$

$$0_S \neq x \wedge 0_S \neq y \Rightarrow 0_S \neq x \oplus y. \tag{S12}$$

Proof. Let $x, y \in S$ such that $x, y \neq 0_S$. By definition, $\prec$ is not empty, therefore, there are $a, b \in S$ such that $a \prec b$. $\prec$ is irreflexive, because $\mu : (S, \prec) \to (\overline{\mathbb{R}}, <_{\overline{\mathbb{R}}})$ is a homomorphism.

- S10: Suppose $(x \odot y) = 0_S$. We have

  $$\begin{aligned} 0_S &= a \odot 0_S && 0_S \text{ is annihilator for } \odot \\ &= a \odot (x \odot y) && \text{by assumption on } x \odot y \\ &= a \odot x \odot y && \text{by associativity} \\ &\prec b \odot x \odot y && \text{by (S5) and } x, y \neq 0_S \\ &= b \odot (x \odot y) && \text{by associativity} \\ &= b \odot 0_S && \text{by assumption on } x \odot y \\ &= 0_S \end{aligned}$$

  which contradicts the irreflexivity of $\prec$.

- S11: Suppose $1_S \preceq x$ and $1_S \preceq y$. We have either $1_S = x$ or $1_S \prec x$. If $1_S = x$ then

  $$\begin{aligned} x \odot y &= 1_S \odot y \\ &= y \\ &\geq 1_S && \text{by assumption} \end{aligned}$$

  If $1_S \prec x$ then $1_S \odot y \prec x \odot y$ by (S5) since $y \neq 0_S$ by assumption.

- S12: By (S5), we have $a \odot x \prec b \odot x$ and $a \odot y \prec b \odot y$.

  $$\begin{aligned} a \odot (x \oplus y) &= (a \odot x) \oplus \big(a \odot y\big) && \text{by distributivity} \\ &\prec (b \odot x) \oplus (b \odot y) && \text{by (S2)} \\ &= b \odot (x \oplus y) && \text{by distributivity} \end{aligned}$$

  if $x \oplus y = 0_S$ then we have $0_S = a \odot 0_S = a \odot (x \oplus y) \prec b \odot (x \oplus y) = b \odot 0_S = 0_S$ which contradicts the irreflexivity of $\prec$.

□

Suppose that for all $x \in \mathcal{S}$, if $1_{\mathcal{S}} \preceq x \neq 0_{\mathcal{S}}$ then $\mu(x) \geq_{\mathbb{R}^+} \mu(1_{\mathcal{S}})$ and $\mu(x) \in \mathbb{R}$. By definition of a weighted type graph, $1_{\mathcal{S}} \leq w(e) \neq 0_{\mathcal{S}}$ for all $e \in \mathbb{E}$. Therefore, by the proposition above and Definition 3.13 of the weight of a morphism, for all morphisms h from a finite graph G to $\mathcal{T}$, we have $0_{\mathcal{S}} \neq w_{\mathcal{T}}(h) \geq 1_{\mathcal{S}}$. By the above proposition, (S0) and Definition 3.14 of the weight of an object, we conclude that for all objects G with $\mathrm{Hom}(G, T) \neq \varnothing$, we have $0_{\mathcal{S}} \neq w_{\mathcal{T}}(G) \geq 1_{\mathcal{S}}$. By assumption, we have that for all objects G with $\mathrm{Hom}(G, T) \neq \varnothing$,

- $\mu(w_{\mathcal{T}}(G)) \geq_{\mathbb{R}^+} \mu(1_{\mathcal{S}}) = 0$ if $\mathcal{S}$ is the real tropical or arctic semiring, and
- $\mu(w_{\mathcal{T}}(G)) \geq_{\mathbb{R}^+} 1$ if $\mathcal{S}$ is the arithmetic semiring.

Suppose additionally that for every graph G subject to rewriting, we have $\mathrm{Hom}(G,T) \neq \varnothing$. The rewriting relation $\Rightarrow_{\mathcal{A},\mathfrak{F}}$ terminates relative to $\Rightarrow_{\mathcal{B},\mathfrak{F}}$ if there is a constant $\delta > 0$ such that the following conditions hold:

- for all $G \Rightarrow_{\mathcal{A},\mathfrak{F}} H$, $\mu(w_{\mathcal{T}}(G)) >_{\mathbb{R}^+} \mu(w_{\mathcal{T}}(H)) + \delta$, and
- for all $G \Rightarrow_{\mathcal{B},\mathfrak{F}} H$, $\mu(w_{\mathcal{T}}(G)) \geq \mu(w_{\mathcal{T}}(H))$.

However, directly verifying all rewriting steps is infeasible due to the potentially infinite number of rewriting steps.

Consider the following DPO diagram that defines a rewriting step $G \Rightarrow_{\rho,\mathfrak{F}} H$.

$$\begin{array}{ccccc} L & \xleftarrow{\ l\ } & K & \xrightarrow{\ r\ } & R \\ \downarrow{\scriptstyle m} & \mathrm{PO} & \downarrow{\scriptstyle u} & \mathrm{PO} & \downarrow{\scriptstyle m'} \\ G & \xleftarrow{\ l'\ } & C & \xrightarrow{\ r'\ } & H \end{array}$$

If the left pushout square is weighable with $\mathcal{T}$ and the right pushout square is bounded above by $\mathcal{T}$, by Lemma 3.24, we obtain:

$$w_{\mathcal{T}}(G) = \bigoplus_{t_K : K \to T} \left(\bigoplus_{\substack{t_C : C \to T \\ t_K = u \star t_C}} w_{\mathcal{T}}(t_C - u) \right) \odot \left(\bigoplus_{\substack{t_L : L \to T \\ t_K = l \star t_L}} w_{\mathcal{T}}(t_L) \right),$$

$$w_{\mathcal{T}}(H) \leq \bigoplus_{t_K : K \to T} \left(\bigoplus_{\substack{t_C : C \to T \\ t_K = u \star t_C}} w_{\mathcal{T}}(t_C - u) \right) \odot \left(\bigoplus_{\substack{t_R : R \to T \\ t_K = r \star t_R}} w_{\mathcal{T}}(t_R) \right).$$

According to these results, a termination criterion will be established by comparing the weights $\bigoplus_{\substack{t_L : L \to T \\ t_K = l \star t_L}} w_{\mathcal{T}}(t_L)$ and $\bigoplus_{\substack{t_R : R \to T \\ t_K = r \star t_R}} w_{\mathcal{T}}(t_R)$ for every $t_K : K \to T$.

3.2.3 Decreasing rules

The definition below adapts the concept of decreasing rules by Endrullis and Overbeek [38]. The difference lies in the treatment of weight comparisons for uniformly decreasing and closure decreasing rules: instead of directly comparing weights, we evaluate the difference between weights relative to the homomorphism μ of the strongly monotonic measurable semirings.

Definition 3.38. *Let $\mathcal{T} = (T, \mathbb{E}, (S, \oplus, \odot, 0_S, 1_S, \prec, \mu), w)$ be a finitary weighted type graph, $\mathfrak{F}$ a DPO rewriting framework, $\rho = (L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$ a DPO rewriting rule, and $\delta \in \mathbb{R}_{>0}$. The rule ρ is said to be δ-**uniformly decreasing** with respect to $\mathcal{T}$ in $\mathfrak{F}$ if the following conditions hold:*

- there is a context closure c_ρ for ρ and $\mathcal{T}$ in $\mathfrak{F}$,

- *for every* $t_K : K \to T$,

  - $\{l \star - = t_K\} = \varnothing = \{r \star - = t_K\}$, *or*
  - $\mu(w_{\mathcal{T}}(\{l \star - = t_K\})) >_{\mathbb{R}^+} \mu(w_{\mathcal{T}}(\{r \star - = t_K\})) + \delta$.

The rule ρ *is said to be* δ***-closure decreasing*** *with respect to* $\mathcal{T}$ *in* $\mathfrak{F}$ *if the following conditions hold:*

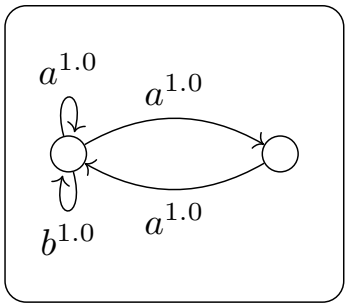
- S *is strictly monotonic measurable semiring,*

- ρ *is weakly decreasing,*

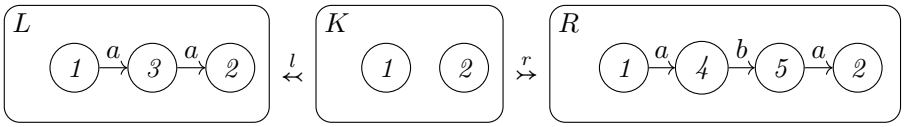
- *there is a context closure* c_ρ *for* ρ *and* $\mathcal{T}$ *in* $\mathfrak{F}$,

- $\mu(w_{\mathcal{T}}(\{l \star - = t_K\})) >_{\mathbb{R}^+} \mu(w_{\mathcal{T}}(\{r \star - = t_K\})) + \delta$ *for* $t_K = l \star c_\rho$.

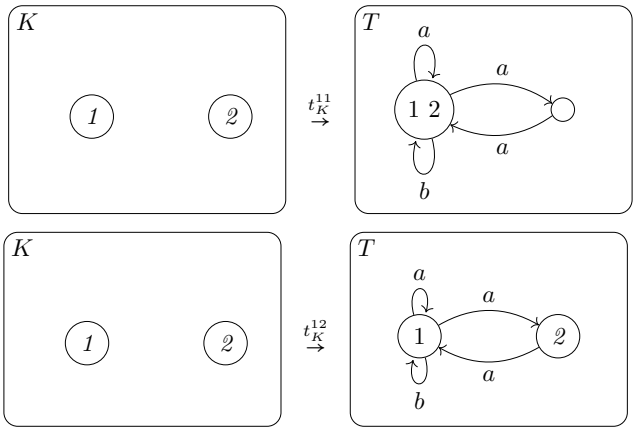
Example 3.39. *Consider the following weighted type graph over the real arithmetic semiring* $\mathfrak{N}' = (\mathbb{R}^+, +, *, 0_{\mathbb{R}}, 1_{\mathbb{R}}, <, \mathrm{id}_{\mathbb{R}^+})$*:*

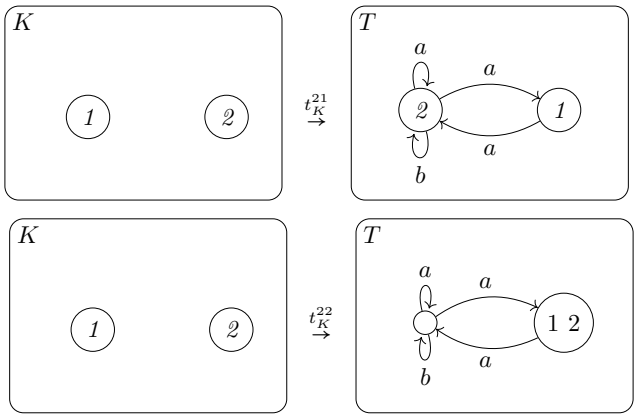


and the rule:

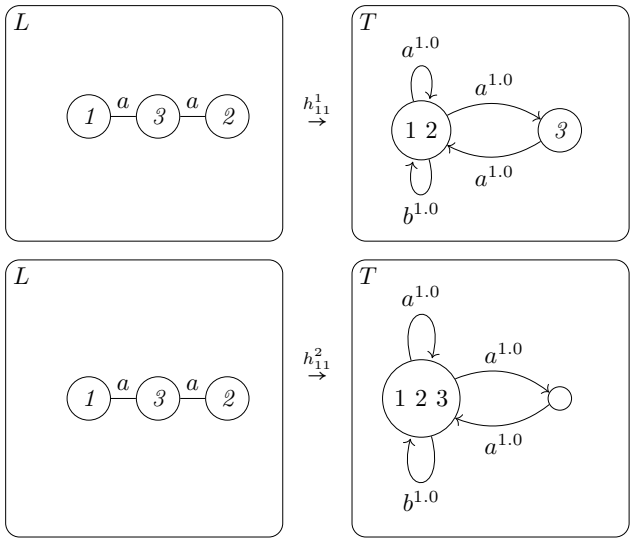


There are 4 morphisms $t_K^{11}, t_K^{12}, t_K^{21}, t_K^{22}$ *from* K *to* T*:*





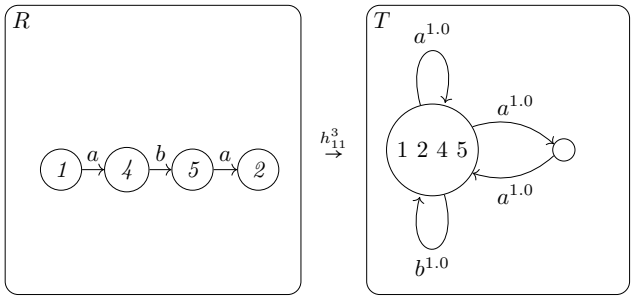
The set $\{l \star - = t_K^{11}\}$ *has two morphisms* h_{11}^1 *and* h_{11}^2*:*



Therefore, we have

$$\begin{aligned} w_{\mathcal{T}}(\{l \star - = t_K^{11}\}) =& w_{\mathcal{T}}(\{h_{11}^1, h_{11}^2\}) \\ =& w_{\mathcal{T}}(h_{11}^1) + w_{\mathcal{T}}(h_{11}^2) \\ =& (1.0^1 * 1.0^1) + (1.0^1 * 1.0^1) \\ =& 2.0. \end{aligned}$$

The set $\{r \star - = t_K^{11}\}$ *has one morphism* h_{11}^3 *which is illustrated below:*



Therefore, we have:

$$w_{\mathcal{T}}(\{r \star -= t_K^{11}\}) = w_{\mathcal{T}}(h_{11}^3) = 1.0^1 * 1.0^1 * 1.0^1 = 1.0.$$

The following inequality holds:

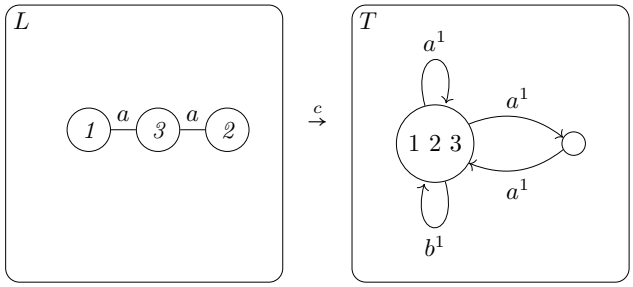
$$w_{\mathcal{T}}(\{l \star -= t_K^{11}\}) = 2.0 \geq 1.0 = w_{\mathcal{T}}(\{r \star -= t_K^{11}\}).$$

Similarly, we can check that

- $w_{\mathcal{T}}(\{l \star -= t_K^{12}\}) = 1.0 \geq 1.0 = w_{\mathcal{T}}(\{r \star -= t_K^{12}\})$, *and*
- $w_{\mathcal{T}}(\{l \star -= t_K^{21}\}) = 1.0 \geq 1.0 = w_{\mathcal{T}}(\{r \star -= t_K^{21}\})$, *and*
- $w_{\mathcal{T}}(\{l \star -= t_K^{22}\}) = 1.0 \geq 1.0 = w_{\mathcal{T}}(\{r \star -= t_K^{22}\})$.

Therefore, the rule is weakly decreasing.

Moreover, there is a context closure c for the DPO rule in the weighted type graph, as explained in Example 3.26, which is illustrated below:



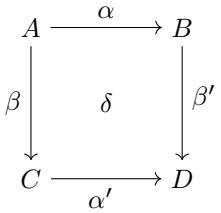
Additionally, we have

- $t_K^{11} = l \star c$, *and*
- $w_{\mathcal{T}}(\{l \star -= t_K^{11}\}) = 2.0 > 1.0 + \delta = w_{\mathcal{T}}(\{r \star -= t_K^{11}\}) + \delta$ *for* $0 < \delta < 1.0$.

Thus, we conclude that the rule is δ-closure decreasing with $0 < \delta < 1.0$ since the semiring is strictly monotonic measurable.

The following lemma by Endrullis and Overbeek [38] guarantees that, under certain constraints: exact weights of host graphs can be computed, and upper bounds for result-graph weights can be derived.

Lemma 3.40. *Let $\mathcal{T} = (T, \mathbb{E}, (S, \oplus, \odot, 0_S, 1_S, <, \mu), w)$ be a finitary weighted type graph. Consider the pushout square δ:*



We define

$$k = \bigoplus_{t_A : A \to T} \left(\bigoplus_{\substack{t_C : C \to T \\ t_A = \beta \star t_C}} w_{\mathcal{T}}(t_C - \beta) \right) \odot w_{\mathcal{T}}(\{\alpha \star - = t_A\})$$

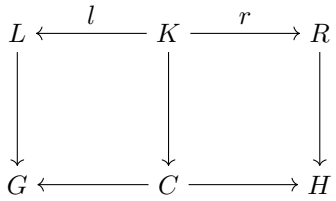
The following conditions hold

(A) $w_{\mathcal{T}}(D) = k$ *if* δ *is weighable with* $\mathcal{T}$*, and*

(B) $w_{\mathcal{T}}(D) \preceq k$ *if* δ *is bounded above by* $\mathcal{T}$ *and* $w(e) \succeq 1_S$ *for all* $e \in \mathbb{E}$*.*

Adapted from [38, Theorem C.3] by Endrullis and Overbeek, the lemma below relates decreasing rules to weight reduction under specific constraints.

Lemma 3.41 (Decreasing steps)**.** *Let* $\mathcal{T} = (T, \mathbb{E}, (S, \oplus, \odot, 0_S, 1_S, \prec, \mu), w)$ *be a finitary weighted type graph,* ρ *a rewriting rule and* $\Delta \in \mathfrak{F}(\rho)$ *the DPO diagram:*



such that the following conditions hold:

- $\mathrm{left}(\Delta)$ *is weighable with* $\mathcal{T}$*, and*
- $\mathrm{right}(\Delta)$ *is bounded above by* $\mathcal{T}$*, and*
- $w(e) \succeq 1_S$ *for all* $e \in \mathbb{E}$*.*

We have:

- $\mu(w_{\mathcal{T}}(G)) \succeq \mu(w_{\mathcal{T}}(H))$ *if* ρ *is weakly decreasing,*
- $\mu(w_{\mathcal{T}}(G)) >_{\mathbb{R}^+} \mu(w_{\mathcal{T}}(H)) + \delta$ *if* ρ *is* δ*-uniformly or* δ*-closure decreasing for some* $\delta >_{\mathbb{R}^+} 0$ *and* $w(e) \succeq 1_S$ *for all* $e \in \mathbb{E}$*.*

Proof. For every $t_K : K \to T$, we define $S_{t_K} \stackrel{\text{def}}{=} \bigoplus_{\substack{t_C : C \to T \\ t_K = h_{KC} \star t_C}} w_{\mathcal{T}}(t_C - h_{KC})$.
For all $t_K : K \to T$ and $X, Y \in S$, the following claims hold:

(a) $S_{t_K} \neq 0_S$ if there is t_C with $t_K = h_{KC} \star t_C$.

Proof. By definition of weighted type graph, for all $e \in \mathbb{E}$, we have

$$w(e) \neq 0_S. \tag{3.1}$$

For every $t_C : C \to T$, we have

$$\begin{aligned} w_{\mathcal{T}}(t_C - h_{KC}) &= \bigodot_{e \in \mathbb{E}} w_e(t_C - h_{KC}) && \text{by Definition 3.17} \\ &= \bigodot_{e \in \mathbb{E}} \bigodot_{\substack{\alpha \in \{-\star t_C = e\} \\ \alpha \notin \{\iota \in \mathrm{Hom}(X,C) \mid \exists \zeta : X \to K,\ \zeta \star h_{KC} = \iota\}}} w(e) && \text{by Definition 3.16} \\ &\neq 0_S. && \text{by (3.1), (S10) and Definition 3.7} \end{aligned}$$

Therefore, $S_{t_K} \overset{\text{def}}{=} \bigoplus_{\substack{t_C : C \to T \\ t_K = h_{KC} \star t_C}} w_{\mathcal{T}}(t_C - h_{KC}) \neq 0_S$ if there is be a morphism such that $t_K = h_{KC} \star t_C$ by Proposition 3.37 (S12). □

(b) $S_{t_K} \geq 1_S$ if there is t_C with $t_K = h_{KC} \star t_C$ and $w_{\mathcal{T}}(e) \geq 1_S$ for all $e \in \mathbb{E}$,

Proof. By the definition of weighted type graph, for all $e \in \mathbb{E}$, we have $w(e) \neq 0_S$. By assumption, we have $w_{\mathcal{T}}(e) \geq 1_S$ for all $e \in \mathbb{E}$. Thus, we have

$$1_S \leq w(e) \neq 0_S. \tag{3.2}$$

By Proposition 3.37 (S11), we have

$$1_S \leq w_{\mathcal{T}}(t_C - h_{KC}). \tag{3.3}$$

Therefore, $S_{t_K} \overset{\text{def}}{=} \bigoplus_{\substack{t_C : C \to T \\ t_K = h_{KC} \star t_C}} w_{\mathcal{T}}(t_C - h_{KC}) \geq 1_S$ if there is be a morphism such that $t_K = h_{KC} \star t_C$ by Definition 3.31 (S1). □

(c) if there is t_C with $t_K = h_{KC} \star t_C$ then

$$\mu(Y) >_{\mathbb{R}^+} \mu(X) + \delta \implies \mu(Y \odot S_{t_K}) >_{\mathbb{R}^+} \mu(X \odot S_{t_K}).$$

Proof. Suppose that there is t_C with $t_K = h_{KC} \star t_C$. We have $S_{t_K} \neq 0_S$ by Claim (a), and we conclude by Definition 3.31 (S7). □

(d) if there is t_C with $t_K = h_{KC} \star t_C$, and $w_{\mathcal{T}}(e) \geq 1_S$ for all $e \in \mathbb{E}$ then

$$\mu(Y) >_{\mathbb{R}^+} \mu(X) + \delta \implies \mu(Y \odot S_{t_K}) >_{\mathbb{R}^+} \mu(X \odot S_{t_K}) + \delta.$$

Proof. Suppose that there is t_C with $t_K = h_{KC} \star t_C$. We have $1_S \leq S_{t_K} \neq 0_S$ by Claim (a) and Claim (b), and we conclude by Definition 3.31 (S6). □

(e) if there is no t_C with $t_K = h_{KC} \star t_C$ then $S_{t_K} = 0_S$, thus

$$Y \odot S_{t_K} = 0_S = X \odot S_{t_K}.$$

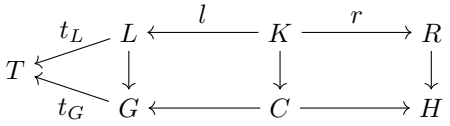
(f) If there is a context closure t_L for ρ and T in $\mathfrak{F}$, then, let $t_K = l \star t_L$, we have

$$\mu(Y) >_{\mathbb{R}^+} \mu(X) + \delta \implies \mu(Y \odot S_{t_K}) >_{\mathbb{R}^+} \mu(X \odot S_{t_K}).$$

Proof. By Definition 3.25 of context closure, there is $t_G : G \to T$ such that

$$t_L = h_{LG} \star t_G. \tag{1}$$

i.e. we have the following commutative diagram:



Let $t_C \stackrel{\text{def}}{=} h_{CG} \star t_G$ and $t_K \stackrel{\text{def}}{=} l \star t_L$. We have

$$\begin{aligned}
t_K &= l \star t_L && \text{by definition of } t_K \\
&= l \star (h_{LG} \star t_G) && \text{by (1) above} \\
&= (l \star h_{LG}) \star t_G && \text{by associativity} \\
&= (h_{KC} \star h_{CG}) \star t_G && \text{by commutativity of } \square KLGC \\
&= h_{KC} \star (h_{CG} \star t_G) && \text{by associativity} \\
&= h_{KC} \star t_C. && \text{by definition of } t_C
\end{aligned}$$

and the claim follows from Claim (c), since t_C is a morphism such that $t_K = h_{KC} \star t_C$. □

(g) If there is a context closure t_L for ρ and T in $\mathfrak{F}$, and $w_{\mathcal{T}}(e) \geq 1_S$ for all $e \in \mathbb{E}$ then, let $t_K = l \star t_L$, we have

$$\mu(Y) >_{\mathbb{R}^+} \mu(X) + \delta \implies \mu(Y \odot S_{t_K}) \underset{\mathbb{R}^+}{>} \mu(X \odot S_{t_K}) + \delta.$$

Proof. The proof is analogous to the proof of Claim (f) but with Claim (d) instead of Claim (c) in the end. □

For every $t_K : K \to T$, let

$$\begin{aligned}
\Lambda_{t_K} &\stackrel{\text{def}}{=} w_{\mathcal{T}}(\{l \star - = t_K\}), \\
\Omega_{t_K} &\stackrel{\text{def}}{=} w_{\mathcal{T}}(\{r \star - = t_K\}).
\end{aligned}$$

By Lemma 3.40, we have

$$\begin{aligned}
w_{\mathcal{T}}(G) &= \bigoplus_{t_K : K \to T} (S_{t_K} \odot \Lambda_{t_K}), \\
w_{\mathcal{T}}(H) &\leq \bigoplus_{t_K : K \to T} (S_{t_K} \odot \Omega_{t_K}).
\end{aligned}$$

We complete the proof of Lemma 3.41 by case analysis:

1. If ρ is weakly decreasing, then, by Definition 3.38 of weakly decreasing rule, we have $\Lambda_{t_K} \geq \Omega_{t_K}$ for every $t_K : K \to T$. By Definition 3.31 (S4), for every $t_K : K \to T$, we have

$$S_{t_K} \odot \Lambda_{t_K} \geq S_{t_K} \odot \Omega_{t_K}. \tag{NE}$$

Thus, we have $w_{\mathcal{T}}(G) \geq w_{\mathcal{T}}(H)$, from Definition 3.31 (S1).

2. Suppose that ρ is δ-uniformly decreasing for $\delta \in \mathbb{R}_{>0}$, and $w_{\mathcal{T}}(e) \geq 1_S$ for all $e \in \mathbb{E}$. By Definition 3.38 of δ-uniformly decreasing rule, for all $t_K : K \to T$, we have

   - $\mu(\Lambda_{t_K}) >_{\mathbb{R}^+} \mu(\Omega_{t_K}) + \delta$, or
   - $\{l \star - = t_K\} = \varnothing = \{r \star - = t_K\}$.

   From Claim (d) and Claim (e), for every $t_K : K \to T$, we obtain

   (i) $S_{t_K} \odot \Lambda_{t_K} = 0_S = S_{t_K} \odot \Omega_{t_K}$, or
   (ii) $\mu(\Lambda_{t_K} \odot S_{t_K}) >_{\mathbb{R}^+} \mu(\Omega_{t_K} \odot S_{t_K}) + \delta$.

   To establish $\mu(w_{\mathcal{T}}(G)) >_{\mathbb{R}^+} \mu(w_{\mathcal{T}}(H)) + \delta$, using Definition 3.31 (S3), it suffices to show that we have Case (ii) for some $t_K : K \to T$. This follows from Claim (g) since we have a context closure for ρ and $\mathcal{T}$ by assumption.

3. If ρ is δ-closure decreasing, and $w_{\mathcal{T}}(e) \geq 1_S$ for all $e \in \mathbb{E}$ then it is also weakly decreasing and we obtain Inequality (NE) for every $t_K : K \to T$. Since the semiring is a strictly monotonic museurable semiring, by Definition 3.31 (S3) and Definition 3.31 (S9), it suffices to show that there is some $t_K : K \to T$ such that

$$\mu(S_{t_K} \odot \Lambda_{t_K}) >_{\mathbb{R}^+} \mu(S_{t_K} \odot \Omega_{t_K}) + \delta. \qquad (\star\star)$$

   in order to conclude $\mu(w_{\mathcal{T}}(G)) >_{\mathbb{R}^+} \mu(w_{\mathcal{T}}(H)) + \delta$. There is a context closure t_L for ρ and T, and $\mu(\Lambda_{t_K}) >_{\mathbb{R}^+} \mu(\Omega_{t_K}) + \delta$ for $t_K = l \star t_L$. Thus, we obtain Inequality $(\star\star)$ by Claim (g).

□

3.2.4 Termination Criterion

Finally, we formulate a termination criterion leveraging type graphs weighted over non-well-founded semirings.

Theorem 3.42 (Termination of DPO rewriting system)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be sets of DPO rewriting rules, $\mathcal{T} = (T, \mathbb{E}, (S, \oplus, \odot, 0_S, 1_S, <, \mu), w)$ a finitary weighted type graph and $\mathfrak{F}$ a DPO rewriting framework such that*

i) $w(e) \geq 1_S$ for all $e \in \mathbb{E}$, and

ii) for all $x \in S$, if $1_S \leq x \neq 0_S$ then $\mu(x) \geq \mu(1_S)$ and $\mu(x) \in \mathbb{R}$, and

iii) for every rule $\rho \in (\mathcal{A} \cup \mathcal{B})$ and every double pushout diagram $\Delta \in \mathfrak{F}(\rho)$

   - $\mathrm{left}(\Delta)$ *is weighable with $\mathcal{T}$,*
   - $\mathrm{right}(\Delta)$ *is bounded above by $\mathcal{T}$.*

If the following conditions hold:

1. *there is $\delta >_{\mathbb{R}^+} 0$ such that either every $\rho \in \mathcal{A}$ is δ-uniformly decreasing, or every $\rho \in \mathcal{A}$ is δ-closure decreasing, and*

2. *every rule $\rho \in \mathcal{B}$ is weakly decreasing,*

then $\Rightarrow_{\mathcal{A},\mathfrak{F}}$ *terminates relative to* $\Rightarrow_{\mathcal{B},\mathfrak{F}}$.

Proof. From the definition of weighted type graph, we have

$$\text{for all } e \in \mathbb{E}, w(e) \neq 0_S.$$

From i), we have

$$\text{for all } e \in \mathbb{E}, 1_S \leq w(e).$$

Therefore, we have

$$\text{for all } e \in \mathbb{E}, 1_S \leq w(e) \neq 0_S. \tag{3.4}$$

Let G be a graph admitting a match of a DPO rewriting rule. We have

$$\begin{aligned} w_{\mathcal{T}}(G) &\overset{\text{def}}{=} \bigoplus_{h \in \mathrm{Hom}(G,T)} w_{\mathcal{T}}(h) \\ &\overset{\text{def}}{=} \bigoplus_{h \in \mathrm{Hom}(G,T)} \left(\bigodot_{e \in \mathbb{E}} \left(\bigodot_{\alpha \in \{- \star h = e\}} w(e) \right) \right). \end{aligned}$$

By (3.4), Proposition 3.37, Definition 3.7 and $1_S \neq 0_S$, for every $h \in \mathrm{Hom}(G,T)$, we have

$$1_S \leq \bigodot_{e \in \mathbb{E}} \left(\bigodot_{\alpha \in \{- \star h = e\}} w(e) \right) \neq 0_S. \tag{3.5}$$

Since G is a graph admitting a match of a DPO rewriting rule, by Proposition 3.37 and Definition 3.31 (S0), we have

$$1_S \leq w_{\mathcal{T}}(G) \neq 0_S$$

By Assumption ii), we have

$$\mu(w_{\mathcal{T}}(G)) \geq_{\mathbb{R}^+} \mu(1_S).$$

By Lemma 3.41, every rewriting step with rules in $\mathcal{A}$ strictly decreases the weight by at least δ and no rewriting step with rules in $\mathcal{B}$ increases the weight. Consequently, there is no infinite rewriting chain with an infinite rewriting steps with rules in $\mathcal{A}$ from G. □

Example 3.43. *Termination of the DPO rule in Example 2.29 can be established using Theorem 3.42 together with the weighted type graph in Example 3.36 over the real arithmetic semiring* $\mathfrak{N}' = (\mathbb{R}^+, +, \star, 0_{\mathbb{R}}, 1_{\mathbb{R}}, <, \mathrm{id}_{\mathbb{R}^+})$. *It is* δ*-closure decreasing with* $\delta = 0.5$ *as explained in Example 3.39.*

3.2.5 Implementation

Chapter 6 provides a description of the implementation of our approach and the one proposed by Endrullis and Overbeek [38] in our tool, called LyonParallel and described in Section 6.

Searching Strategy. Consider a DPO rewriting system $\mathcal{R}$. Given a fixed semiring S, a processor is spawned to iteratively search for a suitable weighted type graph over S. The search begins with a weighted type graph containing 1 node and increases the node count by 1 until it reaches a maximum of 4 nodes.

For a fixed number of nodes, if S is a semiring over the natural numbers, the maximum edge weight is initialized to 1 and incremented by 1 (up to a limit of 3) if no suitable weighted type graph is found; if S is a semiring over the real numbers, weights are constrained to be positive real numbers, and no upper bound is imposed. Processors targeting different semirings can run in parallel to analyze the same DPO rewriting system. When a processor discovers a weighted type graph that witnesses relative termination of a subset of rules $\mathcal{A}$ with respect to another subset of rules $\mathcal{B}$ such that $\mathcal{R} = \mathcal{A} \cup \mathcal{B}$, it broadcasts this result to all processors and waits for them to terminate. If $\mathcal{B} = \varnothing$, the system's termination is proven. Otherwise, we initiate a new search for the rules in $\mathcal{B}$ unless a timeout is reached.

For a fixed number of nodes k, we adopt the approach proposed in [17, 19, 85] to reduce the search space. Specifically, we construct a weighted type graph with k nodes and no parallel edges of the same label. The search proceeds as follows:

1. decide if $s \xrightarrow{l} t$ exists for every pair of nodes s, t and label l;
2. assign a weight to every existing edge;
3. verify the existence of a constant $\delta > 0$ and a partition of the rule set $\mathcal{R}$ into a non-empty subset $\mathcal{A}$ and a subset $\mathcal{B}$ such that:
   - either all rules in $\mathcal{A}$ are δ-uniformly decreasing or all rules in $\mathcal{A}$ are δ-closure decreasing, and
   - all rules in $\mathcal{B}$ are weakly decreasing.

This procedure amounts to checking the satisfiability of an existential Presburger arithmetic, Peano arithmetic, linear real arithmetic, or non-linear real arithmetic formula depending on the semiring S considered.

Z3 Modeling. The type graph T is modeled in Z3 by defining a boolean variable $x_{u,v,l} \in \mathbb{B}$ for every directed labeled edge $u \xrightarrow{l} v$ in the type graph, where $u, v \in \{1, ..., k\}$ are nodes and $l \in \Sigma$ is an edge label. The variable $x_{u,v,l}$ has the value *true* if the directed edge $u \xrightarrow{l} v$ exists in the result type graph, *false* otherwise.

The weight function w and the set $\mathbb{E}$ of morphism-rulers are modeled by defining a variable $y_{u,v,l}$ of Z3's *Real* sort or *Int* sort (the solver's default theory for real numbers and integers), depending on the semiring $\mathcal{S}$ considered, for every pair $u, v \in \{1, ..., k\}$ of nodes and edge label $l \in \Sigma$. The variable $y_{u,v,l}$ represents the weight of the directed labeled edge $u \xrightarrow{l} v$ in the resulting weighted type graph, but only if edge $u \xrightarrow{l} v$ exists (i.e. $x_{u,v,l}$ has the value *true*). They are constrained to be non-negative.

A variable $\delta \in \mathbb{R}_{>0}$ is defined to ensure that there is a partition of the rule set $\mathcal{R}$ into a non-empty subset $\mathcal{A}$ and a subset $\mathcal{B}$ such that all rules in $\mathcal{A}$ are either

δ-uniformly decreasing or δ-closure decreasing, and all rules in $\mathcal{B}$ are weakly decreasing.

For convenience, the following auxiliary variables are also defined:

- a boolean variable $v_h \in \mathbb{B}$ for every morphism h from L or R to T;
- a real-valued variable $v_{h'} \in \mathbb{R}_{\geq 0}$ for every morphism h from L or R to T;
- a boolean variable $v_c \in \mathbb{B}$ for every morphism c from L to T.

The value of v_h has the value *true* if the morphism h exists in the result type graph (i.e. all edges in its image exist), and *false* otherwise. The variable $v_{h'}$ holds the weight of the morphism h, provided the morphism h exists (i.e., v_h has the value *true*). The variable v_c has the value *true* if the morphism c exists in the result type graph and can serve as a context closure for the rule, and *false* otherwise.

If there is an assignment of values to

- $(x_{u,v,l})$, for all $u, v \in \{1, ..., k\}$ and for all $l \in \Sigma$, and
- $(y_{u,v,l})$, for all $u, v \in \{1, ..., k\}$ and for all $l \in \Sigma$, and
- δ,

such that all conditions of Theorem 3.42 are satisfied, then a suitable weighted type graph that witnesses termination of the DPO rewriting system can be constructed, because

- the values of $(x_{u,v,l})_{u,v \in \{1,...,k\}, l \in \Sigma}$ define the type graph T, and
- the values of $(y_{u,v,l})_{u,v \in \{1,...,k\}, l \in \Sigma}$ define the weight function w and the set $\mathbb{E}$ of morphism-rulers.

3.2.6 Empirical Results

We evaluated our OCaml implementation using 13 examples from [38, 72, 71, 17, 19], excluding [38, Example 6.4] which is a DPO rewriting system in an unsupported category, and [19, Example 6] because it duplicates [17, Example 4]. Our approach with semirings over the real numbers proves termination for 8 out of 13 systems tested. Table 3.1 summarizes runtime performance across benchmark examples under different semiring configurations [4].

3.2.7 Discussion

Acceleration with the real tropical / arctic semiring. Consider the examples in Table 3.1 whose termination is provable using the tropical or arctic semirings. For all such examples except [71, Example 5] and [17, Example 4], the real-number semiring approach achieves better runtime performance than the natural-number semiring approach. The acceleration rate varies across examples due to Z3's internal heuristics. This result is notable because termination proofs for these systems only require weighted type graphs with small edge weights (e.g., values less than 2 for most examples and at most 3 for all), which theoretically favors the implementation of the natural-number semiring approach.

[4] Experiments were conducted on a laptop equipped with an i5-1038NG7 CPU, which features 4 cores, a base clock speed of 2 GHz, a boost speed of 3.8 GHz, and 16 GB RAM.

	A	a	T	t	N	n
[38, Example 6.2]					2.68	1.15
[38, Example 6.3]					2.74	1.16
[38, Example D.3]	2.25	1.18			2.24	1.18
[72, Example 3.8]	2.95	1.90	2.94	1.87	3.49	1.87
[71, Example 4]	4.26	3.19	4.24	3.13	5.82	timeout
[71, Example 5]	5.54	5.55	5.53	5.50	9.11	5.62
[71, Example 6]						
[17, Example 4]	2.44	2.46	2.47	2.54	4.58	2.46
[17, Example 5]					7.80	timeout
[17, Example 6]					9.75	timeout
[19, Example 1]	2.26	1.18			2.24	1.18
[19, Example 4]	2.25	1.22	2.24	1.18	2.25	1.19
[19, Example 5]	4.23	3.23	4.25	3.28	5.82	timeout

Table 3.1: Runtime performance in seconds (200s timeout). Columns denote semirings: "A", "T", "N" denote the arctic, tropical, and arithmetic semirings over the (extended) natural numbers; "a", "t", "n" denote the arctic, tropical, and arithmetic semirings over the (extended) real numbers. Empty cells indicate termination not proven.

Timeout with the real arithmetic semiring. The systems whose termination is not proven by our tool are [71, Example 6], [17, Examples 5 and 6], [71, Example 4], and [19, Example 5]. The first case ([71, Example 6]) is due to a limitation of the type graph method in general, as the system has a rule whose right-hand side graph can be embedded into its left-hand side graph, which is problematic for the type graph method as pointed out in [38, Example D.4]. For the remaining cases, the failure stems from the double-exponential time complexity [22, 48] in the number of variables $(x_{u,v,l})_{u,v\in\{1,\dots,k\},l\in\Sigma}$ and $(y_{u,v,l})_{u,v\in\{1,\dots,k\},l\in\Sigma}$, which is also influenced by the length of the conditions of Theorem 3.42 encoded in Z3.

Acceleration with the real arithmetic semiring. For some examples, the approach with semirings on real numbers can achieve a better runtime performance than the approach with semirings on natural numbers. These are (1) examples which need weighted type graphs with 1 or 2 nodes, and the constraints can be expressed in Z3 with a short formula, or (2) examples which can first be simplified by eliminating some rules using weighted type graphs with 1 node, and then the remaining rules can be easily shown to terminate as the first case.

Interest of the approach with semiring on natural numbers. For [71, Example 4], [17, Example 5 and 6] and [19, Example 5], the semiring on natural numbers is more efficient, as it allows one to restrict the search space to an extremely small size by bounding the maximum weight of edges, which is not

possible for the semiring on real numbers. However, this advantage can be achieved only if the termination can be proven with a weighted type graph with whose weights are in a extremely small set.

Limitations of our experiments. While our search strategy is very close to the real-world scenario that a non-expert user would use to prove termination of DPO rewriting systems, it makes it difficult to estimate the runtime performance gain of our approach over the one proposed by Endrullis and Overbeek [38]. This is because the reported runtimes include (1) translation of constraints into Z3's input format, (2) the time spent in Z3 to solve the constraint system, and (3) the parallelism overhead. Another problem is that the examples in our experiments are limited to those whose termination can be proven with a weighted type graph whose edge weights are small (at most 3). These examples favor the approach with semirings on natural numbers, as the search space is extremely small when the maximum edge weight is bounded by 3.

Comparison with `GraphTT-wtg`. A comparison of our implementation of the approach with semirings on real numbers with the one proposed in `GraphTT-wtg` by Endrullis and Overbeek [40] is desired. However, at the time of writing and to our knowledge, this tool is not publicly available, and their article does not provide sufficient implementation details to enable a detailed comparison.

3.3 Conclusion

The type graph method is a technique for proving termination of DPO rewriting systems. To apply this method, one needs to construct a weighted type graph over a semiring that witnesses termination of the rewriting system, and three concrete well-founded semirings over the natural numbers have been proposed in prior work: the tropical semiring, the arctic semiring, and the arithmetic semiring. However, constructing suitable weighted type graphs over these concrete semirings for DPO rewriting systems on edge-labeled directed multigraphs is difficult. To address this challenge, we investigate weighted type graphs over non-well-founded semirings, and propose three corresponding concrete semirings over the real numbers: the tropical semiring, the arctic semiring, and the arithmetic semiring. Construction of weighted type graphs over these concrete semirings is computationally easier.

We implemented both our approach and the one proposed by Endrullis et al. [38] for DPO rewriting systems on edge-labeled directed multigraphs into a unified tool. Experiments show that for the tropical and arctic semirings, the weighted type graphs over the real semiring can be constructed more easily, but for the arithmetic semiring, the weighted type graphs over the real semiring are more difficult to construct than those over the natural semiring.

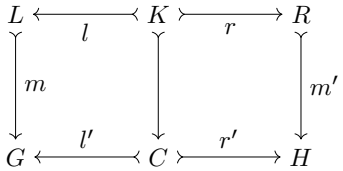
Chapter 4

Termination of Injective DPO Graph Rewriting Systems using Morphism Counting

We present a new automated method for relative termination of double-pushout (DPO) graph rewriting systems with injective rules [30, 43]. [1] To prove the relative termination of a set of rules $\mathcal{A}$ with respect to another set of rules $\mathcal{B}$, the method requires identifying patterns whose counts decrease with each application of a rule from $\mathcal{A}$ and do not increase with each application of a rule from $\mathcal{B}$. Left-hand sides of the rules are typical examples of such patterns, and we leave it to future work to explore whether other patterns are useful.

Let $\mathbb{X}$ denote a fixed set of patterns. We assign weights (natural numbers) to monomorphisms from graphs in $\mathbb{X}$, and define the weight of a graph G as the sum of the weights of all monomorphisms from graphs in $\mathbb{X}$ to G.

Let $\mathcal{A}$ and $\mathcal{B}$ be sets of DPO rewriting rules. Let $X \in \mathbb{X}$ and $G \Rightarrow H$ be a rewriting step defined by the double-pushouts diagram shown below.



The image of a monomorphism from X to G falls into exactly one of the following three mutually exclusive cases:

1. the image is fully included in $m(L)$;
2. the image is fully included in $l'(C)$;
3. the image is neither fully included in $m(L)$ nor fully included in $l'(C)$.

Similary, the image of a monomorphism from X to H falls into exactly one of the following three mutually exclusive cases:

[1] This work resulted in a publication at the 18th International Conference on Graph Transformation (ICGT 2025) [78]. Reproduced with permission from Springer Nature.

1. the image is fully included in $m'(R)$;
2. the image is fully included in $r'(C)$;
3. the image is neither fully included in $m'(R)$ nor fully included in $r'(C)$.

The number of monomorphisms $h : X \to G$ whose image is fully included in $m(L)$ can be computed exactly; likewise, the number of monomorphisms $x : X \to H$ whose image is fully included in $m'(R)$ can be computed exactly. Moreover, the number of monomorphisms $x : X \to G$ whose image is fully included in $l'(C)$ equals to the number of monomorphisms $x : X \to H$ whose image is fully included in $r'(C)$. Therefore, to estimate the change in weight when applying a rewriting rule, it suffices to compare the numbers of monomorphisms in the third cases above.

To enable this comparison, we suppose that for every $X \in \mathbb{X}$, the following conditions hold:

- For every rule in $\mathcal{A}$, the number of monomorphisms $x : X \to G$ whose image is fully included in $m(L)$ is strictly greater than the number of monomorphisms $x : X \to H$ whose image is fully included in $m'(R)$;
- For every rule in $\mathcal{B}$, the number of monomorphisms $x : X \to G$ whose image is fully included in $m(L)$ is greater than or equal to the number of monomorphisms $x : X \to H$ whose image is fully included in $m'(R)$;

We suppose additionaly that there is an injective mapping from the set of monomorphisms $x : X \to H$ whose the image is neither fully included in $m'(R)$ nor fully included in $r'(C)$ to the set of monomorphisms from $x : X \to G$ whose image is neither fully included in $m(L)$ nor fully included in $l'(C)$.

Under these conditions, rewriting steps using rules in $\mathcal{A}$ strictly decrease the weights of the host graphs, while rewriting steps using rules in $\mathcal{B}$ do not increase them. Consequently, the rewriting rules in $\mathcal{A}$ can only be applied a finite number of times.

This chapter is organized as follows. We first define the weight of a graph in § 4.2. Using this definition, we outline the general approach and identify the key challenge in § 4.3 and introduce the concept of non-increasing rules in § 4.4. A solution to the challenge is then proposed in § 4.5, culminating in our termination criterion in § 4.6. § 4.7 illustrates the method with some examples. § 4.8 compares our approach with some existing methods. Finally, § 4.9 concludes with remarks and future research directions. Proofs of some propositions, lemmas, and theorems of this chapter have been moved to Appendix 4.10 to improve readability.

4.1 Preliminaries

4.1.1 DPO rewriting with injective rules and matches

While Definition 4.1 of injective rewriting rules and rewriting steps may appear more restrictive, they are equivalent to those introduced in § 2.3. This equivalence arises because algebraic graph rewriting via double pushouts is defined up to isomorphism. Our preference for these definitions stems from their practical

advantage: they provide a clearer view of how the number of monomorphisms from specific graphs changes after a rewriting step.

Definition 4.1. *An **inclusion** is a function f such that $\mathrm{dom}(f) \subseteq \mathrm{codom}(f)$ and $f(x) = x$ for every $x \in \mathrm{dom}(f)$. A rewriting rule $\varphi = (L \overset{l}{\hookleftarrow} K \overset{r}{\hookrightarrow} R)$ consists of two inclusions l and r. Two rules $\varphi' = (L' \overset{l'}{\hookleftarrow} K' \overset{r'}{\hookrightarrow} R')$ and φ are **equivalent** if there are isomorphisms a, b, c such that $l' \star a = b \star l$ and $r' \star c = b \star r$, i.e. the following diagram is commutative:*

$$\begin{array}{ccccc} L' & \xhookleftarrow{l'} & K' & \xhookrightarrow{r'} & R' \\ \downarrow{\scriptstyle a} & = & \downarrow{\scriptstyle b} & = & \downarrow{\scriptstyle c} \\ L & \xhookleftarrow{l} & K & \xhookrightarrow{r} & R \end{array}$$

A rewriting step $G \Rightarrow_\varphi H$ is a double-pushout:

$$\begin{array}{ccccc} L' & \xhookleftarrow{l'} & K' & \xhookrightarrow{r'} & R' \\ \downarrow & \mathrm{PO} & \downarrow & \mathrm{PO} & \downarrow \\ G & \xhookleftarrow{} & C & \xhookrightarrow{} & H \end{array}$$

where all arrows are inclusions and $L' \overset{l'}{\hookleftarrow} K' \overset{r'}{\hookrightarrow} R'$ is equivalent to φ.

4.1.2 pregraphs

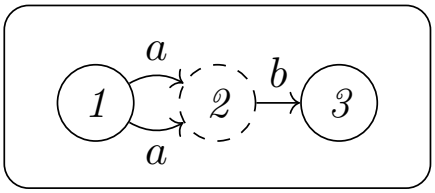
Graphs in double-pushouts that define rewriting steps in **Graph** can be decomposed into unions of pregraphs. pregraphs model finite directed edge-labeled multigraphs that may contain dangling edges.

Definition 4.2. *A **pregraph** G consists of*

- *a finite set of nodes, denoted by $V(G)$,*
- *a subset $V_{\nexists}(G) \subseteq V(G)$ of missing nodes,*
- *a finite set of edges, denoted by $E(G)$,*
- *a source function, denoted by $\mathbf{src}_G$, (resp. a target function, denoted by $\mathbf{dst}_G$) that assigns to each edge an existing or missing node,*
- *an edge-labeling function, denoted by l_G, that assigns labels to edges.*

The set of existing nodes, $V(G) \setminus V_{\nexists}(G)$, is denoted by $V_{\exists}(G)$.

Example 4.3. *A pregraph with node-set $\{1, 2, 3\}$, a missing node 2, and 3 dangling edges is illustrated below:*



A homomorphism of pregraphs is a mapping between two pregraphs that preserves structure and respects missing and existing nodes.

Definition 4.4. *Let G and H be pregraphs. A **homomorphism of pregraphs** $h : G \to H$ consists a pair of functions $h_V : V(G) \to V(H)$ and $h_E : E(G) \to E(H)$ such that*

- *for every edge $a : s \to t$ in G, the edge $h_E(a)$ in H has source $h_V(s)$ and target $h_V(t)$,*
- *for all $s \in V_{\nexists}(G)$, we have $h_V(s) \in V_{\nexists}(H)$,*
- *for all $t \in V_{\exists}(G)$, we have $h_V(t) \in V_{\exists}(H)$.*

Definition 4.5. *Let $f : (G, \lambda) \to (G', \lambda')$ and $g : (G', \lambda') \to (G'', \lambda'')$ be homomorphisms of pregraphs. Their composition, denoted by $g \circ f$, is the homomorphism defined by the component-wise composition of functions $g_V \circ f_V$ and $g_E \circ f_E$.*

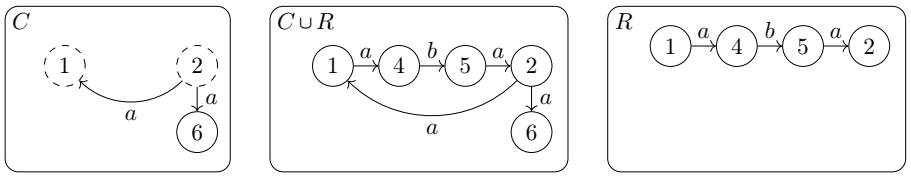
Definition 4.6. *Let G be a pregraph. A pregraph H is a **subpregraph** of G, written $H \rightarrowtail G$, if all of the following conditions hold:*

- *$V(H) \subseteq V(G)$,*
- *$V_{\nexists}(H) = V(H) \cap V_{\nexists}(G)$,*
- *$E(H) \subseteq E(G)$,*
- *the source and target functions of H are the restrictions of those of G to $E(H)$,*
- *the edge-labeling function of H is the restriction $l_H = l_{G|_{E(H)}}$.*

We define two operations on pregraphs that will be used in the following discussion. The **union** $C \cup R$ of two subpregraphs $C \subseteq G$ and $R \subseteq G$ is a pregraph with

- Node set: $V(C) \cup V(R)$,
- Missing nodes: $(V_{\nexists}(C) \cup V_{\nexists}(R)) \setminus (V_{\exists}(C) \cup V_{\exists}(R))$
- Edge set: $E(C) \cup E(R)$,
- Source function: $\mathrm{src}_C \cup \mathrm{src}_R$,
- Target function: $\mathrm{dst}_C \cup \mathrm{dst}_R$,
- Edge-labeling function: $l_C \cup l_R$.

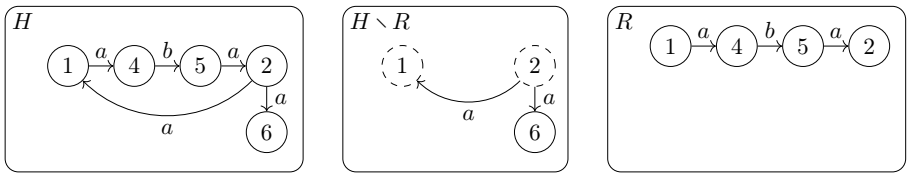
The definition is illustrated below:



The **relative complement** of R in H where $R \subseteq H$, denoted by $H \setminus R$, is the pregraph with

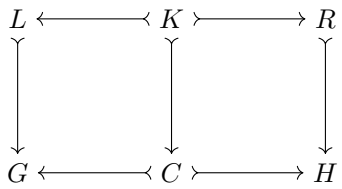
- Edge set: $X \stackrel{\text{def}}{=} E(H) \setminus E(R)$,
- Node set: $Y \stackrel{\text{def}}{=} \bigcup_{e \in X} \{\text{src}(e), \text{dst}(e)\}$,
- Set of missing nodes: $(V_{\nexists}(H) \cup (V_{\exists}(H) \cap V_{\exists}(R))) \cap Y$,
- Source function: src_H restricted to X,
- Target function: dst_H restricted to X,
- Edge-labeling function : l_H restricted to X.

An example is shown below where the graph H, R and the relative complement $H \setminus R$ are illustrated:

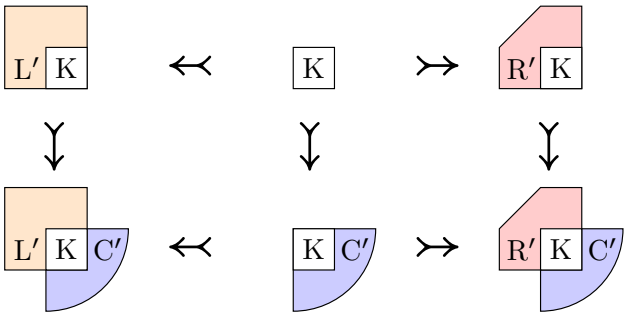


4.1.3 X-occurrences

Consider the following double pushout diagram in the category **Graph**, where all arrows are inclusions:



The graphs in the double pushout above can be decomposed as unions of pregraphs as illustrated below, where K, L', R', C' are pregraphs and $L = L' \cup K$, $R = R' \cup K$, $C = C' \cup K$, $G = L' \cup K \cup C'$, $H = R' \cup K \cup C'$.



Definition 4.7. *Let X be a subgraph of L. An X-**occurrence** in a graph G is a monomorphism from X to G. An X-occurrence in G is*

- ***shared** if its image is included in C,*
- ***explicit** if its image is included in L,*
- ***implicit** if its image has elements in both C' and L'.*

An X-occurrence in H is

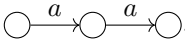
- ***shared** if its image is included in C,*
- ***explicit** if its image is included in R,*
- ***implicit** if its image has elements in both C' and R'.*

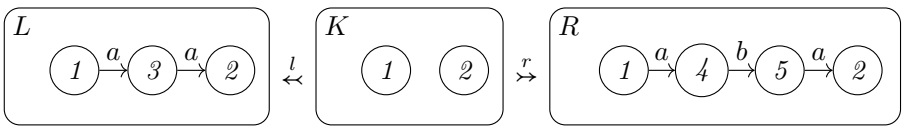
Intuitively, the rewriting step implicitly creates the implicit X-occurrences in H and implicitly destroys implicit X-occurrences in G.

Given an implicit X-occurrence x in H, its image can be decomposed as the union $R_x \cup C_x$, where $R_x = \mathrm{Im}(x) \cap R$ (the part of the image overlapping the right-hand side R), and $C_x = \mathrm{Im}(x) \cap C$ (the part overlapping the context C).

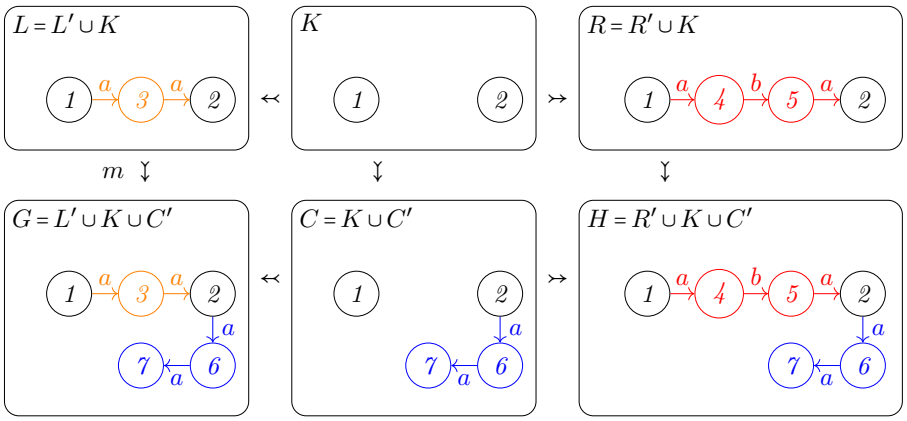
*For every graph monomorphism $h : R_x \rightarrowtail L$ that preserves interface elements (i.e., $h(k) = k$ for all $k \in K$), we define the **corresponding X-occurrence of x relative to** h as the X-occurrence y in G given by:*

$$y(z) = \begin{cases} z & \text{if } z \in x^{-1}(C_x), \\ h(x(z)) & \text{if } z \in x^{-1}(R_x \setminus C_x). \end{cases}$$

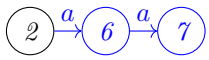
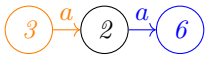
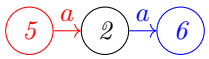
Example 4.8. *Let X be the graph*  *. Consider the X-occurrences in the decomposition of a rewriting step using the rule in Example 2.29 and recalled below:*

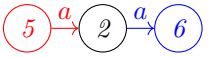
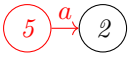
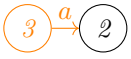
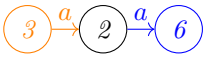


We illustrate the aforementioned decomposition using the visual notation for graph homomorphisms introduced in Notation 2.4, with distinct pregraphs highlighted in different colors.



In this example, the graph X has 4 distinct X-occurrences in G and H. There are

- *1 occurrence shared by G and H:* 
- *1 explicit occurrence in G:* 
- *1 implicit occurrences in G:* 
- *0 explicit occurrence in H,*
- *1 implicit occurrence in H:* 

The corresponding X-occurrence of the implicit occurrences  *in H relative to the unique monomorphism from*  *to*  *is the implicit occurrences*  *in G.*

4.2 Measuring graphs by counting morphisms from specific graphs

Let $\mathcal{R} = \mathcal{A} \cup \mathcal{B}$ be a set of injective rules. Define $\mathrm{Sub}(\mathrm{left}(\mathcal{R}))$ as the set of all subgraphs of left-hand sides of rules in $\mathcal{R}$. We call the elements of $\mathrm{Sub}(\mathrm{left}(\mathcal{R}))$ **ruler-graphs**. Ruler-graphs in a set $\mathbb{X}$ provide a measure for a graph G, and the total weight of G is determined by a weighted linear combination of these measures.

Definition 4.9. *For a ruler-graph X and a graph G, the **measurement** $m_X(G)$ is defined as the number of X-occurrences in G, i.e. the cardinality of the set of monomorphisms from X to G: $m_X(G) = |\mathrm{Mono}(X, G)|$.*

Counting monomorphisms from ruler-graphs before and after a rewriting step provides a termination criterion: If there is a ruler-graph X such that every rewriting step using a rule in $\mathcal{A}$ strictly decreases the number monomorphism

from X, and every rewriting step using a rule in $\mathcal{B}$ never increases it, then the rewriting relation induced by $\mathcal{A}$ terminates relative to the rewriting relation induced by $\mathcal{B}$.

For some ruler-graphs, the change in the number of monomorphism when applying a rewriting step using an injective rule is precisely computable by simply counting the numbers of monomorphisms from these ruler-graphs in the left- and right-hand side graphs.

Examples include:

- the ruler-graph $\bigcirc$ consisting of a single node;
- the ruler-graph $\bigcirc\!\!\circlearrowright x$ consisting of a single node and a single loop labeled by $x \in \Sigma$;
- the ruler-graph $\bigcirc \xrightarrow{x} \bigcirc$ consisting of two nodes and a single labeled edge between them.

However, systems like Example 4.16 require more complex ruler-graphs such as $\bigcirc \xrightarrow{a} \bigcirc \xrightarrow{a} \bigcirc$, because neither the number of nodes nor the number of edges decreases in every rewriting step. For such ruler-graph, the change in the number of monomorphisms when applying a rewriting rule is unpredictable in general, because the number of monomorphisms depends on the context graph—a challenge that we address in later sections.

To combine measurements from multiple ruler-graphs (analogous to combining measurements of physical objects from multiple rulers), we introduce a weight function that assigns a weight to each ruler-graph.

Definition 4.10. *A **weight function** for a set of ruler-graphs $\mathbb{X}$ is a map $s_{\mathbb{X}} : \mathbb{X} \to \mathbb{N}$ assigning a weight in $\mathbb{N}$ to each ruler-graph. The weight of a monomorphism from a ruler-graph X is defined to be $s_{\mathbb{X}}(X)$.*

Assigning different weights to measurements from distinct ruler-graphs enhences the technique, as we will demonstrate in Example 4.29.

Definition 4.11. *Let $\mathbb{X}$ be a set of ruler-graphs, $s_{\mathbb{X}}$ a weight function, and G a graph. The **weight of graph** G relative to $s_{\mathbb{X}}$, denoted by $w_{s_{\mathbb{X}}}(G)$, is defined as:*

$$w_{s_{\mathbb{X}}}(G) = \sum_{X \in \mathbb{X}} s_{\mathbb{X}}(X) \cdot m_X(G).$$

4.3 General idea and key challenge

For the remainder of this section, we assume: a set $\mathcal{R} = \mathcal{A} \cup \mathcal{B}$ of injective DPO rewriting rules, a set $\mathbb{X}$ of ruler-graphs, and a weight function $s_{\mathbb{X}}$. Let $\rho = (L \overset{l}{\hookleftarrow} K \overset{r}{\hookrightarrow} R)$ denote a rule in $\mathcal{R}$ and $X \in \mathbb{X}$. Recall that $\mathfrak{M}$, defined in Definition 2.32, denotes the DPO rewriting framework that associates each rule with the collection of all DPO diagrams witnessing rewriting steps using the rule, where the matches are monic.

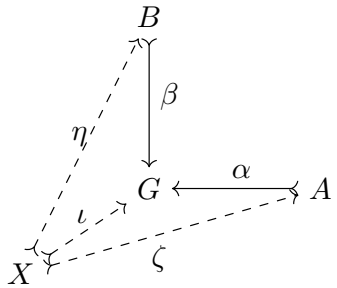
The rewriting relation $\Rightarrow_{\mathcal{A},\mathfrak{M}}$ terminates relative to $\Rightarrow_{\mathcal{B},\mathfrak{M}}$ if the following conditions hold:

- for all $G \Rightarrow_{\mathcal{A},\mathfrak{M}} H$, $w_{s_{\mathbb{X}}}(G) > w_{s_{\mathbb{X}}}(H)$, and

- for all $G \Rightarrow_{\mathcal{B},\mathfrak{M}} H$, $w_{s_{\mathbb{X}}}(G) \geq w_{s_{\mathbb{X}}}(H)$.

However, directly verifying all rewriting steps is infeasible due to their potential infiniteness. Thus, we aim to provide a rule-based sufficient condition for termination.

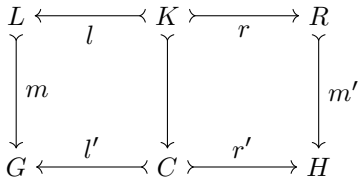
Notation 4.12. *The disjoint union of two sets S and S' will be denoted by $S \uplus S'$. Let X, A, B, G be graphs, and let $\alpha : A \to G$ and $\beta : B \to G$ be morphisms shown below:*



We define the following sets of monomorphisms from X in G based on their interaction with α and β:

$$\begin{aligned}
\mathrm{Mono}(X,G,\alpha) &= \{\iota : X \rightarrowtail G \mid \exists \zeta : X \rightarrowtail A.\, \iota = \zeta \star \alpha\}, \\
\mathrm{Mono}(X,G,\neg\alpha) &= \{\iota : X \rightarrowtail G \mid \nexists \zeta : X \rightarrowtail A.\, \iota = \zeta \star \alpha\}, \\
\mathrm{Mono}(X,G,\neg\alpha,\beta) &= \left\{ \iota : X \rightarrowtail G \;\middle|\; \begin{array}{l} (\nexists \zeta : X \rightarrowtail A.\, \iota = \zeta \star \alpha) \\ \wedge\, (\exists \eta : X \rightarrowtail B.\, \iota = \eta \star \beta) \end{array} \right\}, \\
\mathrm{Mono}(X,G,\neg\alpha,\neg\beta) &= \left\{ \iota : X \rightarrowtail G \;\middle|\; \begin{array}{l} (\nexists \zeta : X \rightarrowtail A.\, \iota = \zeta \star \alpha) \\ \wedge\, (\nexists \eta : X \rightarrowtail B.\, \iota = \eta \star \beta) \end{array} \right\}.
\end{aligned}$$

Consider a rewriting step $G \Rightarrow_{\rho,\mathfrak{M}} H$ defined by the DPO diagram illustrated below:



The sets of monomorphisms $\mathrm{Mono}(X,G)$ and $\mathrm{Mono}(X,H)$ can be decomposed into the following disjoint unions:

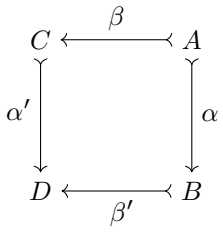
$$\begin{aligned}
\mathrm{Mono}(X,G) &= \mathrm{Mono}(X,G,l') \uplus \mathrm{Mono}(X,G,\neg l',m) \uplus \mathrm{Mono}(X,G,\neg l',\neg m), \\
\mathrm{Mono}(X,H) &= \mathrm{Mono}(X,H,r') \uplus \mathrm{Mono}(X,H,\neg r',m') \uplus \mathrm{Mono}(X,H,\neg r',\neg m').
\end{aligned}$$

Thus, the following equalities hold:

$$\begin{aligned}
|\mathrm{Mono}(X,G)| &= |\mathrm{Mono}(X,G,l')| + |\mathrm{Mono}(X,G,\neg l',m)| \\
&\quad + |\mathrm{Mono}(X,G,\neg l',\neg m)|, \\
|\mathrm{Mono}(X,H)| &= |\mathrm{Mono}(X,H,r')| + |\mathrm{Mono}(X,H,\neg r',m')| \\
&\quad + |\mathrm{Mono}(X,H,\neg r',\neg m')|.
\end{aligned}$$

The following lemma provides a way to rewrite the terms $|\mathrm{Mono}(X,G)|$ and $|\mathrm{Mono}(X,H)|$.

Lemma 4.13. *Let X be a ruler-graph. For a pushout square:*



the following equalities hold:

$$|\mathrm{Mono}(X,B)| = |\mathrm{Mono}(X,D,\beta')|,$$
$$|\mathrm{Mono}(X,C,\neg\beta)| = |\mathrm{Mono}(X,D,\neg\beta',\alpha')|.$$

Proof. See § 4.10. □

By Lemma 4.13, the following equalities hold:

$$|\mathrm{Mono}(X,G)| = |\mathrm{Mono}(X,C)| + |\mathrm{Mono}(X,L,\neg l)| + |\mathrm{Mono}(X,G,\neg l',\neg m)|,$$
$$|\mathrm{Mono}(X,H)| = |\mathrm{Mono}(X,C)| + |\mathrm{Mono}(X,R,\neg r)| + |\mathrm{Mono}(X,H,\neg r',\neg m')|.$$

Therefore, we obtain:

$$|\mathrm{Mono}(X,G)| - |\mathrm{Mono}(X,H)| = |\mathrm{Mono}(X,L,\neg l)| - |\mathrm{Mono}(X,R,\neg r)| + |\mathrm{Mono}(X,G,\neg l',\neg m)| - |\mathrm{Mono}(X,H,\neg r',\neg m')|$$

The following lemma provides a way to rewrite the term $|\mathrm{Mono}(X,L,\neg l)| - |\mathrm{Mono}(X,R,\neg r)|$.

Lemma 4.14. *Let X be a graph. Let $L \overset{l}{\hookleftarrow} K \overset{r}{\rightarrowtail} R$ be an injective DPO graph rewriting rule. We have*

$$|\mathrm{Mono}(X,L,\neg l)| - |\mathrm{Mono}(X,R,\neg r)| = |\mathrm{Mono}(X,L)| - |\mathrm{Mono}(X,R)|$$

Proof. See § 4.10. □

By Lemma 4.14, the following equalities hold:

$$|\mathrm{Mono}(X,G)| - |\mathrm{Mono}(X,H)| = |\mathrm{Mono}(X,L)| - |\mathrm{Mono}(X,R)| + |\mathrm{Mono}(X,G,\neg l',\neg m)| - |\mathrm{Mono}(X,H,\neg r',\neg m')|$$

Consequently, the weight difference satisfies:

$$
\begin{aligned}
w_{s_{\mathbb{X}}}(G) - w_{s_{\mathbb{X}}}(H) &\stackrel{\text{def}}{=} \sum_{X \in \mathbb{X}} s_{\mathbb{X}}(X) * m_X(G) - \sum_{X \in \mathbb{X}} s_{\mathbb{X}}(X) * m_X(H) \\
&\stackrel{\text{def}}{=} \sum_{X \in \mathbb{X}} s_{\mathbb{X}}(X) * |\mathrm{Mono}(X, G)| - \sum_{X \in \mathbb{X}} s_{\mathbb{X}}(X) * |\mathrm{Mono}(X, H)| \\
&= \sum_{X \in \mathbb{X}} s_{\mathbb{X}}(X) * (|\mathrm{Mono}(X, G)| - |\mathrm{Mono}(X, H)|) \\
&= \sum_{X \in \mathbb{X}} s_{\mathbb{X}}(X) * (|\mathrm{Mono}(X, L)| - |\mathrm{Mono}(X, R)|) + \\
&\quad \sum_{X \in \mathbb{X}} s_{\mathbb{X}}(X) * (|\mathrm{Mono}(X, G, \neg l', \neg m)| - |\mathrm{Mono}(X, H, \neg r', \neg m')|) \,.
\end{aligned}
$$

Since $\sum_{X \in \mathbb{X}} s(X) * (|\mathrm{Mono}(X, L)| - |\mathrm{Mono}(X, R)|)$ can be precisely computed, the key challenge is to establish a lower bound for

$$\sum_{X \in \mathbb{X}} s_{\mathbb{X}}(X) * (|\mathrm{Mono}(X, G, \neg l', \neg m)| - |\mathrm{Mono}(X, H, \neg r', \neg m')|) \,.$$

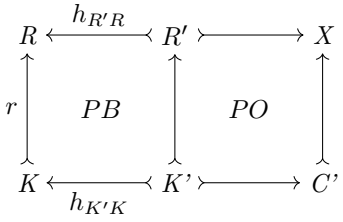
In the next section, we introduce the concept of non-increasing rules, which helps to address this challenge.

4.4 Non-increasing rules

We define the set $D(R, X)$ of all subgraphs of R which, when glued along some common interface elements with a subgraph C' of a context graph C, can form an implicit X-occurrence.

Definition 4.15. *Let X be a graph and $L \overset{l}{\hookleftarrow} K \overset{r}{\rightarrowtail} R$ an injective DPO rewriting rule. The* ***set $D(R, X)$ of distinguished subgraphs of the right-hand side graph*** *consists of all subgraphs $R' \subseteq R$ satisfying the following conditions:*

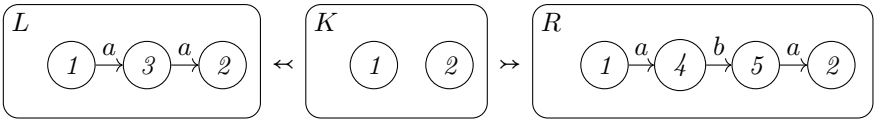
1. *there are graphs K', C' and morphisms such that the following diagram where $h_{K'K}$ and $h_{R'R}$ are inclusion functions can be constructed:*

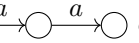
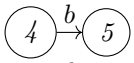


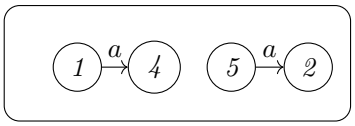
2. *R' is not a subgraph of $r(K)$,*
3. *R' is not isomorphic to X,*

The first and second conditions ensure that such X-occurrences are included neither in C nor in R, thereby guaranteeing that they are implicit. The third condition ensures that any $R' \in D(R, X)$ can be glued along some common interface elements with a subgraph C' of a context graph C to form an X-occurrence.

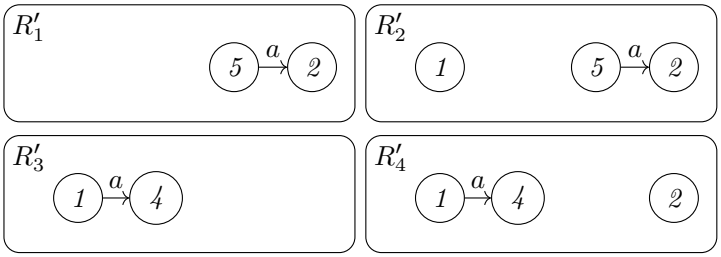
Example 4.16. *Consider the rewriting rule [19, Example 6]:*



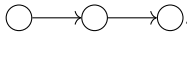
Let X be  *and $R' \in D(R, X)$. Firstly, R' is neither the empty graph nor isomorphic to X, by Conditions 2 and 3 of Definition 4.15. Secondly, R' must contain node 4 or node 5 to avoid being a subgraph of $r(K)$ (Condition 2). Thirdly, if R' contains nodes 4 or 5, they must be connected to node 1 or node 2 respectively, otherwise R' would be disconnected and it would be impossible to construct the diagram in the first condition of Definition 4.15. Fourthly, R' must not contain*  *by Condition 1. Finally, the following graph can not be R' because it violates Condition 1.*

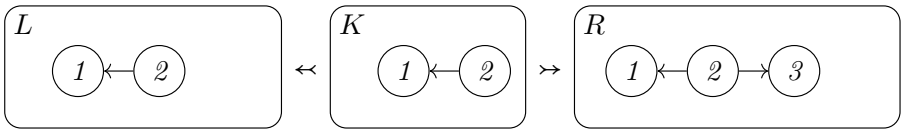


Thus, $D(R, X)$ consists of the four graphs R'_1, R'_2, R'_3 and R'_4:



For each of them, the construction of the pullback square in Condition 1 is demonstrated in Example 4.19 and the construction of the pushout square required by Condition 1 is straightforward.

Example 4.17. *Let X be the graph*  *. Consider the following rule:*



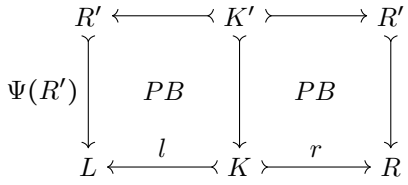
Let $R' \in D(R, X)$. R' must contain node 3, otherwise it would be a subgraph of $r(K)$, violating Condition 2 of Definition 4.15. Node 3 must be connected to an interface node, otherwise Condition 1 of Definition 4.15 would be violated. Therefore, R' must include  *. The graph R is not in $D(R, X)$ because it violates Condition 1. Thus, the set $D(R, X)$ consists of the following graphs:*

- *R'_1:* *,*
- *R'_2:* *.*

We introduce the concept of non-increasing occurrences. Intuitively, if a rule ρ is X-non-increasing, then for any rewriting step using ρ, there is an injective mapping from the set of X-occurrences implicitly created by the step to the set of X-occurrences implicitly destroyed by the step. This intuition is made precise in Lemma 4.24.

Definition 4.18 (X-non-increasing rule)**.** *Let $\rho = (L \overset{l}{\leftarrowtail} K \overset{r}{\rightarrowtail} R)$ be a rule and X a graph. Let Ψ be a function associating $R' \in D(R, X)$ to a homomorphism in $\mathrm{Mono}(R', L)$. Rule ρ is said to be X-**non-increasing** under Ψ if the following four conditions hold:*

1. *For all $R' \in D(R, X)$, we can construct the following diagram where all morphisms other than $\Psi(R')$, l and r are inclusion functions:*



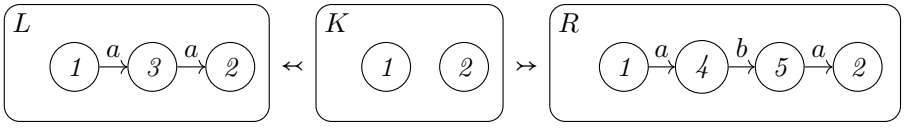
2. *For all $R' \in D(R, X)$, for all nodes and edges x in R', if $x \notin \mathrm{Im}(r)$ then $\Psi(R')(x) \notin \mathrm{Im}(l)$,*
3. *For all $R', R'' \in D(R, X)$, for all edges x in R' and y in R'', if $x \neq y$, then $\Psi(R')(x) \neq \Psi(R'')(y)$,*
4. *If X has isolated nodes, then for all $R', R'' \in D(R, X)$ and nodes x and y in R' and R'' respectively, if $x \neq y$, then $\Psi(R')(x) \neq \Psi(R'')(y)$.*

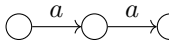
*When Ψ is clear from context or irrelevant to the discussion, we may simply say that ρ is X-**non-increasing**.*

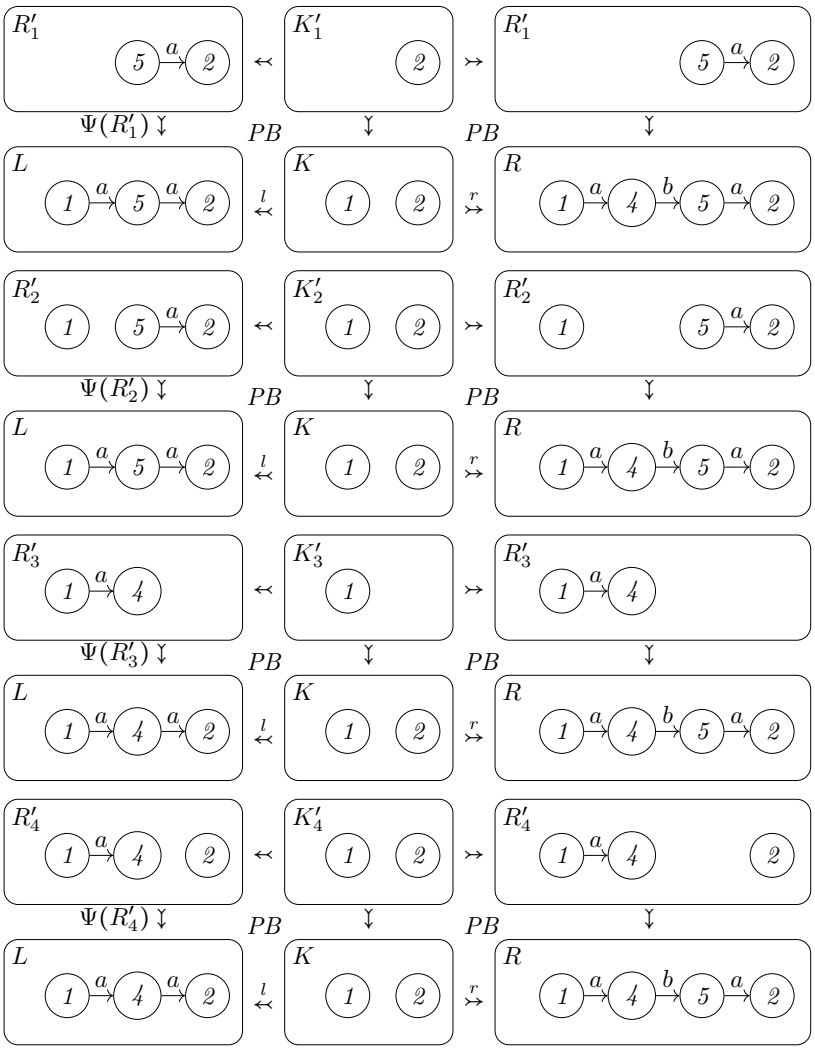
The first condition ensures that whenever $R' \in D(R, X)$ forms an X-occurrence in the result graph H, this occurrence has a corresponding X-occurrence relative to the morphism $h_{R'L}$ (as defined in Definition 4.7) in the host graph G. The second condition guarantees that any X-occurrence implicitly created by the rewriting step has its corresponding X-occurrence not included in the context (Consequently, its corresponding X-occurrence is implicitly destroyed by the rewriting step). Example 4.20 and Example 4.21 show the necessity of this condition. The third and fourth conditions ensure that distinct X-occurrences created implicitly have distinct corresponding X-occurrences. Example 4.22 and Example 4.23 show the necessity of these conditions.

The following example shows an X-non-increasing rule.

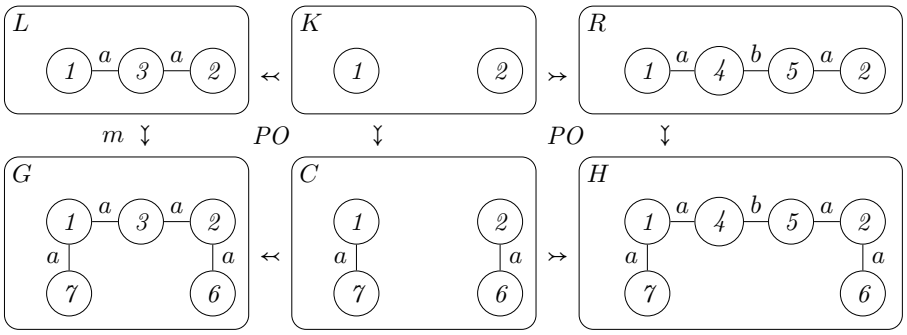
Example 4.19. *Consider the rule illustrated below and previously presented in Example 4.16.*



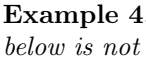
Let X be the graph  . *As explained in Example 4.16, the set $D(R, X)$ consists of exactly four graphs: R'_1, R'_2, R'_3 and R'_4. The rule is X-non-increasing under the function Ψ that maps each R'_i to the inclusion morphism from R'_i to L. specifically, it satisfies Condition 1 of Definition 4.18 because the following diagrams can be constructed, and other conditions in Definition 4.18 are straightforward to verify.*

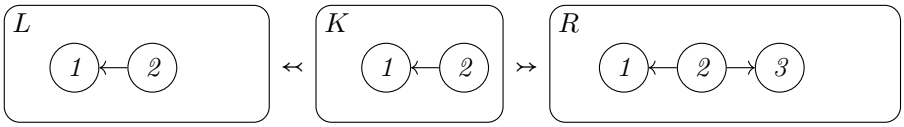


To illustrate the X-non-increasing property captured by Definition 4.18, consider the rewriting step defined by the DPO diagram below.

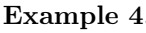


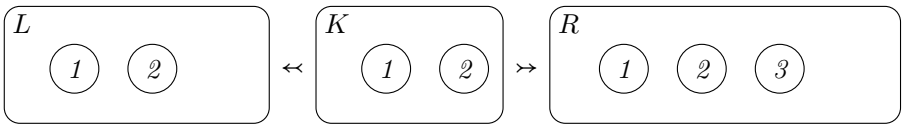
Whenever R'_i (for $1 \leq i \leq 4$) forms an X-occurrence x in H that is included neither in R nor in C with a subgraph $C' \subseteq C \cap x \subseteq C$, the graph $\Psi(\mathrm{R'}) \subseteq L$ forms a X-occurrence in G with C' that is included neither in L nor in C. Furthermore, distinct X-occurrences in H that are included neither in R nor in C have distinct corresponding X-occurrences in G that are included neither in L nor in C.

Example 4.20. *Let X be the graph*  *. The rewriting rule shown below is not necessarily X-non-increasing.*

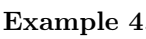


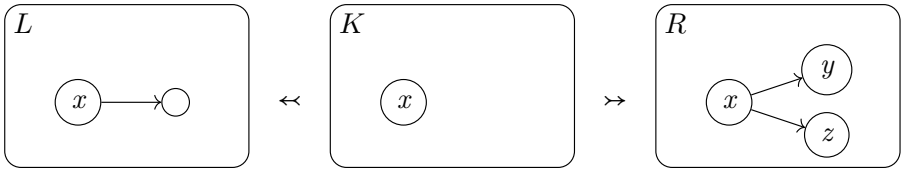
As shown in Example 4.17, $D(R, X)$ contains the graph R':  *. The unique monomorphism $h_{R'L} : R' \rightarrowtail L$ fails the second condition of Definition 4.18. For any rewriting step using this rule, the corresponding X-occurrence relative to $h_{R'L}$ of any implicitly created X-occurrence is included in the context.*

Example 4.21. *Let X be the graph*  *. The rewriting rule shown below is not necessarily X-non-increasing.*



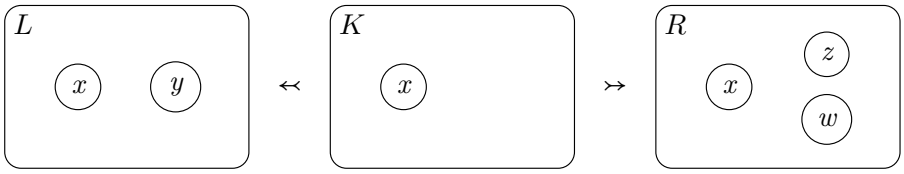
The set $D(R, X)$ contains R':  *. Despite the existence of a unique monomorphism $h_{R'L} : R' \rightarrowtail L$ preserving interface elements, this rule fails the second condition of Definition 4.18. For any rewriting step using this rule, the corresponding X-occurrence relative to $h_{R'L}$ of any implicitly created X-occurrence is included in the context.*

Example 4.22. *Let X be the graph*  *. The rewriting rule shown below is not necessarily X-non-increasing.*



The set $D(R, X)$ contains exactly two elements R': $x \to y$ and R'': $x \to z$. Despite unique monomorphisms $h_{R'L} : R' \rightarrowtail L$ and $h_{R''L} : R'' \rightarrowtail L$ preserving interface elements, they fail the third condition of Definition 4.18. For any rewriting step using this rule, implicitly created X-occurrences with the same subgraph of the context have the same corresponding X-occurrence.

Example 4.23. *Let X be the graph $\bigcirc \longrightarrow \bigcirc$ $\bigcirc$, which has an isolated node. The rewriting rule shown below is not necessarily X-non-increasing.*



The set $D(R, X)$ contains exactly two elements R': x z and R'': x w. There are unique monomorphisms $h_{R'L} : R' \rightarrowtail L$ and $h_{R''L} : R'' \rightarrowtail L$ preserving interface elements, but they fail the fourth condition of Definition 4.18. For any rewriting step using this rule, implicitly created X-occurrences with the same subgraph of the context have the same corresponding X-occurrence.

4.5 Solution to the key challenge

Using the notion of X-non-increasing, we state the following lemma. It formalizes the intuition that if ρ is X-non-increasing, then, for any rewriting step using ρ, more X-occurrences are implicitly destroyed than implicitly created by the rewriting step.

Lemma 4.24. *Let X be a ruler-graph and $\rho = (L \overset{l}{\hookleftarrow} K \overset{r}{\rightarrowtail} R)$ an injective DPO rewriting rule. Suppose that ρ is X-non-increasing. For every rewriting step induced by the following DPO diagram:*

$$\begin{array}{ccccc} L & \overset{l}{\hookleftarrow} & K & \overset{r}{\hookrightarrow} & R \\ \downarrow m & PO & \downarrow & PO & \downarrow m' \\ G & \underset{l'}{\hookleftarrow} & C & \underset{r'}{\hookrightarrow} & H \end{array}$$

The following inequality holds:

$$|\,\mathrm{Mono}(X, G, \neg m, \neg l')| \geq |\,\mathrm{Mono}(X, H, \neg m', \neg r')|$$

Proof. See § 4.10. □

4.6 Termination criterion

Finally, we present a lemma that allows us to overapproximate the change in weight of a graph when a rule is applied, followed by our termination criterion.

Lemma 4.25 (Decreasing step)**.** *Let $\rho = (L \overset{l}{\leftarrowtail} K \overset{r}{\rightarrowtail} R)$ be an injective DPO rewriting rule, $\mathbb{X}$ a set of ruler-graphs, $s_{\mathbb{X}} : \mathbb{X} \to \mathbb{N}$ a weight function, and $G \Rightarrow_{\rho,\mathfrak{M}} H$ a rewriting step. If ρ is X-non-increasing for every ruler-graph $X \in \mathbb{X}$, then*

$$w_{s_{\mathbb{X}}}(G) - w_{s_{\mathbb{X}}}(H) \geq w_{s_{\mathbb{X}}}(L) - w_{s_{\mathbb{X}}}(R).$$

Proof. See § 4.10. □

Theorem 4.26 (Termination)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be sets of injective DPO rewriting rules, $\mathbb{X}$ a set of ruler-graphs, and $s_{\mathbb{X}}$ a weight function. If the following conditions hold:*

1. *for every $\rho \in \mathcal{A} \cup \mathcal{B}$ and for every $X \in \mathbb{X}$, the rule ρ is X-non-increasing,*
2. *for every $\rho \in \mathcal{A}$, we have $w_{s_{\mathbb{X}}}(lhs(\rho)) > w_{s_{\mathbb{X}}}(rhs(\rho))$,*
3. *for every $\rho \in \mathcal{B}$, we have $w_{s_{\mathbb{X}}}(lhs(\rho)) \geq w_{s_{\mathbb{X}}}(rhs(\rho))$.*

Then $\Rightarrow_{\mathcal{A},\mathcal{M}}$ terminates relative to $\Rightarrow_{\mathcal{B},\mathcal{M}}$.

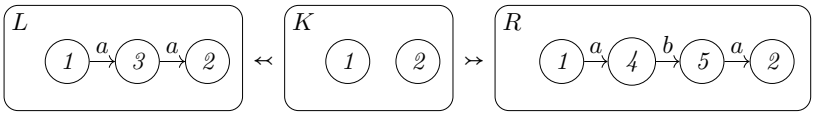
Proof. See § 4.10. □

Our technique is described for DPO rewriting with injective matches due to their expressiveness, but it also extends to non-injective matches [43].

4.7 Examples

While the type graph method [19, 17, 38] can prove termination for this example, it requires constructing an appropriate weighted type graph, a task that is difficult in general [17, §6]. In contrast, our criterion requires checking simpler conditions.

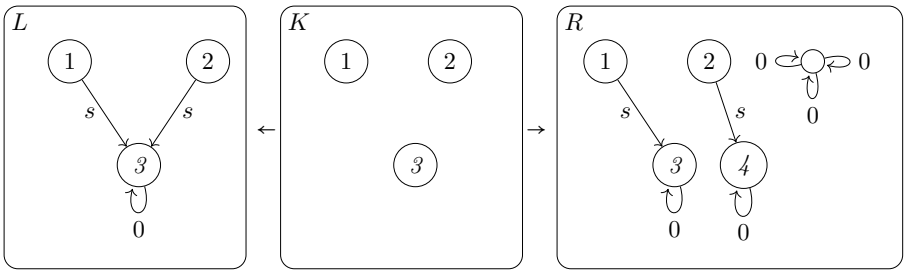
Example 4.27. *Consider the following rewriting rule ρ ([19, Examples 4 and 6] and [17, Example 2]):*



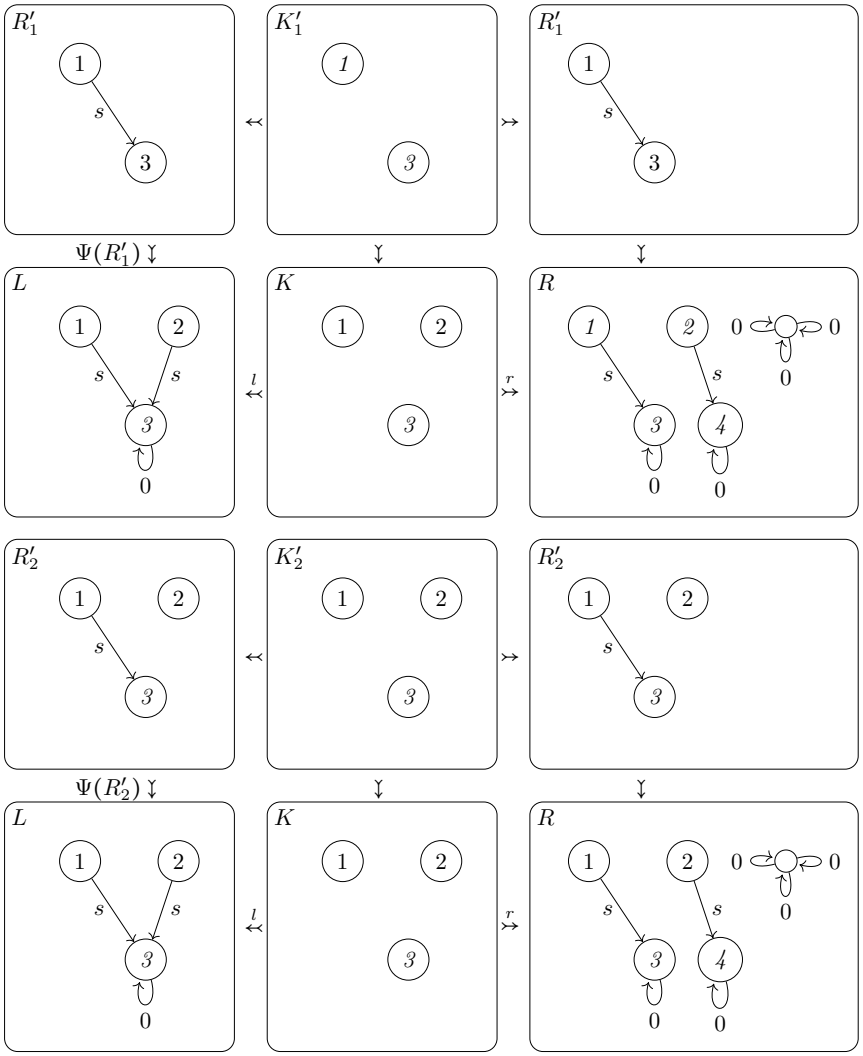
Let X be the graph $\bigcirc \xrightarrow{a} \bigcirc \xrightarrow{a} \bigcirc$, $\mathbb{X} = \{X\}$ and $s_{\mathbb{X}}(X) = 1$. The rule is X-non-increasing as shown in Example 4.19. Since $w_{s_{\mathbb{X}}}(\mathrm{lhs}(\rho)) = 1 > 0 = w_{s_{\mathbb{X}}}(\mathrm{rhs}(\rho))$, it terminates by Theorem 4.26.

The techniques from some previous works [19, 17, 38, 71, 66] fail to prove termination of the following examples, while our method can be applied to prove their termination.

Example 4.28. *The rewriting rule shown below is obtained by removing all edges in the interface of a rule presented by Plump [71, Example 6]:*

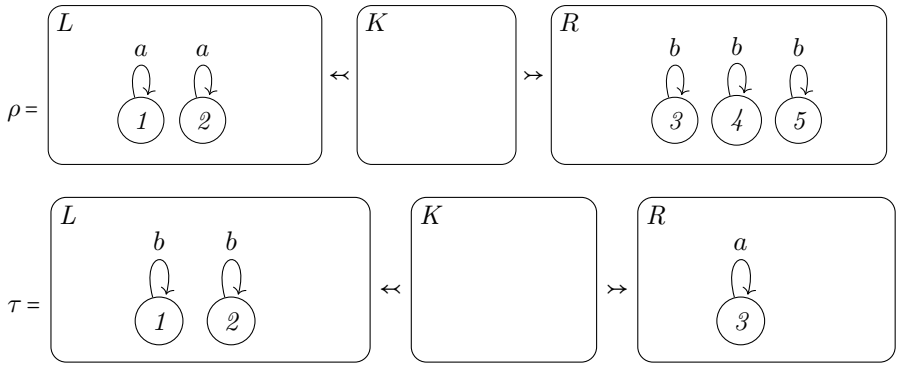


Let X be the ruler-graph $\bigcirc \xrightarrow{s} \bigcirc \xleftarrow{s} \bigcirc$, $\mathbb{X}=\{X\}$ *and* $s_{\mathbb{X}}(X)=1$. *The set* $D(R,X)$ *consists of two elements* R'_1: $1 \xrightarrow{s} 3$ *and* R'_2: $2 \quad 1 \xrightarrow{s} 3$. *Each* R'_i *admits a unique monomorphism* $h_{R'_iL}: R'_i \rightarrowtail L$ *preserving interface elements. The rule is* X*-non-increasing under the function* Ψ *that maps each element* $R'_i \in D(R,X)$ *to* $h_{R'_iL}$, *because the following diagrams can be constructed, and other conditions in Definition 4.18 are straightforward to verify. Since* $w_{s_{\mathbb{X}}}(L)=1>0=w_{s_{\mathbb{X}}}(R)$, *the system terminates by Theorem 4.26.*



The following example demonstrates the usefulness of a weight function that assigns distinct weight to measurements from different ruler-graphs.

Example 4.29. *Consider the rewriting rules [66, Example 5.6]:*



Let X be the ruler-graph *a , Y the ruler-graph* *b and $\mathbb{X}=\{X,Y\}$. The sets $D(R_\rho,X)$, $D(R_\rho,Y)$, $D(R_\tau,X)$ and $D(R_\tau,Y)$ are all empty.*

Let Ψ be the empty function (i.e., a function with empty domain). Both rules ρ and τ are X- and Y-non-increasing under Ψ, since all conditions in Definition 4.18 are trivially met: every condition quantifies over elements in the empty set.

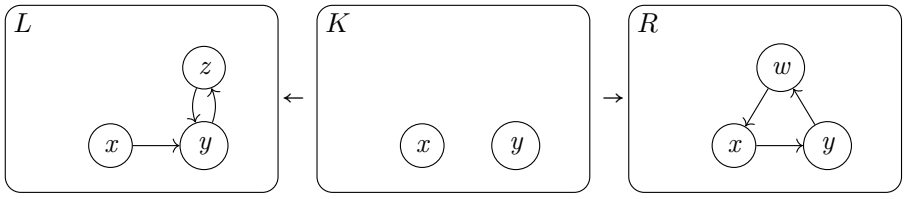
For $s_{\mathbb{X}}(X)=5$ and $s_{\mathbb{X}}(Y)=3$, we have:

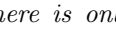
- $w_{s_{\mathbb{X}}}(L_\rho)=10>9=w_{s_{\mathbb{X}}}(R_\rho)$, *and*
- $w_{s_{\mathbb{X}}}(L_\tau)=6>5=w_{s_{\mathbb{X}}}(R_\tau)$.

Therefore, the system terminates by Theorem 4.26.

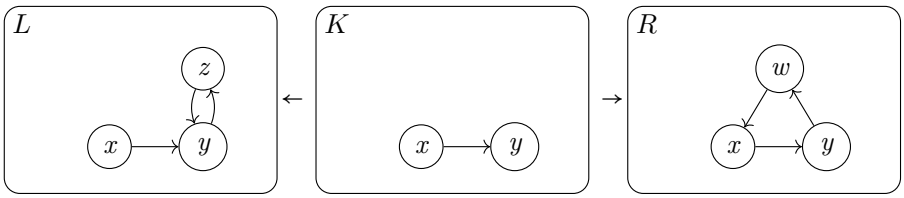
However, for $s'_{\mathbb{X}}(X)=1$ and $s'_{\mathbb{X}}(Y)=1$, we have $w_{s'_{\mathbb{X}}}(L_\rho)=2 \not> 3=w_{s'_{\mathbb{X}}}(R_\rho)$ which prevents us from applying Theorem 4.26 to prove termination of the rewriting system.

Example 4.30. *Consider the following rule [38, Example 6.3]:*



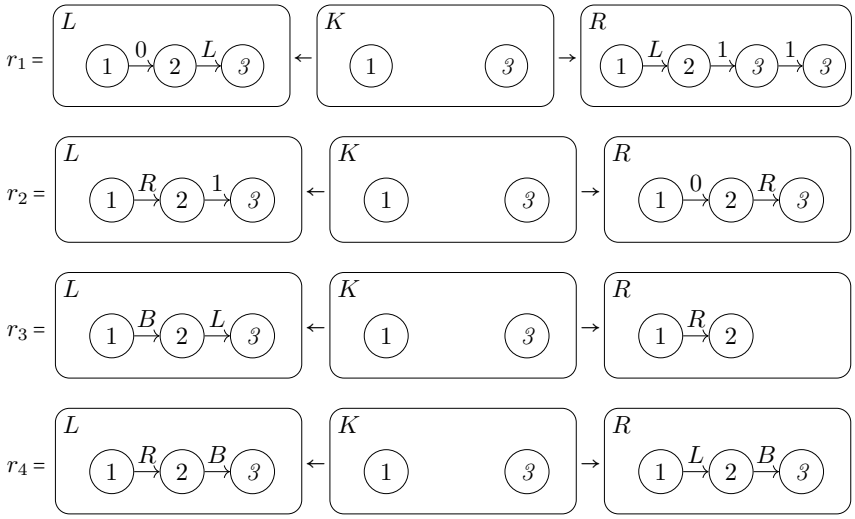
Let X be the graph  *and $\mathbb{X}=\{X\}$. Let $s_{\mathbb{X}}$ be the weight function with $s_{\mathbb{X}}(X)=1$. There is only one element*  *in $D(X,R)$. Since $w_{s_{\mathbb{X}}}(L)=1>0=w_{s_{\mathbb{X}}}(R)$, the rule terminates by Theorem 4.26.*

Example 4.31. *Consider the following rewriting rule [66, Example 5.5]:*



The termination of the system in the DPO rewriting framework with monic matches can be established using our approach; the argument is identical to that of Example 4.30.

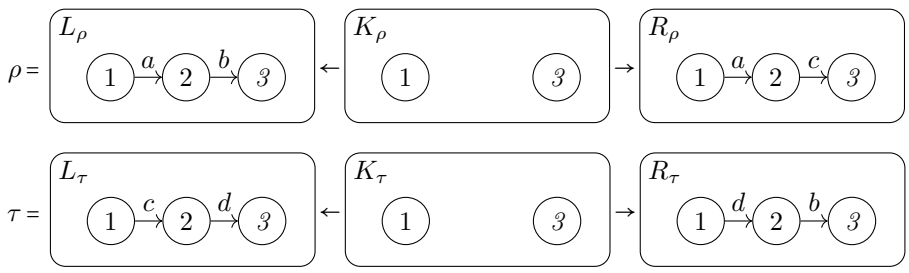
Example 4.32. *Our method cannot prove the termination of the DPO rewriting system with monic matches [71, Example 4]:*

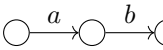


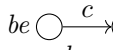
The rule r_3 can be eliminated by considering occurrences of $\bigcirc \xrightarrow{B} \bigcirc$*, and the rule r_4 can be eliminated by considering occurrences of* $\bigcirc \xrightarrow{R} \bigcirc$*.*

However, our method cannot eliminate r_1 and r_2. Yet, combined with the technique proposed by Plump [71], we can prove the termination of this rewriting system: $\{r_1\}$ and $\{r_2\}$ are both terminating, and their union is terminating [71].

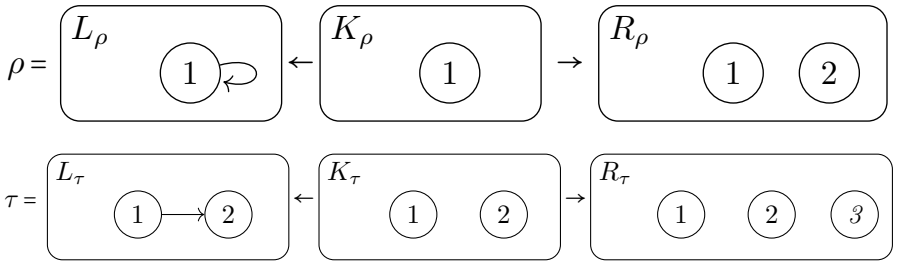
Example 4.33. *Consider the rewriting rules [72, Example 3.8]:*

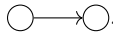


Let X be  . *We have $D(R_\rho, X) = \varnothing$ and $D(R_\tau, X) = \varnothing$. Therefore, both rules are X-non-increasing. We have $|\mathrm{Mono}(X, L_\rho)| = 1 > 0 = |\mathrm{Mono}(X, R_\rho)|$ for ρ, and $|\mathrm{Mono}(X, L_\tau)| = 0 = |\mathrm{Mono}(X, R_\tau)|$ for τ, therefore ρ can be eliminated.*

Let Y be  . *We have $D(R_\tau, Y) = \varnothing$. Therefore, τ is Y-non-increasing. Moreover, we have $|\mathrm{Mono}(Y, L_\tau)| = 1 > 0 = |\mathrm{Mono}(Y, R_\tau)|$. Thus, this rewriting system terminates by Theorem 4.26.*

Example 4.34. *Consider the following rewriting rules [66, Example 5.3]:*



Let X be  . *Both ρ and τ are X-non-increasing because*

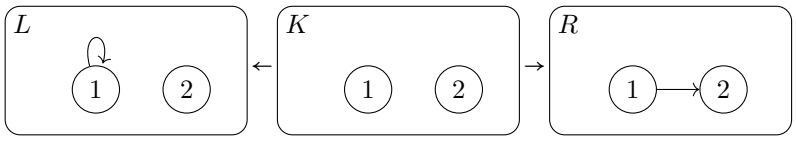
- *$D(R_\tau, X) = \varnothing$, and*
- *$D(R_\rho, X) = \varnothing$, and*
- *all conditions of Definition 4.18 are trivially satisfied.*

Since the following inequality holds:

- *$|\mathrm{Mono}(X, L_\rho)| = 1 > 0 = |\mathrm{Mono}(X, R_\rho)|$, and*
- *$|\mathrm{Mono}(X, L_\tau)| = 1 > 0 = |\mathrm{Mono}(X, R_\tau)|$,*

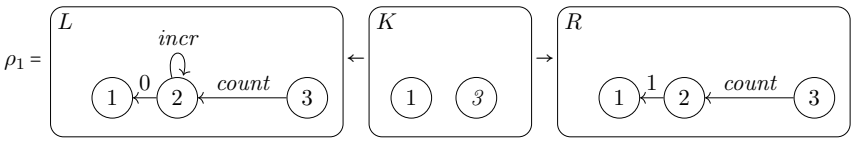
the rewriting system terminates by Theorem 4.26.

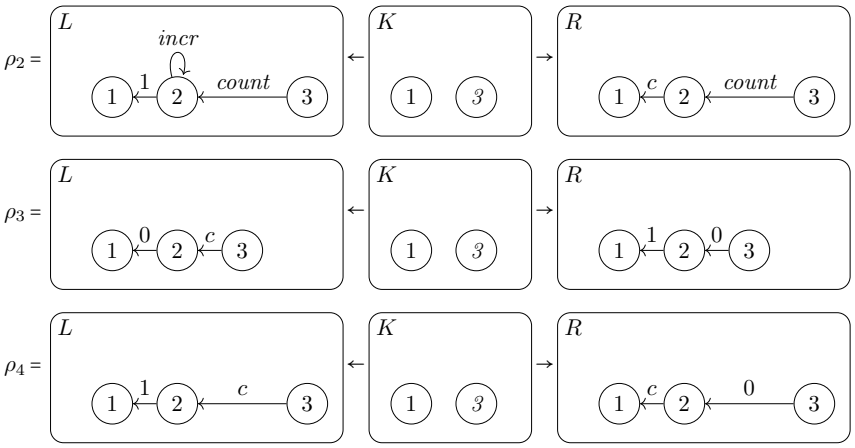
Example 4.35. *Consider the rule [38, Example 6.2]:*

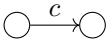
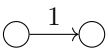


Our method proves termination for this rule using the ruler-graph  *with weight* 1.

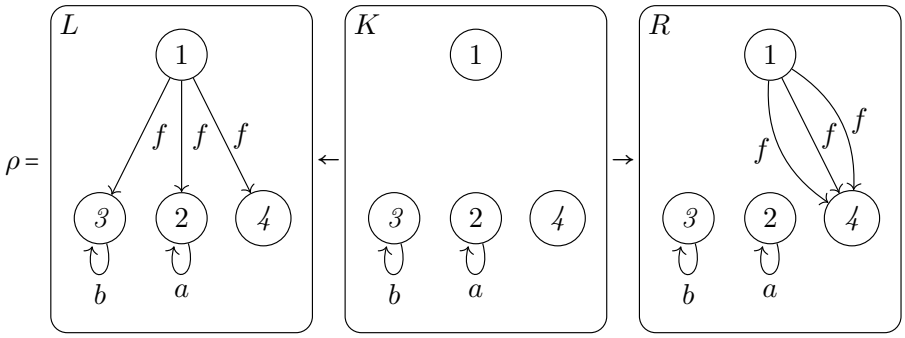
Example 4.36. *Our method can prove the termination of the following DPO rewriting system ([17, Example 4]):*



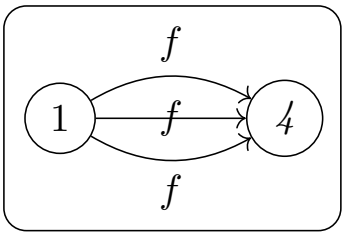


The rules ρ_1 and ρ_2 can be eliminated by considering the ruler graph  *with weight* 1*; the rule ρ_3 can be eliminated by considering the ruler-graph*  *with weight* 1*; and, finally, ρ_4 can be eliminated by considering occurrences of*  *with weight* 1*.*

Example 4.37 (Limitation)**.** *Consider the rule [72, Example 4.1]:*

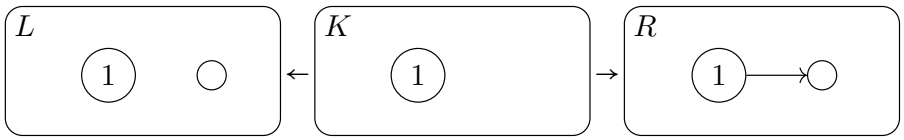


To apply our method, the ruler-graph X must be a subgraph of L that contains at least one edge labeled by f. Indeed, the number of occurrences of any graph made up solely of edges labeled a and b is invariant under rewriting steps using ρ. Consequently, $D(R, X)$ contains a graph R' which includes the following subgraph:



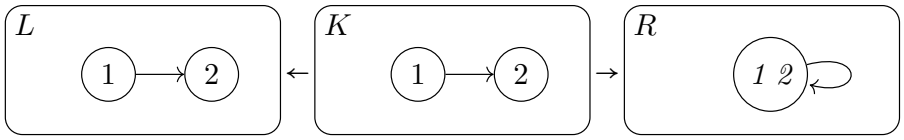
However, for graph R' the diagram required by the condition 1 of Definition 4.18 cannot be constructed. Therefore, our method cannot be applied.

Example 4.38 (Limitation)**.** *Our method fails to prove termination of the rewriting rule [38, Example D.3]:*



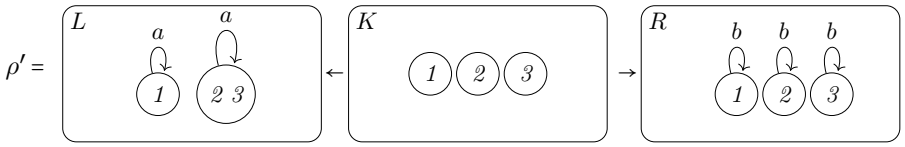
This happens because, for any system $\mathcal{R}$, any rewriting step $G \Rightarrow_{\mathcal{R},\mathfrak{M}} H$, any set of ruler-graphs $\mathbb{X}$, and any weight function $s_{\mathbb{X}}$, we have $w_{s_{\mathbb{X}}}(G) \geq w_{s_{\mathbb{X}}}(H)$ as G is a subgraph of H.

Example 4.39 (Limitation)**.** *Consider the rewriting rule [66, Example 5.2]:*



The termination of this rule is not in the scope of our method due to its non-node-injective right-hand side morphism. It can be proved using the method proposed by Overbeek et al. [66].

Example 4.40 (Limitation)**.** *Consider the rewriting rule with monic matches [66, Example 5.6]:*



Its termination is not in the scope of our method due to its non-node-injective left-hand side morphism.

4.8 Comparison with previous approaches

A comparative analysis of termination techniques for DPO graph rewriting systems, drawn from prior work [72, 71, 19, 17, 38, 66], is summarized in Table 4.1. Our approach successfully proves termination for 14 of these systems. For ease of reading, many cells in Table 4.1 are marked with "−", indicating that the corresponding technique is out of scope for the given example or, more often, that it is not relevant to our discussion, since other entries already show that the technique and ours are not directly comparable.

In the remainder of this section, we compare our method with some existing methods in more detail.

Table 4.1: Applicability of termination techniques to DPO rewriting examples. The symbol ✓ indicates termination can be proved by the technique, ✗ indicates it cannot be proved, and – denotes irrelevance or out-of-scope cases.

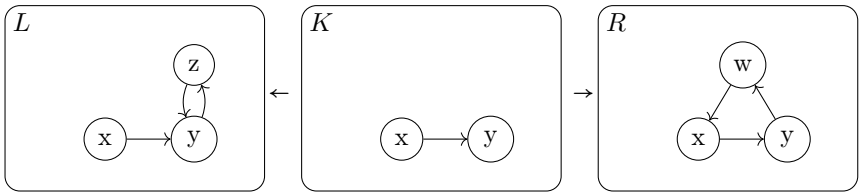
Examples \ **Techniques**		Forward closure [72]	Modular criterion [71]	Type graph [19]	Type graph [17]	Type graph [38]	Subgraph counting [66]	Morphism Counting (Chapter 4)
Chapter 4	Example 4.28	✓	✗	✗	✗	✗	✗	✓
[72]	Example 3.8	✓	–	–	–	–	–	✓
[71]	Example 3	–	✓	–	–	–	–	✓
	Example 5	–	✓	–	–	–	–	✓
[19]	Examples 4 and 6	–	–	✓	–	–	–	✓
[17]	Example 2	–	–	–	✓	–	–	✓
	Example 4	–	–	–	✓	–	–	✓
[38]	Example 6.2	–	–	–	–	✓	–	✓
	Example 6.3	–	–	–	–	✓	✗	✓
	Example D.1	–	–	–	–	✓	–	✓
[66]	Example 5.3	–	–	–	–	–	✓	✓
	Example 5.5	–	–	–	–	–	✓	✓
	Example 5.6	–	–	–	–	–	✓	✓
	Example 5.8	–	–	–	–	–	✓	✓
[71]	Example 6	–	✓	–	–	–	–	–
[38]	Example 6.4	–	–	–	–	✓	–	–
	Example 6.5	–	–	–	–	✓	–	–
	Example D.4	–	–	–	–	✓	–	–
[66]	Example 5.2	–	–	–	–	–	✓	–
	Example 5.7	–	–	–	–	–	✓	–
	Example 5.9	–	–	–	–	–	✓	–
[72]	Example 4.1	✓	–	–	–	–	✓	✗
[71]	Example 4	–	✓	–	–	–	–	✗
[19]	Example 1	–	–	✓	–	–	–	✗
	Routing Protocol	–	–	✓	–	–	–	✗
	Example 5	–	–	✓	–	–	–	✗
[17]	Example 5	–	–	–	✓	–	–	✗
	Example 6	–	–	–	✓	–	–	✗
[38]	Example D.2	–	–	–	–	✓	–	✗
	Example D.3	–	–	–	–	✓	✗	✗

The subgraph-counting method by Overbeek and Endrullis (2024) [66] is very closely related to our work in the setting where they are both defined. Both methods weigh objects by summing weighted morphisms targeting them, and, for both methods, the key challenge lies in estimating weights for morphisms whose images partially overlap with the rewriting context, because morphisms fully embedded in the context are shared between host and resulting graphs, while those entirely within left- or right-hand-side graphs are trivial to quantify.

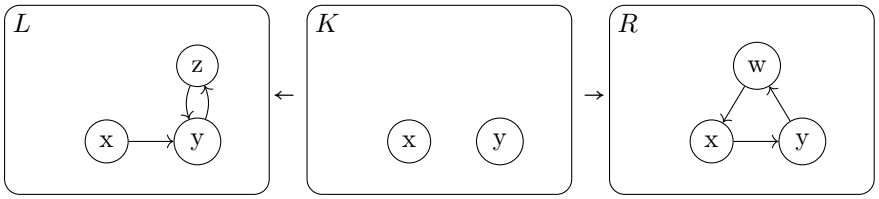
The two methods employ distinct strategies to overcome this challenge. The

subgraph counting method relies on type-morphisms (see [66, page 9, remark 4.11, Lemma 4.23]), whereas our approach requires injective mappings from (i) subgraph occurrences partially overlapping the rewriting context in the result graph to (ii) those in the host graph.

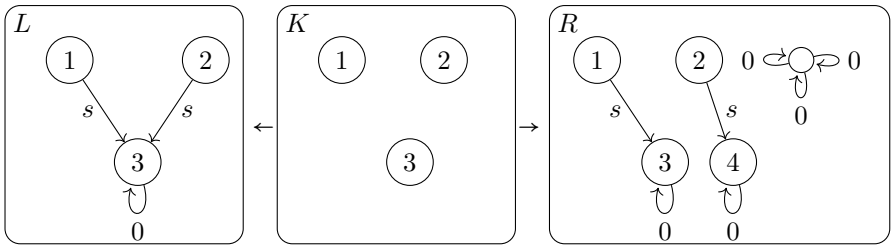
Both approaches have limitations and strengths. Endrullis and Overbeek's approach suffers from discrete interface—interface graph with no edges, as mentioned in [66, Example 5.5]. For instance, it proves termination of Example 4.31:



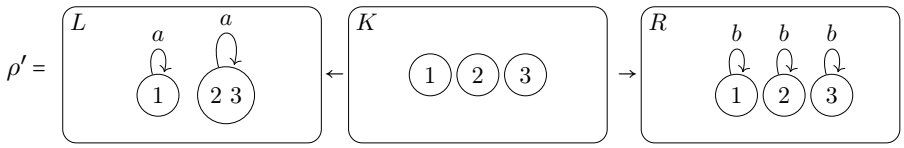
but fails for Example 4.30:



while the rules differ only by an edge in the interface graph. Similarly, it fails to prove termination for Example 4.28:



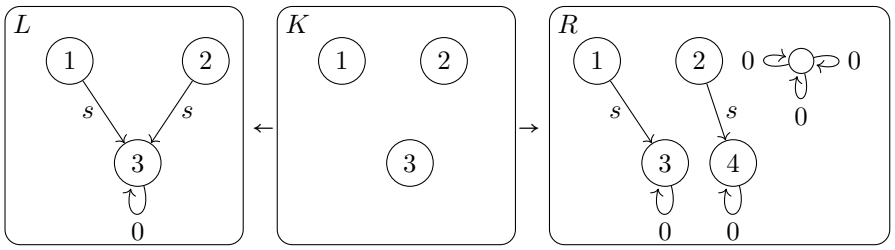
but if an edge labeled by "s" from node 1 to node 3 is added to the interface graph, then it succeeds. Our approach, however, can handle all the aforementioned cases. Nevertheless, their method can address the system in Example 4.37:



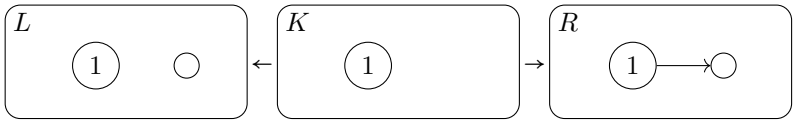
which remains beyond the scope of our technique. Finally, both approaches cannot handle Example 4.38 (see [66, Remark 6.2]).

The type graph method [85, 19, 17, 39] is also related to our approach, as both methods involve counting morphisms. However, they differ in direction: the type graph method counts morphisms from the object to a fixed type graph, while our method counts morphisms from a fixed pattern graph to the object.

Concerning their applicability, neither method strictly subsumes the other. On the one hand, the termination of Example 4.28, shown below, can be proved by our method but not by the type graph methods due to the existence of a surjection from the output graph to the input graph as explained in [38, Example D.4].

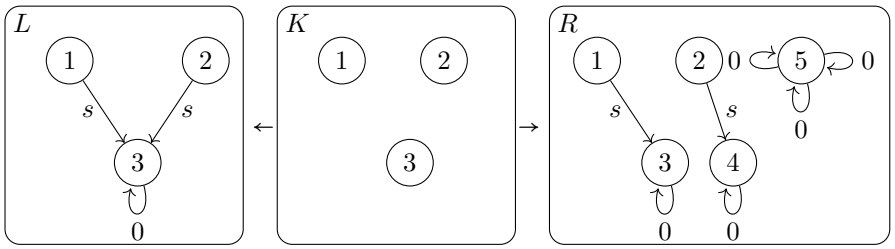


On the other hand, our method cannot prove the termination of Example 4.38, shown below, because of the injection from the left-hand side graph L to the right-hand side graph R, but the type graph methods can handle it.

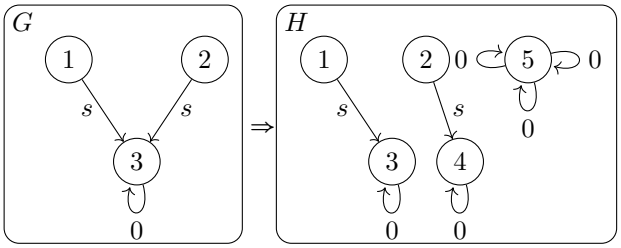


Plump (1995) [72] gives a necessary and sufficient termination criterion for left-injective DPO hypergraph rewriting in terms of forward closure. However, the verification of the criterion is undecidable in general.

Our method complements the approach proposed by Plump (2018) [71]: whereas the method decomposes a rewriting system into subsystems and proves termination of the entire system by combining termination proofs for the subsystems, the termination of each subsystem must still be established separately. For example, the measure based on the indegree proposed in [71] cannot prove the termination of Example 4.28, illustrated below, due to the loops on the node 5 in the right-hand side graph R.



Specifically, consider the following rewriting step with the rule:



For all $k \in \mathbb{N}$, the weight of the host graph G—defined as the sum of node in-degrees raised to the power of k—is $0^k + 0^k + 3^k = 3^k$, while the weight of the result graph H is $0^k + 0^k + 2^k + 2^k + 3^k = 2^k + 2^k + 3^k$. Since $3^k < 2^k + 2^k + 3^k$, the weight does not decrease. However, our method succeeds (see Example 4.28).

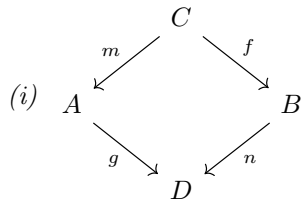
4.9 Conclusion

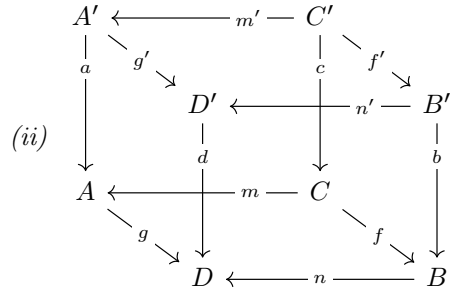
We have presented a machine-checkable sufficient condition for relative termination of DPO systems with injective rules on edge-labeled multigraphs. This method resolves cases where prior interpretation-based techniques [85, 19, 17, 38, 66] fail (e.g.,Example 4.28) and, unlike the subgraph-counting approach in [66], avoids limitations arising from interfaces with no edges. An implementation of the method is described in Chapter 6.

4.10 Appendix

Adhesive Category and VK-square

Definition 4.41 (Van Kampen Square [54, Definition 2])**.** *A van Kampen (VK) square (i) is a pushout square which satisfies the following condition: given a commutative cube (ii) of which (i) forms the bottom face and the back faces are pullbacks, the front faces are pullbacks if and only if the top face is a pushout.*





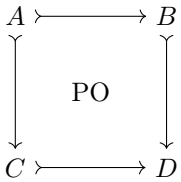
Definition 4.42 (Adhesive category[54, Definition 5])**.** *A category $\mathcal{C}$ is said to be **adhesive** if*

- *$\mathcal{C}$ has pushouts along monomorphisms, and*
- *$\mathcal{C}$ has pullbacks, and*
- *pushouts along monomorphisms are VK squares.*

Proposition 4.43 ([54, Proposition 8])**.** *The category **Graph** of directed labeled multigraphs is adhesive.*

Auxiliary lemmas

Lemma 4.44. *Consider the pushout square in* **Graph** *of form*



The following statements hold:

1. *If $b \in B$ and $c \in C$ such that $h_{BD}(b) = h_{CD}(c)$ then there is $a \in A$ such that $h_{AB}(a) = b$ and $h_{AC}(a) = c$;*

2. *For all* $b \in B \setminus h_{AB}(A)$ *and* $c \in C$*, we have* $\beta'(b) \neq \alpha'(c)$.

Proof. Let $b \in B$ and $c \in C$ such that $h_{BD}(b) = h_{CD}(c)$.

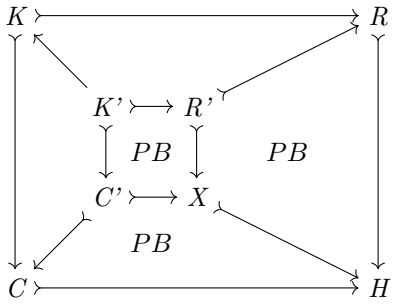
There are two cases: if b and c are two nodes then let S be the singleton graph with a unique node u, otherwise—b and c must be two edges with the same label l—let S be the graph with two distinct nodes and a unique edge u labeled by l.

Define morphisms $h_{SB} : S \to B$ and $h_{SC} : S \to C$ such that $h_{SB}(u) = b$ and $h_{SC}(u) = c$. We have $h_{SB} \star h_{BD} = h_{SC} \star h_{CD}$. The pushout square $ACDB$ along monomorphisms is also a pullback square by Proposition 2.27, and the universal property of the pullback guarantees the existence of a unique morphism $h_{SA} : S \to A$ such that $h_{SA} \star h_{AB} = h_{SB}$ and $h_{SA} \star h_{AC} = h_{SC}$. Let $a = h_{SA}(u)$. We have $a \in A$, and by composition we obtain:

- $h_{AB}(a) = (h_{SA} \star h_{AB})(u) = h_{SB}(u) = b$, and
- $h_{AC}(a) = (h_{SA} \star h_{AC})(u) = h_{SC}(u) = c$.

The second assertion is a direct consequentce of the first. Suppose, for contradiction, there are $b \in B \setminus h_{AB}(A)$ and $c \in C$ such that $h_{BD}(b) = h_{CD}(c)$. Since the first assertion holds, there must exist $a \in A$ with $h_{AB}(a) = b$ and $h_{AC}(a) = c$. However, this contradicts the assumption that $b \notin h_{AB}(A)$. □ □

Lemma 4.45. *Consider the following commutative diagram in* ***Graph****.*

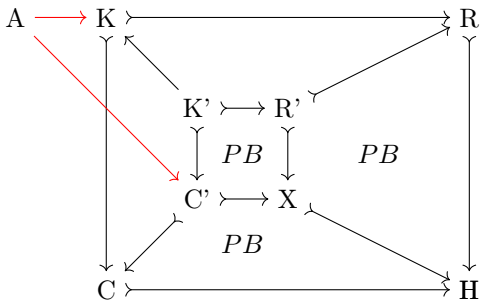


The squares $K'KCC'$ *and* $K'KRR'$ *are pulllback square.*

Proof. Let $K \leftarrow A \rightarrow C'$ be a span such that

$$h_{AK} \star h_{KC} = h_{AC'} \star h_{C'C}. \tag{4.1}$$

We have the following commutative diagram



We show first the existence of a morphism $h_{AK'} : A \to K'$ such that the following statements hold:

$$h_{AK'} \star h_{K'C'} = h_{AC'},$$
$$h_{AK'} \star h_{K'K} = h_{AK}.$$

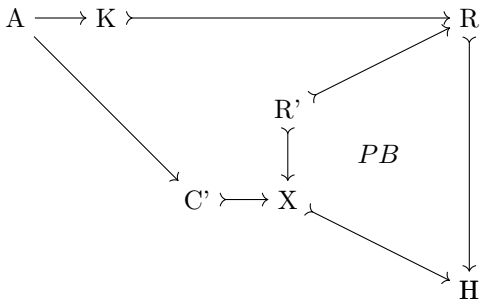
From Equation (4.1), we obtain

$$h_{AK} \star h_{KC} \star h_{CH} = h_{AC'} \star h_{C'C} \star h_{CH}. \tag{4.2}$$

From Equation (4.2) and commutativity of $C'CHX$ and $KCHR$, we have $h_{KC} \star h_{CH} = h_{KR} \star h_{RH}$ and $h_{C'C} \star h_{CH} = h_{C'X} \star h_{XH}$ and deduce

$$h_{AK} \star h_{KR} \star h_{RH} = h_{AC'} \star h_{C'X} \star h_{XH}. \tag{4.3}$$

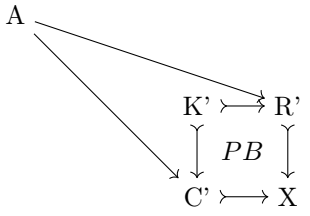
Therefore, the following commutative diagram holds



Since $R'XHR$ is a pullback square, by the universal property Definition 2.24, there is a unique morphism $h_{AR'} : A \to R'$ such that

$$h_{AR'} \star h_{R'R} = h_{AK} \star h_{KR}, \tag{4.4}$$
$$h_{AR'} \star h_{R'X} = h_{AC'} \star h_{C'X}. \tag{4.5}$$

We have thus the following commutative diagram



Since $K'C'XR'$ is a pullback square, there is a unique morphism $h_{AK'} : A \to K'$ such that

$$h_{AK'} \star h_{K'C'} = h_{AC'}, \tag{4.6}$$
$$h_{AK'} \star h_{K'R'} = h_{AR'}. \tag{4.7}$$

We have $h_{AK'} \star h_{K'C'} = h_{AC'}$ by (4.6), and $h_{AK'} \star h_{K'K} = h_{AK}$ because h_{KR} is injective and the following holds:

$$\begin{aligned}(h_{AK'} \star h_{K'K}) \star h_{KR} &= h_{AK} \star h_{K'R'} \star h_{R'R} && \text{by commutativity of } K'KRR' \\ &= h_{AR'} \star h_{R'R} && \text{by (4.7)} \\ &= h_{AK} \star h_{KR}. && \text{by (4.4)}\end{aligned}$$

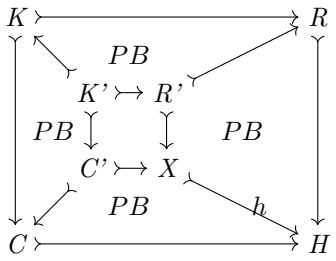
Now, we show the unicity of such a morphism from A to K'. Let $h'_{AK'} : A \to K'$ be a morphism such that

$$\begin{aligned}h'_{AK'} \star h_{K'C'} &= h_{AC'}, \\ h'_{AK'} \star h_{K'K} &= h_{AK}.\end{aligned}$$

From $h_{AK'} \star h_{K'C'} = h_{AC'}$, $h'_{AK'} \star h_{K'C'} = h_{AC'}$ and injectivity of $h_{K'C'}$, we deduce $h_{AK'} = h'_{AK'}$.

□ □

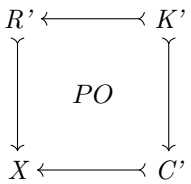
Lemma 4.46. *Consider the following commutative diagram in* ***Graph*** *where $KCHR$ is a pushout.*



The square $K'C'XR'$ is a pushout.

Proof. **Graph** is adhesive by Proposition 4.43. By Definition 4.42, pushouts along monomorphisms are VK squares in **Graph**. The square $KCHR$ is a pushout along monomorphisms in **Graph**. Hence $KCHR$ is VK square. By Definition 4.41, the square $K'C'XR'$ is pushout if the squares $C'CHX$ and $R'XHR$ are pullbacks. By assumption, the squares $C'CHX$ and $R'XHR$ are pullbacks. Therefore, $K'C'XR'$ is pushout. □

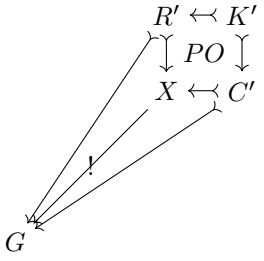
Lemma 4.47. *Consider the following pushout square in* ***Graph****.*



Let x be a node (resp. an edge) in X. There is either a node (resp. an edge) r' in R' such that $h_{R'X}(r') = x$ or a node (resp. an edge) c' in C' such that $h_{C'X}(c') = x$.

Proof. Suppose, by contradiction, that there is neither a node (resp. an edge) r' in R' such that $h_{R'X}(r') = x$ nor a node (resp. an edge) c' in C' such that $h_{C'X}(c') = x$.

Let G be a graph with two distinct nodes (resp. edges) y and y'. By the universal property of pushout square Definition 2.19, there is a unique morphism $h_{XG} : X \rightarrowtail G$ such that $h_{C'G} = h_{C'X} \star h_{XG}$ and $h_{R'G} = h_{R'X} \star h_{XG}$.



However, if we define two morphisms f and g by

$$f(z) = \begin{cases} h_{XG}(z), & \text{if } z \neq x, \\ y, & \text{if } z = x, \end{cases}$$

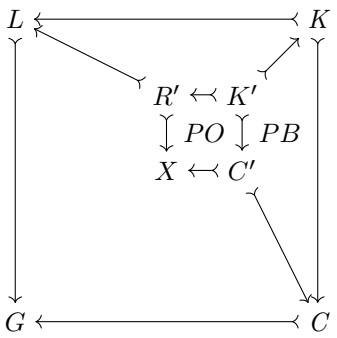
$$g(z) = \begin{cases} h_{XG}(z), & \text{if } z \neq x, \\ y', & \text{if } z = x. \end{cases}$$

we have $f \neq g$ and

$$\begin{aligned} h_{C'G} &= h_{C'X} \star f, \\ h_{R'G} &= h_{R'X} \star f, \\ h_{C'G} &= h_{C'X} \star g, \\ h_{R'G} &= h_{R'X} \star g. \end{aligned}$$

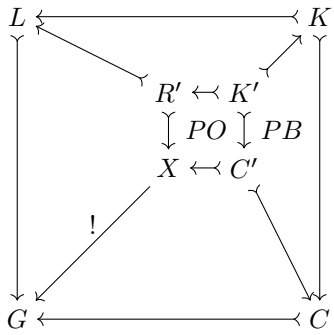
which is a contradiction. □

Lemma 4.48. *Consider the following commutative diagram in* ***Graph****.*



there is a unique monomorphism $h_{XG} : X \rightarrowtail G$ such that $C'XGC$ and $R'XGL$ are commutative squares.

Proof. there is a unique morphism $h_{XG} : X \to G$, by Definition 2.19, because $K'C'XR'$ is a pushout and $h_{K'C'} \star h_{C'C} \star h_{CG} = h_{K'R'} \star h_{R'L} \star h_{LG}$. Therefore, the following commutative diagram holds



It remains to show that h_{XG} is injective.

Let x_1, x_2 be two elements in the graph X such that $x_1 \neq x_2$. Since $K'R'XC'$ is a pushout square, by Lemma 4.47, there are 4 possible cases:

1. there are $c_1, c_2 \in C'$ such that $h_{C'X}(c_1) = x_1$ and $h_{C'X}(c_2) = x_2$;
2. there are $r_1, r_2 \in R'$ such that $h_{R'X}(r_1) = x_1$ and $h_{R'X}(r_2) = x_2$;
3. there are $r' \in R'$ and $c' \in C'$ such that $h_{C'X}(c') = x_1$ and $h_{R'X}(r') = x_2$;
4. there are $r' \in R'$ and $c' \in C'$ such that $h_{C'X}(c') = x_2$ and $h_{R'X}(r') = x_1$.

Since the second and fourth cases are symmetric to the first and third case, respectively, it suffices to show that in the first and third cases, the inequality $h_{XG}(x_1) \neq h_{XG}(x_2)$ holds.

Case (1) Suppose that there are $c_1, c_2 \in C'$ such that $h_{C'X}(c_1) = x_1$ and $h_{C'X}(c_2) = x_2$.

We have $c_1 \neq c_2$ since $x_1 \neq x_2$ and $h_{C'X}$ is injective.

Therefore we have

$$
\begin{aligned}
h_{XG}(x_1) &= h_{XG}(h_{C'X}(c_1)) && \text{by definition of } c_1 \\
&= (h_{C'X} \star h_{XG})(c_1) \\
&= (h_{C'C} \star h_{CG})(c_1) && \text{by commutativity of } C'CGX \\
&\neq (h_{C'C} \star h_{CG})(c_2) && \text{by injectivity of } h_{C'C} \star h_{CG} \text{ and } c_1 \neq c_2 \\
&= (h_{C'X} \star h_{XG})(c_2) && \text{by commutativity of } C'CGX \\
&= h_{XG}(h_{C'X}(c_2)) \\
&= h_{XG}(x_2). && \text{by definition of } c_2
\end{aligned}
$$

Case (3) Suppose that there are $r' \in R'$ and $c' \in C'$ such that $h_{C'X}(c') = x_1$ and $h_{R'X}(r') = x_2$.

We have three cases:

(3.1) Suppose that $h_{C'C}(c') \notin \mathrm{Im}(h_{KC})$. We have

$$\begin{aligned}
h_{XG}(x_1) &= h_{XG}(h_{C'X}(c')) && \text{by definition of } c' \\
&= (h_{C'X} \star h_{XG})(c') \\
&= (h_{C'C} \star h_{CG})(c') && \text{by commutativity of } C'CGX \\
&= h_{CG}(h_{C'C}(c')) \\
&\neq h_{LG}(h_{R'L}(r')) && \text{by Lemma 4.44 and } h_{C'C}(c') \notin \mathrm{Im}(h_{KC}) \\
&= (h_{R'L} \star h_{LG})(r') \\
&= (h_{R'X} \star h_{XG})(r') && \text{by commutativity of } R'XGL \\
&= h_{XG}(h_{R'X}(r')) \\
&= h_{XG}(x_2). && \text{by definition of } r'
\end{aligned}$$

(3.2) Suppose $h_{R'L}(r') \notin \mathrm{Im}(h_{KL})$. This case is symmetric to the previous case.

(3.3) Suppose that there are $k_1, k_2 \in K$ such that $h_{KC}(k_1) = h_{C'C}(c')$ and $h_{KL}(k_2) = h_{R'L}(r')$.
We cannot have $k_1 = k_2$.
Suppose, by contradiction, that we have

$$k_1 = k_2. \tag{4.8}$$

Since $K'C'CK$ is also a pullback square by Proposition 2.27 and $h_{KC}(k_1) = h_{C'C}(c')$ and $h_{C'C}(c') = h_{C'C}(c')$, there is $k' \in K'$ such that $h_{K'K}(k') = k_1$ and $h_{K'C'}(k') = c'$.
We have $h_{K'R'}(k') = r'$ because $h_{R'L}$ is injective, $h_{R'L}(r') = h_{R'L}(r')$ and

$$\begin{aligned}
& h_{R'L}(h_{K'R'}(k')) \\
=& (h_{K'R'} \star h_{R'L})(k') \\
=& (h_{K'K} \star h_{KL})(k') && \text{by commutativity of } K'R'LK \\
=& (h_{KL})(h_{K'K}(k')) \\
=& (h_{KL})(k_1) \\
=& (h_{KL})(k_2). && \text{by assumption (4.8)}
\end{aligned}$$

Therefore, we have the contradiction:

$$\begin{aligned}
x_1 &= h_{C'X}(c') && \text{by definition of } c' \\
&= h_{C'X}(h_{K'C'}(k')) && \text{by definition of } k' \\
&= (h_{K'C'} \star h_{C'X})(k') \\
&= (h_{K'R'} \star h_{R'X})(k') && \text{by the commutativity of } K'C'XR' \\
&= h_{R'X}(h_{K'R'}(k')) \\
&= h_{R'X}(r') && \text{by definition of } r' \\
&= x_2. && \text{by definition of } x_2
\end{aligned}$$

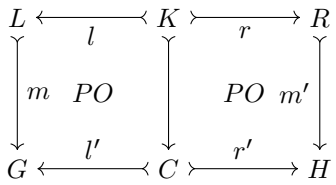
Thus, the following inequality holds

$$k_1 \neq k_2. \tag{4.9}$$

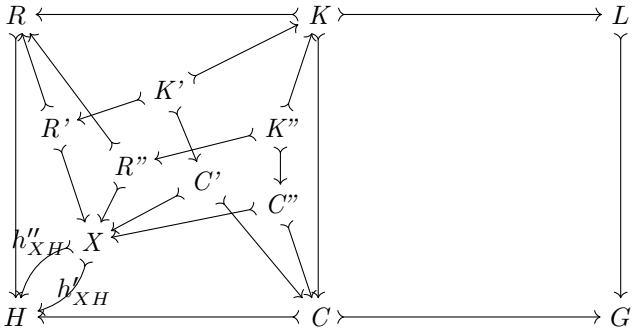
$$
\begin{aligned}
h_{XG}(x_1) &= h_{XG}(h_{C'X}(c')) && \text{by definition of } c' \\
&= (h_{C'X} \star h_{XG})(c') \\
&= (h_{C'C} \star h_{CG})(c') && \text{by commutativity of } C'CGX \\
&= (h_{CG})(h_{KC}(k_1)) && \text{by definition of } k_1 \\
&= (h_{KC} \star h_{CG})(k_1) \\
&\neq (h_{KC} \star h_{CG})(k_2) && \text{by injectivity of } h_{KC} \star h_{CG} \text{ and (4.9)} \\
&= (h_{KL} \star h_{LG})(k_2) && \text{by commutativity of } KLGC \\
&= h_{LG}(h_{KL}(k_2)) \\
&= h_{LG}(h_{R'L}(r')) && \text{by the definition of } k_2 \\
&= (h_{R'L} \star h_{LG})(r') \\
&= (h_{R'X} \star h_{XG})(r') && \text{by commutativity of } R'XLG \\
&= h_{XG}(h_{R'X}(r')) \\
&= h_{XG}(x_2). && \text{by definition of } r'
\end{aligned}
$$

□

Lemma 4.49. *Let X be a graph. Let $\rho = (L \overset{l}{\leftarrowtail} K \overset{r}{\rightarrowtail} R)$ be an injective DPO rewriting rule which is X-non-increasing under Ψ. Consider the following DPO diagram:*



*Let $h'_{XH}, h''_{XH} : X \rightarrowtail H$ be monomorphisms. In the category **Graph**, we can construct the following commutative diagram*



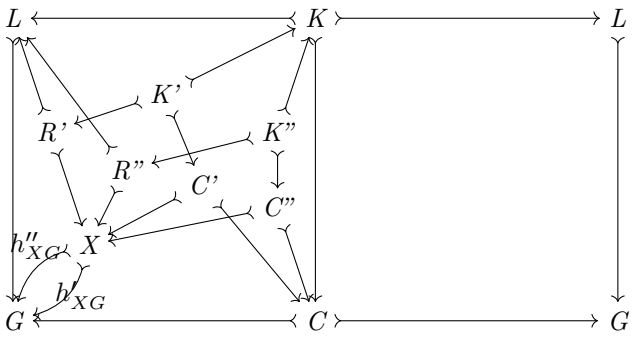
where $K'R'XC'$, $K''R''XC''$, $K'KCC'$, $K''KCC''$, $R'RHX$ and $R''RHX$ are pullbacks.

By Lemma 4.45, $K'KRR'$ and $K''KCC''$ are pullbacks.

By Lemma 4.46, $K'R'XC'$ and $K''R''XC''$ are pushouts.

Since ρ is X-non-increasing under Ψ, there are $\Psi(R') : R' \nrightarrow L$ and $\Psi(R'') : R'' \nrightarrow L$ such that $K'R'LK$ and $K''R''LK$ are pullbacks.

By Lemma 4.48, we have the following commutative diagram



where

- *$K'R'XC'$ is pushout and pullback;*
- *$K''R''XC''$ is pushout and pullback;*
- *$K'KCC'$ and $K''KCC''$ are pullback;*
- *$h'_{XG} : X \to G$ is the unique morphism such that $h_{C'C} \star h_{CG} = h_{C'X} \star h'_{XG}$ and $h_{R'L} \star h_{LG} = h_{R'X} \star h'_{XG}$ where $h_{R'L} = \Psi(R')$;*
- *$h''_{XG} : X \to G$ is the unique morphism such that $h_{C''C} \star h_{CG} = h_{C''X} \star h''_{XG}$ and $h_{R''L} \star h_{LG} = h_{R''X} \star h''_{XG}$ where $h_{R''L} = \Psi(R'')$.*

If $h'_{XH} \neq h''_{XH}$ then $h'_{XG} \neq h''_{XG}$.

Proof. Suppose that $h'_{XH} \neq h''_{XH}$. We are going to show $h'_{XG} \neq h''_{XG}$.

From $h'_{XH} \neq h''_{XH}$, we deduce there is an element $x \in X$ such that the inequality $h_{XH}(x) \neq h''_{XH}(x)$ holds. We can suppose that

$$x \text{ is either an isolated nodes or an edge,} \tag{4.10}$$

because otherwise x would be a non-isolated node and we can take an incident edge of x.

Since $K'R'XC'$ and $K''R''XC''$ are pushouts, by Lemma 4.47, there are 4 possible cases:

1. there are $c' \in C', c'' \in C''$ such that $h_{C'X}(c') = h_{C'X}(c'') = x$;
2. there are $r' \in R', r'' \in R''$ such that $h_{R'X}(r') = h_{R'X}(r'') = x$;
3. there are arrows $r' \in R'$ and $c'' \in C''$ such that

$$\begin{aligned} h_{C''X}(c'') &= x \\ h_{R'X}(r') &= x \\ \nexists c' \in C'.h_{C'X}(c') &= x \\ \nexists r'' \in R''.h_{R''X}(r'') &= x; \end{aligned}$$

4. there are arrows $r'' \in R''$ and $c' \in C'$ such that

$$\begin{aligned} h_{C'X}(c') &= x \\ h_{R''X}(r'') &= x \\ \nexists c'' \in C''.h_{C''X}(c'') &= x \\ \nexists r' \in R'.h_{R'X}(r') &= x. \end{aligned}$$

We are going to show that in each case we have $h'_{XG}(x) \neq h''_{XG}(x)$. Since the fourth case is symmetric to the third case, it suffices to analyse the first three cases.

(1) Suppose that there are $c' \in C', c'' \in C''$ such that $h_{C'X}(c') = h_{C'X}(c'') = x$. Suppose, by contradiction, that the following equality holds

$$h_{C'C}(c') = h_{C''C}(c''). \tag{4.11}$$

We have the following contradiction:

$$\begin{aligned} h'_{XH}(x) =& h'_{XH}(h_{C'X}(c')) \\ =& (h_{C'X} \star h'_{XH})(c') \\ =& (h_{C'C} \star h_{CH})(c') && \text{By commutativity of } C'XHC \\ =& h_{CH}(h_{C'C}(c')) \\ =& h_{CH}(h_{C''C}(c'')) && \text{by (4.11)} \\ =& (h_{C''C} \star h_{CH})(c'') \\ =& (h_{C''X} \star h''_{XH})(c'') && \text{By commutativity of } C''CHX \\ =& h''_{XH}(h_{C''X}(c'')) \\ =& h''_{XH}(x). \end{aligned}$$

Therefore, we have

$$h_{C'C}(c') \neq h_{C''C}(c''). \tag{4.12}$$

$$\begin{aligned} h'_{XG}(x) &= h'_{XG}(h_{C'X}(c')) && \text{by definition of } c' \\ &= (h_{C'X} \star h'_{XG})(c') \\ &= (h_{C'C} \star h_{CG})(c') && \text{by commutativity of } C'CGX \\ &= h_{CG}(h_{C'C}(c')) \\ &\neq h_{CG}(h_{C''C}(c'')) && \text{by (4.12) and the injectivity of } h_{CG} \\ &= (h_{C''C} \star h_{CG})(c'') \\ &= (h_{C''X} \star h''_{XG})(c'') && \text{by commutativity of } C''CGX \\ &= h''_{XG}(h_{C''X}(c'')) \\ &= h''_{XG}(x). && \text{by definition of } c'' \end{aligned}$$

(2) Suppose that there are $r' \in R', r'' \in R''$ such that $h_{R'X}(r') = h_{R'X}(r'') = x$. We have

$$h_{R'R}(r') \neq h_{R''R}(r'') \tag{4.13}$$

by an argument analogous to the argument for $h_{C'C}(c') \neq h_{C''C}(c'')$ in the previous case. Since ρ is X-non-increasing, by (4.13) and (4.10), we have

$$h_{R'L}(r') \neq h_{R''L}(r''). \tag{4.14}$$

Hence, the following inequality holds

$$\begin{aligned}
h'_{XG}(x) &= h'_{XG}(h_{R'X}(r')) && \text{by definition of } r' \\
&= (h_{R'X} \star h'_{XG})(r') \\
&= (h_{R'L} \star h_{LG})(r') && \text{by commutativity of } R'XGL \\
&= h_{LG}(h_{R'L}(r')) \\
&\neq h_{LG}(h_{R''L}(r'')) && \text{By injectivity of } h_{LG} \text{ and (4.14)} \\
&= (h_{R''L} \star h_{LG})(r'') \\
&= (h_{R''X} \star h''_{XG})(r'') && \text{by commutativity of } R''XGL \\
&= h''_{XG}(h_{R''X}(r'')) \\
&= h''_{XG}(x). && \text{by definition of } r''
\end{aligned}$$

(3) Suppose, that there are arrows $r' \in R'$ and $c'' \in C''$ such that the following statements hold:

$$\begin{aligned}
h_{C''X}(c'') &= x, \\
h_{R'X}(r') &= x, \\
\nexists c' \in C'.h_{C'X}(c') &= x, \\
\nexists r'' \in R''.h_{R''X}(r'') &= x.
\end{aligned} \tag{4.15}$$

Suppose, by contradiction, that we have

$$h_{C''C}(c'') \in \mathrm{Im}(h_{KC}). \tag{4.16}$$

Then, there are $k_2 \in K$ such that $h_{KC}(k_2) = h_{C''C}(c'')$.

There is $k'' \in K''$ such that $h_{K''K}(k'') = k_2$ and $h_{K''C''}(k'') = c''$ becase $K''C''CK$ is a pullback square and $h_{KC}(k_2) = h_{C''C}(c'')$.

We have $h_{K''R''}(k'') \in R''$ and

$$\begin{aligned}
h_{R''X}(h_{K''R''}(k'')) =& (h_{K''R''} \star h_{R''X})(k'') \\
=& (h_{K''C''} \star h_{C''X})(k'') && \text{by commutativity of } K''R''XC'' \\
=& h_{C''X}(h_{K''C''}(k'')) \\
=& h_{C''X}(c'') \\
=& x.
\end{aligned}$$

which contradicts the assumption (4.15).

Thus, the following holds

$$h_{C''C}(c'') \notin \mathrm{Im}(h_{KC}). \tag{4.17}$$

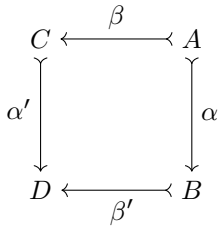
We deduce

$$\begin{aligned}
h'_{XG}(x) &= h'_{XG}(h_{C''X}(c'')) && \text{by definition of } c''\\
&= (h_{C''X} \star h'_{XG})(c'')\\
&= (h_{C''C} \star h_{CG})(c'') && \text{by commutativity of } C''CGX\\
&= h_{CG}(h_{C''C}(c''))\\
&\neq h_{LG}(h_{R'L}(r')) && \text{by Lemma 4.44 and the assumption (4.17)}\\
&= (h_{R'L} \star h_{LG})(r')\\
&= (h_{R'X} \star h''_{XG})(r') && \text{by commutativity of } R'XGL\\
&= h''_{XG}(h_{R'X}(r'))\\
&= h''_{XG}(x). && \text{by definition of } r'
\end{aligned}$$

□

Proofs of Lemma 4.13, Lemma 4.14, Lemma 4.24 and Lemma 4.25

Lemma 4.13 Let X be a ruler-graph. For a pushout square as shown below:



the following equalities hold:

$$\begin{aligned}
|\mathrm{Mono}(X, B)| &= |\mathrm{Mono}(X, D, \beta')|,\\
|\mathrm{Mono}(X, C, \neg\beta)| &= |\mathrm{Mono}(X, D, \neg\beta', \alpha')|.
\end{aligned}$$

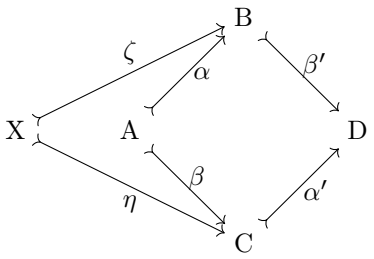
Proof.

Claim 4.50. $|\mathrm{Mono}(X, B)| = |\mathrm{Mono}(X, D, \beta')|$ *holds.*

- The inclusion $\mathrm{Mono}(X, B) \star \beta' \subseteq \mathrm{Mono}(X, D, \beta')$ holds by the definitions of $\mathrm{Mono}(X, B)$ and $\mathrm{Mono}(X, D, \beta')$. In fact, if $\iota \in \mathrm{Mono}(X, B) \star \beta'$, then there is a monomorphism $\zeta : X \rightarrowtail B$ satisfying $\iota = \zeta \star \beta'$. Since $\zeta \star \beta'$ is also a monomorphism, ι is also an element of $\mathrm{Mono}(X, D, \beta')$.
- Inclusion $\mathrm{Mono}(X, B) \star \beta' \supseteq \mathrm{Mono}(X, D, \beta')$ holds by the definition of $\mathrm{Mono}(X, B)$ and $\mathrm{Mono}(X, D, \beta')$. Suppose $\iota \in \mathrm{Mono}(X, D, \beta')$. There is a monomorphism $\zeta : X \rightarrowtail B$ satisfying $\iota = \zeta \star \beta'$. This shows that ι is an element of $\mathrm{Mono}(X, B) \star \beta'$.
- Hence, we have $\mathrm{Mono}(X, B) \star \beta' = \mathrm{Mono}(X, D, \beta')$.
- $|\mathrm{Mono}(X, B)| \leq |\mathrm{Mono}(X, D, \beta')|$ follows by the injectivity of β'.

Claim 4.51. $|\mathrm{Mono}(X, C, \neg\beta)| = |\mathrm{Mono}(X, D, \neg\beta', \alpha')|$ *holds.*

- The inclusion $\mathrm{Mono}(X, C, \neg\beta) \star \alpha' \subseteq \mathrm{Mono}(X, D, \neg\beta', \alpha')$ can be justified as follows: Suppose $\eta \star \alpha' \in \mathrm{Mono}(X, C, \neg\beta) \star \alpha'$. Since $\eta \star \alpha'$ is a monomorphism, it suffices to show that there is no $\zeta : X \nrightarrow B$ such that $\eta \star \alpha' = \zeta \star \beta'$. Suppose, by contradiction, that such a ζ exists, then the following commutative diagram holds:



  The pushout square $\square ABDC$ is also a pullback square, by Proposition 2.27. The universal property of the pullback provides a morphism $\gamma : X \to A$ such that $\eta = \gamma \star \beta$. γ is a monomorphism, because $\eta = \gamma \star \beta$ and η is a monomorphism. Therefore, the existence of γ contradicts the assumption that $\eta \in \mathrm{Mono}(X, C, \neg\beta)$. Thus, ι is also an element of $\mathrm{Mono}(X, D, \neg\beta', \alpha')$.

- The inclusion $\mathrm{Mono}(X, C, \neg\beta) \star \alpha' \supseteq \mathrm{Mono}(X, D, \neg\beta', \alpha')$ can be justified as follows. Suppose that $\iota : X \nrightarrow D$ is an element of $\mathrm{Mono}(X, D, \neg\beta', \alpha')$. According to the definition of $\mathrm{Mono}(X, D, \neg\beta', \alpha')$, there is $\eta : X \nrightarrow C$ making

$$\iota = \eta \star \alpha'. \tag{4.18}$$

  We show that η is an element of $\mathrm{Mono}(X, C, \neg\beta)$ by contradiction.

  Suppose that there is a monomorphism $\zeta : X \nrightarrow A$ such that

$$\eta = \zeta \star \beta. \tag{4.19}$$

  We have:

$$\begin{aligned}
\iota &\overset{\text{def}}{=} \eta \star \alpha' && \text{by (4.18)} \\
&\overset{\text{def}}{=} (\zeta \star \beta) \star \alpha' && \text{by (4.19)} \\
&= \zeta \star (\beta \star \alpha') && \text{by associativity} \\
&= \zeta \star (\alpha \star \beta') && \text{by commutativity of } ABDC \\
&= (\zeta \star \alpha) \star \beta'.
\end{aligned}$$

  There is a contradiction because no such factorization of ι should exist as $\iota \in \mathrm{Mono}(X, D, \neg\beta', \alpha')$.

- We conclude by the monicity of α'.

□

Lemma 4.14 Let X be a graph. Let $L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R$ be an injective DPO graph rewriting rule. We have

$$|\mathrm{Mono}(X, L, \neg l)| - |\mathrm{Mono}(X, R, \neg r)| = |\mathrm{Mono}(X, L)| - |\mathrm{Mono}(X, R)|.$$

Proof.

Claim 4.52. $|\mathrm{Mono}(X,L,l)| = |\mathrm{Mono}(X,R,r)|$ *holds.*

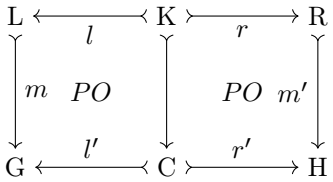
- We prove $|\mathrm{Mono}(X,L,l)| \leq |\mathrm{Mono}(X,R,r)|$ by constructing an injection from $\mathrm{Mono}(X,L,l)$ to $\mathrm{Mono}(X,R,r)$. Let $h \in \mathrm{Mono}(X,L,l)$. By definition, we have $h = g \star l$ for some $g : X \nrightarrow K$. Thus, $g \star r \in \mathrm{Mono}(X,L,r)$. Let $h' \in \mathrm{Mono}(X,L,l)$ such that $h \neq h'$. We have $h' = g' \star l$ for some $g' : X \nrightarrow K$. We have $g \neq g'$ because otherwise we would have $h = g \star l = g' \star l = h'$. From the monicity of r, we conclude $g \star r \neq g' \star r$.
- Analoguously, we can prove $|\mathrm{Mono}(X,L,l)| \geq |\mathrm{Mono}(X,R,r)|$ by constructing an injection from $\mathrm{Mono}(X,R,r)$ to $\mathrm{Mono}(X,L,l)$.

Hence, the following equality holds

$$
\begin{aligned}
&|\mathrm{Mono}(X,L,\neg l)| - |\mathrm{Mono}(X,R,\neg r)| \\
=&|\mathrm{Mono}(X,L) \setminus \mathrm{Mono}(X,L,l)| - |\mathrm{Mono}(X,R) \setminus \mathrm{Mono}(X,R,r)| \\
=&(|\mathrm{Mono}(X,L)| - |\mathrm{Mono}(X,L,l)|) - (|\mathrm{Mono}(X,R)| - |\mathrm{Mono}(X,R,r)|) \\
=&(|\mathrm{Mono}(X,L)| - |\mathrm{Mono}(X,R)|) - (|\mathrm{Mono}(X,L,l)| - |\mathrm{Mono}(X,R,r)|) \\
=&|\mathrm{Mono}(X,L)| - |\mathrm{Mono}(X,R)|.
\end{aligned}
$$

□

Lemma 4.24 Let X be a ruler-graph and $\rho = (L \overset{l}{\hookleftarrow} K \overset{r}{\nrightarrow} R)$ an injective DPO rewriting rule. Suppose that ρ is X-non-increasing. For every rewriting step induced by the following DPO diagram:

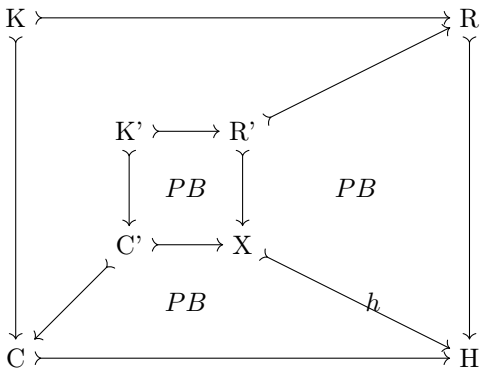


The following inequality holds:

$$|\mathrm{Mono}(X,G,\neg m,\neg l')| \geq |\mathrm{Mono}(X,H,\neg m',\neg r')|$$

Proof. We are going to show that for two arbitrary distinct monomorphisms $h'_{XH}, h''_{XH} \in \mathrm{Mono}(X,H,\neg m',\neg r')$, there are two distinct monomorphisms h'_{XG} and $h''_{XG} \in \mathrm{Mono}(X,G,\neg m,\neg l')$.

Let $h'_{XH} \in \mathrm{Mono}(X,H,\neg m',\neg r')$ be a monomorphism. In the category **Graph**, we can construct the following commutative diagram

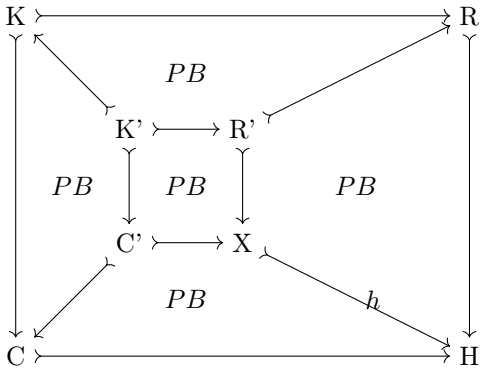


$KCHR$ is also pullback, by Proposition 2.27, because it is pushout. Therefore, since $KCHR$ is also pullback and $h_{K'C'} \star h_{C'C} \star h_{CH} = h_{K'R'} \star h_{R'R} \star h_{R'R}$ holds, there is $h_{K'K} : K' \nrightarrow K$ such that the following equalities hold:

$$h_{K'C'} \star h_{C'C} = h_{K'K} \star h_{KC}, \tag{4.20}$$

$$h_{K'R'} \star h_{R'R} = h_{K'K} \star h_{KR}. \tag{4.21}$$

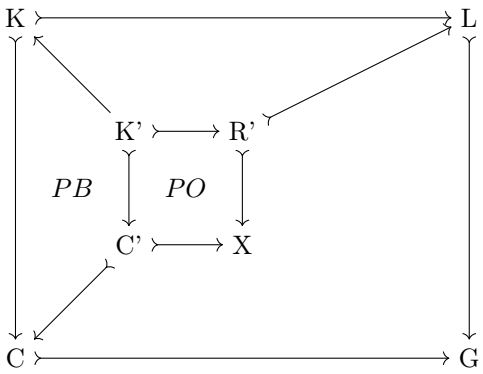
The square $K'KCC'$ and $K'R'RK$ are pullbacks, by Lemma 4.45. Hence, the following commutative diagram holds



Since the rule is by assumption X-non-increasing, there is a morphism $h_{R'L}$ such that $K'KLR'$ is commutative.

$K'C'XR'$ is a pushout by Lemma 4.46.

Thus, the following commutative diagram holds



By Lemma 4.48, there is a unique monomorphism $h'_{XG} : X \rightarrowtail G$ such that $h_{C'X} \star h'_{XG} = h_{C'C} \star h_{CG}$ and $h_{R'X} \star h'_{XG} = h_{R'L} \star h_{LG}$.

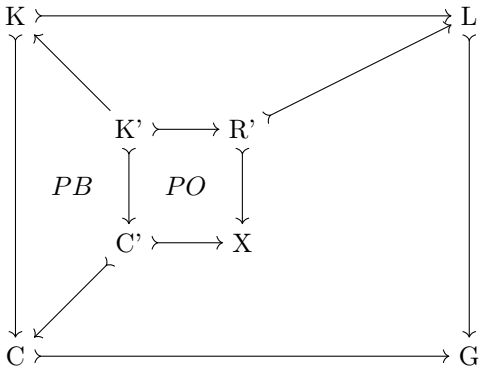
By $h'_{XH} \in \text{Mono}(X, H, \neg m', \neg r')$ and Definition 4.18, the monomorphism h'_{XG} is in the set $\text{Mono}(X, G, \neg m, \neg l')$. Intuitively, some elements in X are mapped onto $h_{RH}(R) \setminus (r \star h_{RH})(K)$ if h'_{XH} is an element of $\text{Mono}(X, H, \neg m', \neg r')$. By the first assumption of Definition 4.18, these elements are mapped onto $h_{LG}(L) \setminus (l \star h_{LG})(K)$.

The monomorphism h'_{XG} belongs to the set $\text{Mono}(X, G, \neg m, \neg l')$, due to $h'_{XH} \in \text{Mono}(X, H, \neg m', \neg r')$ and Definition 4.18. Intuitively, h'_{XH} is an element of $\text{Mono}(X, H, \neg m', \neg r')$ implies that certain elements of X are mapped into $h_{RH}(R) \setminus (r \star h_{RH})(K)$. By the second assumption of Definition 4.18, these elements are also mapped into $h_{LG}(L) \setminus (l \star h_{LG})(K)$.

Let $h''_{XH} : X \rightarrowtail H$ be a monomorphism such that

$$h'_{XH} \neq h''_{XH}. \tag{4.22}$$

In the category **Graph**, we can construct the following commutative diagram:



Analoguously, there is a unique monomorphism $h''_{XG} : X \rightarrowtail G$ such that $h_{C''X} \star h''_{XG} = h_{C''C} \star h_{CG}$ and $h_{R''X} \star h''_{XG} = h_{R''L} \star h_{LG}$.

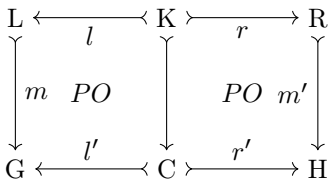
By (4.22) and Lemma 4.49, we have $h_{XG} \neq h'_{XG}$. □

Lemma 4.25[Decreasing step] Let $\rho = (L \overset{l}{\hookleftarrow} K \overset{r}{\rightarrowtail} R)$ be an injective DPO rewriting rule, $\mathbb{X}$ a set of ruler-graphs, $s_{\mathbb{X}} : \mathbb{X} \to \mathbb{N}$ a weight function, and $G \Rightarrow_{\rho,\mathfrak{M}} H$

a rewriting step. If ρ is X-non-increasing for every ruler-graph $X \in \mathbb{X}$, then:

$$w_{s_{\mathbb{X}}}(G) - w_{s_{\mathbb{X}}}(H) \geq w_{s_{\mathbb{X}}}(L) - w_{s_{\mathbb{X}}}(R).$$

Proof. Let the following diagram be the witness diagram of the rewriting step



Let $X \in \mathbb{X}$. We have

$$\begin{aligned}
&|\mathrm{Mono}(X, G)| \\
=&|\mathrm{Mono}(X, G, l') \uplus \mathrm{Mono}(X, G, \neg l', m) \uplus \mathrm{Mono}(X, G, \neg l', \neg m)| \\
=&|\mathrm{Mono}(X, G, l')| + |\mathrm{Mono}(X, G, \neg l', m)| + |\mathrm{Mono}(X, G, \neg l', \neg m)| \\
=&|\mathrm{Mono}(X, C)| + |\mathrm{Mono}(X, L, \neg l)| + |\mathrm{Mono}(X, G, \neg l', \neg m)| && \text{by Lemma 4.13}
\end{aligned}$$

and

$$\begin{aligned}
&|\mathrm{Mono}(X, H)| \\
=&|\mathrm{Mono}(X, H, r') \uplus \mathrm{Mono}(X, H, \neg r', m') \uplus \mathrm{Mono}(X, H, \neg r', \neg m')| \\
=&|\mathrm{Mono}(X, H, r')| + |\mathrm{Mono}(X, H, \neg r', m')| + |\mathrm{Mono}(X, H, \neg r', \neg m')| \\
=&|\mathrm{Mono}(X, C)| + |\mathrm{Mono}(X, R, \neg r)| + |\mathrm{Mono}(X, H, \neg r', \neg m')|. && \text{by Lemma 4.13}
\end{aligned}$$

Therefore, the following inequality holds

$$\begin{aligned}
&|\mathrm{Mono}(X, G)| - |\mathrm{Mono}(X, H)| \\
=&(|\mathrm{Mono}(X, L, \neg l)| - |\mathrm{Mono}(X, R, \neg r)|) + \\
&(|\mathrm{Mono}(X, G, \neg l', \neg m)| - |\mathrm{Mono}(X, H, \neg r', \neg m')|) \\
\geq&|\mathrm{Mono}(X, L, \neg l)| - |\mathrm{Mono}(X, R, \neg r)| && \text{by Lemma 4.24} \\
=&|\mathrm{Mono}(X, L)| - |\mathrm{Mono}(X, R)|. && \text{by Lemma 4.14}
\end{aligned}$$

Thus, the following inequality holds

$$\begin{aligned}
&w_{s_{\mathbb{X}}}(G) - w_{s_{\mathbb{X}}}(H) \\
=&\sum_{X \in \mathbb{X}} w(X) * |\mathrm{Mono}(X, G)| - \sum_{X \in \mathbb{X}} w(X) * |\mathrm{Mono}(X, H)| \\
=&\sum_{X \in \mathbb{X}} w(X) * (|\mathrm{Mono}(X, G)| - |\mathrm{Mono}(X, H)|) \\
\geq&\sum_{X \in \mathbb{X}} w(X) * (|\mathrm{Mono}(X, L)| - |\mathrm{Mono}(X, R)|) \\
=&\sum_{X \in \mathbb{X}} w(X) * |\mathrm{Mono}(X, L)| - \sum_{X \in \mathbb{X}} w(X) * |\mathrm{Mono}(X, R)| \\
=&w_{s_{\mathbb{X}}}(L) - w_{s_{\mathbb{X}}}(R).
\end{aligned}$$

□

Theorem4.26(Termination) Let $\mathcal{A}$ and $\mathcal{B}$ be sets of injective DPO rewriting rules, $\mathbb{X}$ a set of ruler-graphs and $s_{\mathbb{X}}$ a weight function. If the following conditions hold:

1. ρ is X-non-increasing for every rule $\rho \in \mathcal{A} \cup \mathcal{B}$ and for every ruler-graph $X \in \mathbb{X}$,
2. $w_{s_{\mathbb{X}}}(lhs(\rho)) > w_{s_{\mathbb{X}}}(rhs(\rho))$ for every rule $\rho \in \mathcal{A}$,
3. $w_{s_{\mathbb{X}}}(lhs(\rho)) \geq w_{s_{\mathbb{X}}}(rhs(\rho))$ for every rule $\rho \in \mathcal{B}$.

Then $\Rightarrow_{\mathcal{A},\mathcal{M}}$ terminates relative to $\Rightarrow_{\mathcal{B},\mathcal{M}}$.

Proof. By assumption (1) and Lemma 4.25, the following inequality holds for all rules $\rho \in \mathcal{A} \cup \mathcal{B}$ and for all rewriting steps $G \Rightarrow_{\rho,\mathcal{M}} H$:

$$w_{s_{\mathbb{X}}}(G) - w_{s_{\mathbb{X}}}(H) \geq w_{s_{\mathbb{X}}}(\mathrm{lhs}(\rho)) - w_{s_{\mathbb{X}}}(\mathrm{rhs}(\rho)).$$

From assumptions (2) and (3), we deduce

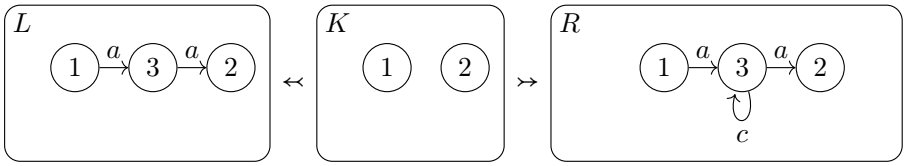
- $w_{s_{\mathbb{X}}}(G) > w_{s_{\mathbb{X}}}(H)$ for every rule $\rho \in \mathcal{A}$;
- $w_{s_{\mathbb{X}}}(G) \geq w_{s_{\mathbb{X}}}(H)$ for every rule $\rho \in \mathcal{B}$.

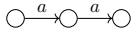
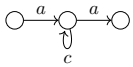
Since $w_{s_{\mathbb{X}}}(G) \in \mathbb{N}$ for all graph G. Every rewriting chain can only have a finite number of rewriting steps using rules in $\mathcal{A}$. □

Chapter 5

Termination of Injective DPO Graph Rewriting Systems using Morphism Counting with antipatterns

Consider the following DPO rewriting rule:



In this system, the number of monomorphisms from every pattern graph increases with each application of the rewriting rule. Therefore, the method introduced in Chapter 4 cannot be applied. However, the number of monomorphisms from the graph  whose images are not included in occurrences of the graph  strictly decreases with each application of the rewriting rule.

We formalize this intuition by using the concept of *forbidden contexts* and extend the morphism counting method presented in Chapter 4 to count only those monomorphisms whose images are not included in any forbidden contexts.

This extension can handle systems (e.g., Example 5.11 and Example 5.12) that cannot be handled by the prior interpretation-based approaches [85, 19, 17, 38, 66] and by the morphism counting method presented in Chapter 4. An implementation of this extension is described in Chapter 6.

This chapter is organized as follows: § 5.1 presents the extension, and § 5.2 provides some examples. Proofs of some propositions, lemmas, and theorems of this chapter have been moved to Appendix 5.5 to improve readability.

5.1 A Termination Criterion

We generalize the concept of the ruler-graph presented in Chapter 4.

Definition 5.1. *A **ruler-graph** is an order pair $(X, \mathcal{C})$ where X is the **underlying graph** and $\mathcal{C}$ is a set of monomorphisms from X with $|\mathcal{C}| \leq 1$. Each $f \in \mathcal{C}$ is called a **forbidden context**.*

In this chapter we restrict attention to the case $|\mathcal{C}| \leq 1$, and defer an analysis for general finite $\mathcal{C}$ to future work.

For $f : X \rightarrowtail F \in \mathcal{C}$, we also call F a forbidden context for simplicity. The definition of measurement needs some modifications to take the forbidden context into account. When $\mathcal{C} = \varnothing$, the definition of measurement is the same as that in Definition 4.9 in Chapter 4. When $\mathcal{C} = \{f\}$ with $f : X \rightarrowtail F$, we exclude those monomorphisms whose images overlap with the forbidden context. These excluded monomorphisms are those that can be extended to a morphism $g : F \rightarrowtail G$. This leads us to the following definition.

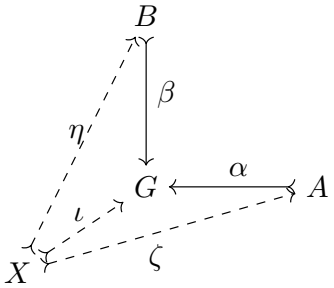
Definition 5.2. *Let $\mathcal{X} = (X, \mathcal{C})$ be a ruler-graph and G a graph. The **measurement** of G with respect to $\mathcal{X}$, denoted $m_{\mathcal{X}}(G)$, is defined as*

$$m_{\mathcal{X}}(G) \stackrel{\text{def}}{=} |\{h \in \mathrm{Mono}(X, G) \mid \forall f \in \mathcal{C}. \nexists g \in \mathrm{Mono}(F, G).\ f \star g = h\}|$$

The definitions of weight function and graph weights are identical to those in Definition 4.10 and Definition 4.11 in Chapter 4, respectively, except that we use the new definition of measurement.

We define the following sets of monomorphisms to facilitate the discussion of the termination criterion. They are very similar to those defined in Notation 4.12.

Notation 5.3. *Let $\mathcal{X}$ be a ruler-graph with underlying graph X. The disjoint union of two sets S and S' is denoted by $S \uplus S'$. Let X, A, B, G be graphs, and let $\alpha : A \to G$ and $\beta : B \to G$ be morphisms as illustrated below:*



We define the following sets of monomorphisms from X into G based on their

interaction with α and β:

$$\begin{aligned}
\mathrm{Mono}(\mathcal{X}, G) &= \mathrm{Mono}(X, G),\\
\mathrm{Mono}(\mathcal{X}, G, \alpha) &= \{\iota : X \rightarrowtail G \mid \exists \zeta : X \rightarrowtail A.\, \iota = \zeta \star \alpha\},\\
\mathrm{Mono}(\mathcal{X}, G, \neg\alpha) &= \{\iota : X \rightarrowtail G \mid \nexists \zeta : X \rightarrowtail A.\, \iota = \zeta \star \alpha\},\\
\mathrm{Mono}(\mathcal{X}, G, \neg\alpha, \beta) &= \left\{\iota : X \rightarrowtail G \;\middle|\; \begin{array}{l}(\nexists \zeta : X \rightarrowtail A.\, \iota = \zeta \star \alpha)\\ \wedge\, (\exists \eta : X \rightarrowtail B.\, \iota = \eta \star \beta)\end{array}\right\},\\
\mathrm{Mono}(\mathcal{X}, G, \neg\alpha, \neg\beta) &= \left\{\iota : X \rightarrowtail G \;\middle|\; \begin{array}{l}(\nexists \zeta : X \rightarrowtail A.\, \iota = \zeta \star \alpha)\\ \wedge\, (\nexists \eta : X \rightarrowtail B.\, \iota = \eta \star \beta)\end{array}\right\}.
\end{aligned}$$

For a set $\mathrm{Mono}(\mathcal{X}, \dots)$, $\mathrm{Mono}(\mathcal{X}, \dots)_{\mathrm{NF}}$ *denotes the subset of* X*-occurrences whose images are not included in any occurrence of the forbidden context if it exists, and* $\mathrm{Mono}(\mathcal{X}, \dots)_{\mathrm{F}}$ *denotes the subset of* X*-occurrences whose images are included in some occurrences of the forbidden context if it exists.*

Let $\mathcal{X} = (X, \mathcal{C})$ be a ruler-graph and $\rho = (L \overset{l}{\leftarrowtail} K \overset{r}{\rightarrowtail} R)$ be a rule. Consider the following DPO diagram:

$$\begin{array}{ccccc}
L & \xleftarrow{\quad l \quad} & K & \xrightarrow{\quad r \quad} & R\\
\downarrow{\scriptstyle m} & & \downarrow{\scriptstyle u} & & \downarrow{\scriptstyle m'}\\
G & \xleftarrow{\quad l' \quad} & C & \xrightarrow{\quad r' \quad} & H
\end{array}$$

The set $\mathrm{Mono}(\mathcal{X}, G)_{\mathrm{NF}}$ decomposes into a union of three disjoint subsets:

- $\mathrm{Mono}(\mathcal{X}, G, m)_{\mathrm{NF}}$,
- $\mathrm{Mono}(\mathcal{X}, G, \neg m, l')_{\mathrm{NF}}$,
- $\mathrm{Mono}(\mathcal{X}, G, \neg m, \neg l')_{\mathrm{NF}}$.

Similarly, the set $\mathrm{Mono}(\mathcal{X}, H)_{\mathrm{NF}}$ decomposes into a union of three disjoint subsets:

- $\mathrm{Mono}(\mathcal{X}, H, m')_{\mathrm{NF}}$,
- $\mathrm{Mono}(\mathcal{X}, H, \neg m', r')_{\mathrm{NF}}$,
- $\mathrm{Mono}(\mathcal{X}, H, \neg m', \neg r')_{\mathrm{NF}}$.

Thus, the following equality holds:

$$\begin{aligned}
&|\mathrm{Mono}(\mathcal{X}, G)_{\mathrm{NF}}| - |\mathrm{Mono}(\mathcal{X}, H)_{\mathrm{NF}}|\\
=&(|\mathrm{Mono}(\mathcal{X}, G, m)_{\mathrm{NF}}| - |\mathrm{Mono}(\mathcal{X}, H, m')_{\mathrm{NF}}|) +\\
&(|\mathrm{Mono}(\mathcal{X}, G, \neg m, l')_{\mathrm{NF}}| - |\mathrm{Mono}(\mathcal{X}, H, \neg m', r')_{\mathrm{NF}}|) +\\
&(|\mathrm{Mono}(\mathcal{X}, G, \neg m, \neg l')_{\mathrm{NF}}| - |\mathrm{Mono}(\mathcal{X}, H, \neg m', \neg r')_{\mathrm{NF}}|). \qquad (5.1)
\end{aligned}$$

In the remainder of this section, we suppose that ρ^{-1} is F-non-increasing if $\mathcal{C} = \{f : X \rightarrowtail F\}$ and that ρ and ρ^{-1} are X-non-increasing. Lemma 5.4 asserts that

$\mathrm{Mono}(\mathcal{X}, G, \neg m, l')_{\mathrm{NF}}$ contains more elements than $\mathrm{Mono}(\mathcal{X}, H, \neg m', r')_{\mathrm{NF}}$, and Lemma 5.5 asserts that $\mathrm{Mono}(\mathcal{X}, G, \neg m, \neg l')_{\mathrm{NF}}$ contains more elements than $\mathrm{Mono}(\mathcal{X}, H, \neg m', \neg r')_{\mathrm{NF}}$. Hence, $|\mathrm{Mono}(\mathcal{X}, G)_{\mathrm{NF}}| - |\mathrm{Mono}(\mathcal{X}, H)_{\mathrm{NF}}|$ can be bounded below using these lemmas.

Lemma 5.4. *Let $\rho = (L \overset{l}{\leftarrowtail} K \overset{r}{\rightarrowtail} R)$ be a rule and $\mathcal{X} = (X, \mathcal{C})$ be a ruler-graph. Suppose that ρ^{-1} is F-non-increasing if $\mathcal{C} = \{f : X \rightarrowtail F\}$. The following inequality holds:*

$$|\mathrm{Mono}(\mathcal{X}, G, \neg m, l')_{\mathrm{NF}}| \geq |\mathrm{Mono}(\mathcal{X}, H, \neg m', r')_{\mathrm{NF}}|.$$

Lemma 5.5. *Let $\rho = (L \overset{l}{\leftarrowtail} K \overset{r}{\rightarrowtail} R)$ be a rule, and let $\mathcal{X} = (X, \mathcal{C})$ be a ruler-graph. Suppose that ρ and ρ^{-1} are X-non-increasing. We have*

$$|\mathrm{Mono}(\mathcal{X}, G, \neg m, \neg l')_{\mathrm{NF}}| \geq |\mathrm{Mono}(\mathcal{X}, H, \neg m', \neg r')_{\mathrm{NF}}|.$$

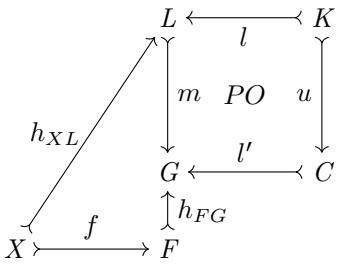
Under the assumptions of the above two lemmas, by (5.1), we obtain the following inequality:

$$|\mathrm{Mono}(\mathcal{X}, G)_{\mathrm{NF}}| - |\mathrm{Mono}(\mathcal{X}, H)_{\mathrm{NF}}| \geq |\mathrm{Mono}(\mathcal{X}, G, m)_{\mathrm{NF}}| - |\mathrm{Mono}(\mathcal{X}, H, m')_{\mathrm{NF}}|.$$

The following definition introduces a subset of $\mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}}$ that consists of occurrences of X whose images are not included in any occurrence of the forbidden context in L, but, in some rewriting step $G \Rightarrow_{\rho}^{m} H$, their images may possibly be included in some occurrence of the forbidden context in the host graph G.

Definition 5.6. *Let $\mathcal{X} = (X, \mathcal{C})$ be a ruler-graph, and let $\rho = (L \overset{l}{\leftarrowtail} K \overset{r}{\rightarrowtail} R)$ be a rule. Define $\Gamma(\mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}})$ as the subset of $\mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}}$ given by the following conditions:*

- *$\Gamma(\mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}}) = \varnothing$ if $\mathcal{C} = \varnothing$, and*
- *if $\mathcal{C} = \{f : X \rightarrowtail F\}$, then a monomorphism $h_{XL} : X \rightarrowtail L$ is an element of the set $\Gamma(\mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}})$ if and only if the following conditions hold:*
  - *there is no $h_{FL} : F \rightarrowtail L$ such that $f \star h_{FL} = h_{XL}$, and*
  - *there are a graph C, a monomorphism $K \overset{u}{\rightarrowtail} C$, and the pushout $L \overset{m}{\rightarrowtail} G \overset{l'}{\leftarrowtail} C$ of $L \overset{l}{\leftarrowtail} K \overset{u}{\rightarrowtail} C$ such that there is a monomorphism $h_{FG} : F \rightarrowtail G$ satisfying $h_{XL} \star m = h_{XF} \star h_{FG}$.*



To improve the readability of the subsequent discussion, we define

$$\Lambda(\mathcal{X}, \rho) \overset{\mathrm{def}}{=} (|\mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}}| - |\Gamma(\mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}})|) - |\mathrm{Mono}(\mathcal{X}, R)_{\mathrm{NF}}|.$$

Note that in the category **Graph**, the pushout of two arrows always exists [30, p.188]. This justifies the existence of the pushout square in the second condition of the above definition. Furthermore, since X and the forbidden context (if exists) are finite graphs, $\Lambda(\mathcal{X},\rho)$ can be precisely computed. Consequently, it provides a lower bound for the change in the number of X-occurrences in a rewriting step using ρ, as shown in the following lemma whose proof is given in § 5.5.

Lemma 5.7. *Let $\mathcal{X}$ be a ruler-graph, and let $\rho=(L \overset{l}{\hookleftarrow} K \overset{r}{\rightarrowtail} R)$ be a rule. We have*

$$|\mathrm{Mono}(\mathcal{X},G,m)_{\mathrm{NF}}|-|\mathrm{Mono}(\mathcal{X},H,m')_{\mathrm{NF}}|\geq\Lambda(\mathcal{X},\rho).$$

Thus, by the unnumbered equation preceding Definition 5.6, the following inequality holds:

$$|\mathrm{Mono}(\mathcal{X},G)_{\mathrm{NF}}|-|\mathrm{Mono}(\mathcal{X},H)_{\mathrm{NF}}|\geq\Lambda(\mathcal{X},\rho). \tag{5.2}$$

Consequently, we have

$$\begin{aligned}
w_{s_{\mathbb{X}}}(G)-w_{s_{\mathbb{X}}}(H) &\overset{\text{def}}{=} \sum_{\mathcal{X}\in\mathbb{X}} s_{\mathbb{X}}(\mathcal{X})*m_X(G)-\sum_{\mathcal{X}\in\mathbb{X}} s_{\mathbb{X}}(\mathcal{X})*m_X(H)\\
&\overset{\text{def}}{=} \sum_{\mathcal{X}\in\mathbb{X}} s_{\mathbb{X}}(\mathcal{X})*|\mathrm{Mono}(\mathcal{X},G)_{\mathrm{NF}}|-\sum_{\mathcal{X}\in\mathbb{X}} s_{\mathbb{X}}(\mathcal{X})*|\mathrm{Mono}(\mathcal{X},H)_{\mathrm{NF}}|\\
&= \sum_{\mathcal{X}\in\mathbb{X}} s_{\mathbb{X}}(\mathcal{X})*(|\mathrm{Mono}(\mathcal{X},G)_{\mathrm{NF}}|-|\mathrm{Mono}(\mathcal{X},H)_{\mathrm{NF}}|)\\
&\geq \sum_{\mathcal{X}\in\mathbb{X}} s_{\mathbb{X}}(\mathcal{X})*\Lambda(\mathcal{X},\rho). \qquad \text{by (5.2)}
\end{aligned}$$

This proves the following key lemma of this section.

Lemma 5.8 (Decreasing step)**.** *Let $\rho=(L \overset{l}{\hookleftarrow} K \overset{r}{\rightarrowtail} R)$ be an injective DPO rewriting rule, $G\Rightarrow_{\rho,\mathfrak{M}} H$ a rewriting step, $\mathbb{X}$ a set of ruler-graphs, and $s_{\mathbb{X}}:\mathbb{X}\to\mathbb{N}$ a weight function. Suppose that for every $\mathcal{X}=(X,\mathcal{C})\in\mathbb{X}$, ρ is X-non-increasing, and if $\mathcal{C}=\{f:X\rightarrowtail F\}$ then ρ^{-1} is X-non-increasing and F-non-increasing. The following inequality holds:*

$$w_{s_{\mathbb{X}}}(G)-w_{s_{\mathbb{X}}}(H)\geq\sum_{\mathcal{X}\in\mathbb{X}} s_{\mathbb{X}}(\mathcal{X})*\Lambda(\mathcal{X},\rho).$$

Finally, our main result follows.

Theorem 5.9 (Termination)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be sets of injective DPO rewriting rules, $\mathbb{X}$ a set of ruler-graphs, and $s_{\mathbb{X}}$ a weight function. If the following conditions hold:*

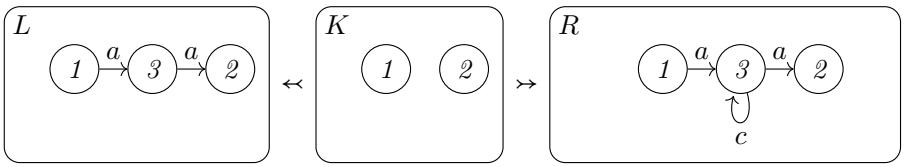
1. *for every $\rho\in\mathcal{A}\cup\mathcal{B}$ and for every $\mathcal{X}=(X,\mathcal{C})\in\mathbb{X}$, ρ is X-non-increasing, and if $\mathcal{C}=\{f:X\rightarrowtail F\}$ then ρ^{-1} is X-non-increasing and F-non-increasing, and*
2. *for every $\rho\in\mathcal{A}$, we have $\sum_{\mathcal{X}\in\mathbb{X}} s_{\mathbb{X}}(\mathcal{X})*\Lambda(\mathcal{X},\rho)>0$, and*
3. *for every $\rho\in\mathcal{B}$, we have $\sum_{\mathcal{X}\in\mathbb{X}} s_{\mathbb{X}}(\mathcal{X})*\Lambda(\mathcal{X},\rho)\geq 0$,*

then $\Rightarrow_{\mathcal{A},\mathcal{M}}$ terminates relative to $\Rightarrow_{\mathcal{B},\mathcal{M}}$.

Remark 5.10. *This work focuses on deriving sufficient conditions for termination, deferring the construction of ruler-graphs to future work.*

5.2 Examples

Example 5.11. *Consider the rule ρ [19, Example 1]:*



Consider the ruler-graph $\mathcal{X}=(L,\{f\})$ where the underlying graph is L and $f : L \rightarrowtail R$ is the inclusion function. Let $s_{\mathbb{X}}$ be the weight function with $s_{\mathbb{X}}(\mathcal{X}) = 1$.

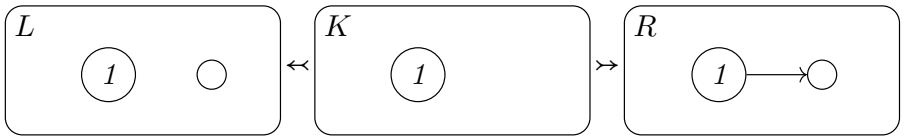
Since $D(R,X)$ is the empty set, all conditions in Definition 4.18 hold, and rule ρ is X-non-increasing. Similarly, ρ^{-1} is X-non-increasing and F-non-increasing, because $D(L,X)=\varnothing$ and $D(L,R)=\varnothing$.

We have

$$\begin{aligned}\Lambda(\mathcal{X},\rho) &\overset{\text{def}}{=} |\operatorname{Mono}(X,L)_{\mathrm{NF}}| - |\Gamma(\operatorname{Mono}(\mathcal{X},L)_{\mathrm{NF}})| - |\operatorname{Mono}(X,R)_{\mathrm{NF}}| \\ &= 1 - 0 - 0 \\ &= 1.\end{aligned}$$

Therefore, $s_{\mathbb{X}}(\mathcal{X}) * \Lambda(\mathcal{X},\rho) = 1 > 0$, and the rule terminates by Theorem 5.9.

Example 5.12. *Our method can prove termination of the rewriting rule ρ [38, Example D.3]:*



Let $\mathcal{X}=(L,\{f\})$ be the ruler-graph where $f : L \rightarrowtail R$ is the unique interface-preserving monomorphism from L to R. Let $s_{\mathbb{X}}$ be the weight function with $s_{\mathbb{X}}(\mathcal{X}) = 1$. Since $D(R,X)$ is the empty set, rule ρ is X-non-increasing. Rule ρ^{-1} is X-non-increasing and R-non-increasing, because $D(L,X)$ and $D(L,R)$ are empty sets. We have

$$\begin{aligned}\Lambda(\mathcal{X},\rho) &\overset{\text{def}}{=} |\operatorname{Mono}(X,L)_{\mathrm{NF}}| - |\Gamma(\operatorname{Mono}(\mathcal{X},L)_{\mathrm{NF}})| - |\operatorname{Mono}(X,R)_{\mathrm{NF}}| \\ &= 1 - 0 - 0 \\ &= 1.\end{aligned}$$

Therefore, $s_{\mathbb{X}}(\mathcal{X}) * \Lambda(\mathcal{X},\rho) = 1 > 0$, and the rule terminates by Theorem 5.9.

5.3 Empirical Results

A comparative analysis of termination techniques for DPO graph rewriting systems, drawn from prior work [72, 71, 19, 17, 38, 66] and Chapter 4, is summarized in Table 5.1. Our tool successfully proves termination for 16 systems,

including [38, Example D.3] and [19, Example 1]. These two cases could not be handled by the subgraph counting methods by Overbeek and Endrullis [66] and the morphism counting method introduced in Chapter 4, but are now provable using our extension.

Table 5.1: Applicability of termination techniques to DPO rewriting examples. The symbol ✓ indicates termination can be proved by the technique, ✗ indicates it cannot be proved, and – denotes irrelevance or out-of-scope cases.

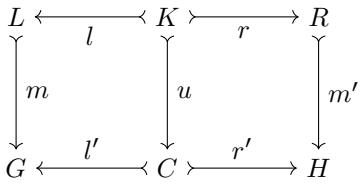
Examples \ **Techniques**		Forward closure [72]	Modular criterion [71]	Type graph [19]	Type graph [17]	Type graph [38]	Subgraph counting [66]	Morphism counting (Chapter 4)	Morphism counting (Chapter 5)
Chapter 4	Example 4.28	✓	✗	✗	✗	✗	✗	✓	✓
[72]	Example 3.8	✓	–	–	–	–	–	✓	✓
[71]	Example 3	–	✓	–	–	–	–	✓	✓
	Example 5	–	✓	–	–	–	–	✓	✓
[19]	Example 1	–	–	✓	–	–	–	✗	✓
	Example 4 and 6	–	–	✓	–	–	–	✓	✓
[17]	Example 2	–	–	–	✓	–	–	✓	✓
	Example 4	–	–	–	✓	–	–	✓	✓
[38]	Example 6.2	–	–	–	–	✓	–	✓	✓
	Example 6.3	–	–	–	–	✓	✗	✓	✓
	Example D.1	–	–	–	–	✓	–	✓	✓
	Example D.3	–	–	–	–	✓	✗	✗	✓
[66]	Example 5.3	–	–	–	–	–	✓	✓	✓
	Example 5.5	–	–	–	–	–	✓	✓	✓
	Example 5.6	–	–	–	–	–	✓	✓	✓
	Example 5.8	–	–	–	–	–	✓	✓	✓
[71]	Example 6	–	✓	–	–	–	–	–	–
[38]	Example 6.4	–	–	–	–	✓	–	–	–
	Example 6.5	–	–	–	–	✓	–	–	–
	Example D.4	–	–	–	–	✓	–	–	–
[66]	Example 5.2	–	–	–	–	–	✓	–	–
	Example 5.7	–	–	–	–	–	✓	–	–
	Example 5.9	–	–	–	–	–	✓	–	–
[72]	Example 4.1	✓	–	–	–	–	✓	✗	✗
[71]	Example 4	–	✓	–	–	–	–	✗	✗
[19]	Routing protocol	–	–	✓	–	–	–	✗	✗
	Example 5	–	–	✓	–	–	–	✗	✗
[17]	Example 5	–	–	–	✓	–	–	✗	✗
	Example 6	–	–	–	✓	–	–	✗	✗
[38]	Example D.2	–	–	–	–	✓	–	✗	✗

5.4 Conclusion

We have extended our subgraph-counting method to handle cases such as Examples 5.11 and 5.12 where both the method of Overbeek and Endrullis [66] and the morphism counting method of Chapter 4 fail. These failures arise when there exists an injection from a rule's left-hand side graph into its right-hand side graph. The extended method has been implemented in the tool described in Chapter 6.

5.5 Appendix

Lemma 5.13. *Let $\mathcal{X}=(X,\{f : X \rightarrowtail F\})$ be a ruler-graph and $\rho=(L \overset{l}{\hookleftarrow} K \overset{r}{\rightarrowtail} R)$ be a rule. Consider the DPO diagram:*

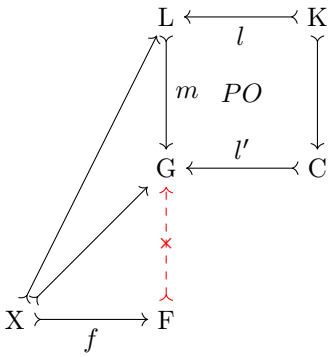


We have the following equalities:

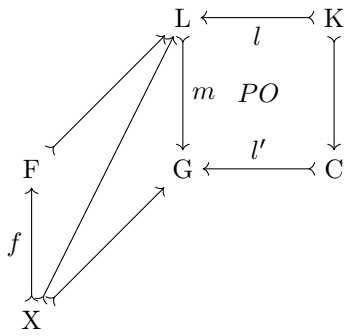
$$
\begin{aligned}
|\,\mathrm{Mono}(\mathcal{X},G,m)_{\mathrm{NF}}| = {} & |\,\mathrm{Mono}(\mathcal{X},L)_{\mathrm{NF}}| - \\
& |\{h_{XL} \in \mathrm{Mono}(\mathcal{X},L)_{\mathrm{NF}} \mid \exists h_{FG}.h_{XL} \star m = f \star h_{FG}\}|, \\
|\,\mathrm{Mono}(\mathcal{X},H,m')_{\mathrm{NF}}| = {} & |\,\mathrm{Mono}(\mathcal{X},R)_{\mathrm{NF}}| - \\
& |\{h_{XR} \in \mathrm{Mono}(\mathcal{X},R)_{\mathrm{NF}} \mid \exists h_{FH}.h_{XR} \star m' = f \star h_{FH}\}|.
\end{aligned}
$$

Proof. By symmetry, it suffices to prove the first equality, and the second equality follows. To prove the first equality, we show that there is a bijection between $\mathrm{Mono}(\mathcal{X},L)_{\mathrm{NF}} \setminus \{h_{XL} \in \mathrm{Mono}(\mathcal{X},L)_{\mathrm{NF}} \mid \exists h_{FG}.h_{XL} \star m = f \star h_{FG}\}$ and $\mathrm{Mono}(\mathcal{X},G,m)_{\mathrm{NF}}$.

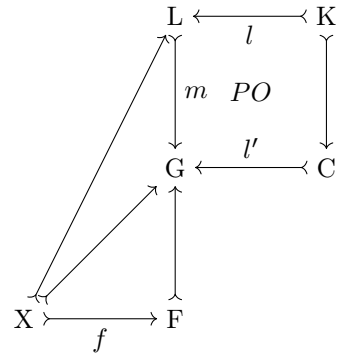
Let $h_{XG} \in \mathrm{Mono}(\mathcal{X},G,m)_{\mathrm{NF}}$. By definition, there is a monomorphism h_{XL} from X to L such that $h_{XG} = h_{XL} \star m$ and there is no monomorphism h_{FG} from F to G such that $f \star h_{FG} = h_{XG}$. We must show $h_{XL} \in \mathrm{Mono}(\mathcal{X},L)_{\mathrm{NF}}$ and there is no $h_{FG} : f \rightarrowtail G$ such that $h_{XL} \star m = f \star h_{FG}$. These can be visualized by the following commutative diagram where edges that cannot exist are in red and dashed:



We show $h_{XL} \in \mathrm{Mono}(\mathcal{X},L)_{\mathrm{NF}}$ by contradiction. Suppose that the contrary holds. There is a morphism $h_{FL} : F \rightarrowtail L$ such that $f \star h_{FL} = h_{XL}$. The image of h_{XG} is, therefore, included in the image of $h_{FL} \star m$, as shown in the commutative diagram below, which contradicts the assumption that h_{XG} is in $\mathrm{Mono}(\mathcal{X},G,m)_{\mathrm{NF}}$.



We show that there is no $h_{FG} : f \nrightarrow G$ such that $h_{XL} \star m = f \star h_{FG}$ by contradiction. Suppose that the contrary holds. Under this assumption, we have the following commutative diagram. The image of h_{XG} is, thus, included in the image of $f \star h_{FG}$ for some morphism f which contradicts the assumption that h_{XG} is in $\mathrm{Mono}(\mathcal{X}, G, m)_{\mathrm{NF}}$.



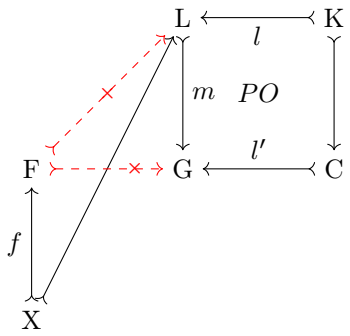
Let $h_{XG}, h'_{XG} \in \mathrm{Mono}(\mathcal{X}, G, m)_{\mathrm{NF}}$ such that $h_{XG} \neq h'_{XG}$. By definition, there are morphisms $h_{XL}, h'_{XL} : X \nrightarrow L$ such that $h_{XG} = h_{XL} \star m$, $h'_{XG} = h'_{XL} \star m$, and no morphism $h_{FG}, h'_{FG} : F \nrightarrow G$ such that $f \star h_{FG} = h_{XG}$ and $f \star h'_{FG} = h'_{XG}$. Since m is injection, we have $h_{XL} \neq h'_{XL}$.

There is, therefore, an injection from the set $\mathrm{Mono}(\mathcal{X}, G, m)_{\mathrm{NF}}$ to the set $\mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}} \setminus \{h_{XL} \in \mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}} \mid \exists h_{FG}. h_{XL} \star m = f \star h_{FG}\}$.

Conversely, let h_{XL} be a morphism of the following set

$$\mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}} \setminus \{h_{XL} \in \mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}} \mid \exists h_{FG}. h_{XL} \star m = f \star h_{FG}\}.$$

We have the following commutative diagram:



To prove $h_{XL} \star m \in \mathrm{Mono}(\mathcal{X}, G, m)_{\mathrm{NF}}$, we show that $h_{XL} \star m$ is an element of $\mathrm{Mono}(\mathcal{X}, G, m)$ and that there is no morphism $h_{FG} : F \nrightarrow G$ such that $f \star h_{FG} = h_{XL} \star m$. We have $h_{XL} \star m \in \mathrm{Mono}(\mathcal{X}, G, m)$, because the equality $h_{XL} \star m = f \star h_{FG}$ holds by definition. There is no morphism $h_{FG} : F \nrightarrow G$ such that $f \star h_{FG} = h_{XL} \star m$, by definition of h_{XL}.

Let h_{XL}, h'_{XL} be morphisms in the following set

$$\mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}} \setminus \{h_{XL} \in \mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}} \mid \exists h_{FG}. h_{XL} \star m = f \star h_{FG}\}$$

such that $h_{XL} \neq h'_{XL}$. Since m is injective, we have $h_{XL} \star m \neq h'_{XL} \star m$.

There is, thus, an injection from the following set

$$(\mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}} \setminus \{h_{XL} \in \mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}} \mid \exists h_{FG}. h_{XL} \star m = f \star h_{FG}\}) \star m$$

to $\mathrm{Mono}(\mathcal{X}, G, m)_{\mathrm{NF}}$. Since the following two sets are in bijection

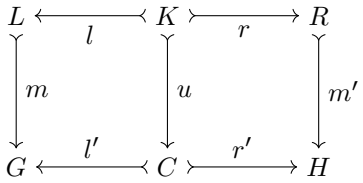
- $(\mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}} \setminus \{h_{XL} \in \mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}} \mid \exists h_{FG}. h_{XL} \star m = f \star h_{FG}\}) \star m$,
- $\mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}} \setminus \{h_{XL} \in \mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}} \mid \exists h_{FG}. h_{XL} \star m = f \star h_{FG}\}$,

we conclude that there is an injection from the set A to $\mathrm{Mono}(\mathcal{X}, G, m)_{\mathrm{NF}}$, where $A \stackrel{\mathrm{def}}{=} \mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}} \setminus \{h_{XL} \in \mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}} \mid \exists h_{FG}. h_{XL} \star m = f \star h_{FG}\}$.

Finally, we conclude. □

Lemma 5.7

Let $\mathcal{X} = (X, \{f : X \nrightarrow F\})$ be a ruler-graph, and let $\rho = (L \stackrel{l}{\hookleftarrow} K \stackrel{r}{\nrightarrow} R)$ be a rule. Consider the DPO diagram shown below.



We have

$$|\mathrm{Mono}(\mathcal{X}, G, m)_{\mathrm{NF}}| - |\mathrm{Mono}(\mathcal{X}, H, m')_{\mathrm{NF}}| \geq \Lambda(\mathcal{X}, \rho)$$

where $\Lambda(\mathcal{X}, \rho) \stackrel{\mathrm{def}}{=} |\mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}}| - |\Gamma(\mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}})| - |\mathrm{Mono}(\mathcal{X}, R)_{\mathrm{NF}}|$.

Proof. There are two cases to consider: either $\mathcal{C} = \varnothing$ or $\mathcal{C} = \{f : X \nrightarrow F\}$. For the first case, since $\Gamma(\mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}}) = \varnothing$ by definition, the inequality is equivalent to the following:

$$|\mathrm{Mono}(\mathcal{X}, G, m)_{\mathrm{NF}}| - |\mathrm{Mono}(\mathcal{X}, H, m')_{\mathrm{NF}}| \geq |\mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}}| - |\mathrm{Mono}(\mathcal{X}, R)_{\mathrm{NF}}|.$$

By definition, it is equivalent to the inequality:

$$|\mathrm{Mono}(X, G, m)| - |\mathrm{Mono}(X, H, m')| \geq |\mathrm{Mono}(X, L)| - |\mathrm{Mono}(X, R)|$$

because the following equalities hold by Lemma 4.13:

- $|\mathrm{Mono}(X, G, m)| = |\mathrm{Mono}(X, L)|$,
- $|\mathrm{Mono}(X, H, m')| = |\mathrm{Mono}(X, R)|$.

For the second case, by Lemma 5.13, we have the following:

$$\begin{aligned}
&|\mathrm{Mono}(\mathcal{X}, G, m)_{\mathrm{NF}}| - |\mathrm{Mono}(\mathcal{X}, H, m')_{\mathrm{NF}}| \\
=&(|\mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}}| - |\{h_{XL} \in \mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}} \mid \exists h_{FG}. h_{XL} \star m = f \star h_{FG}\}|) - \\
&(|\mathrm{Mono}(\mathcal{X}, R)_{\mathrm{NF}}| - |\{h_{XR} \in \mathrm{Mono}(\mathcal{X}, R)_{\mathrm{NF}} \mid \exists h_{FH}. h_{XR} \star m' = f \star h_{FH}\}|)
\end{aligned}$$

where $\{h_{XL} \in \mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}} \mid \exists h_{FG}. h_{XL} \star m = h_{XF} \star h_{FG}\}$ is the set of occurrences of X in L that are not included in any occurrences of its forbidden context in L but included in an occurrence of its forbidden context in G, and, analogously, $\{h_{XR} \in \mathrm{Mono}(\mathcal{X}, R)_{\mathrm{NF}} \mid \exists h_{FH}. h_{XR} \star m' = h_{XF} \star h_{FH}\}$ is the set of X-occurrences in R that are not included in any occurrences of its forbidden context in R but included in an occurrence of its forbidden context in H.

The following equalities and inequalities hold:

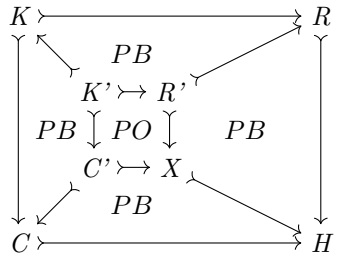
$$\begin{aligned}
&|\mathrm{Mono}(\mathcal{X}, G, m)_{\mathrm{NF}}| - |\mathrm{Mono}(\mathcal{X}, H, m')_{\mathrm{NF}}| \\
=&(|\mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}}| - |\mathrm{Mono}(\mathcal{X}, R)_{\mathrm{NF}}|) + \\
&(|\{h_{XR} \in \mathrm{Mono}(\mathcal{X}, R)_{\mathrm{NF}} \mid \exists h_{FH}. h_{XR} \star m' = f \star h_{FH}\}| - \\
&|\{h_{XL} \in \mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}} \mid \exists h_{FG}. h_{XL} \star m = f \star h_{FG}\}|) \\
\geq&(|\mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}}| - |\mathrm{Mono}(\mathcal{X}, R)_{\mathrm{NF}}|) - \\
&|\{h_{XL} \in \mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}} \mid \exists h_{FG}. h_{XL} \star m = h_{XF} \star h_{FG}\}| \\
=&(|\mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}}| - |\mathrm{Mono}(\mathcal{X}, R)_{\mathrm{NF}}|) - \\
&|\Gamma(\mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}})| \qquad \text{by Definition 5.6} \\
=&|\mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}}| - |\Gamma(\mathrm{Mono}(\mathcal{X}, L)_{\mathrm{NF}})| - |\mathrm{Mono}(\mathcal{X}, R)_{\mathrm{NF}}| \\
=&\Lambda(\mathcal{X}, \rho).
\end{aligned}$$

□

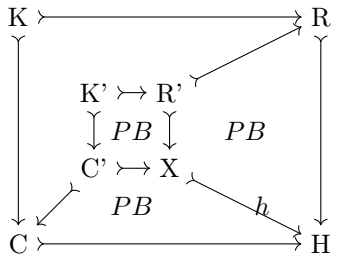
Lemma 5.14. *Consider the following pushout diagram in the category* ***Graph****:*

$$\begin{array}{ccc}
K & \xrightarrow{\quad r \quad} & R \\
\downarrow & PO \quad m' & \downarrow \\
C & \xrightarrow{\quad r' \quad} & H
\end{array}$$

Let X be a graph, and $h_{XH} : X \rightarrowtail H$ a monomorphism. Then, we can construct the following commutative diagram in the category ***Graph****:*



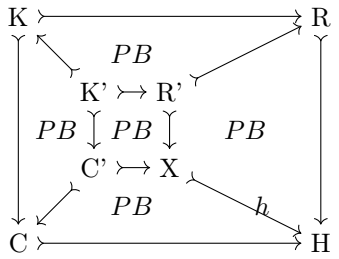
Proof. In the category **Graph**, we can construct the following commutative diagram :



Since $KCHR$ is a pushout, it is also a pullback, by Proposition 2.27. Therefore, from $h_{K'C'} \star h_{C'C} \star h_{CH} = h_{K'R'} \star h_{R'R} \star h_{R'R}$ and the universal property of pullback, there is a unique morphism $h_{K'K} : K' \rightarrowtail K$ such that

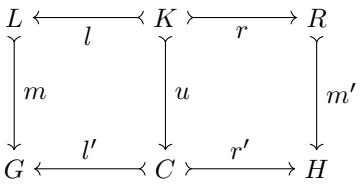
$$h_{K'C'} \star h_{C'C} = h_{K'K} \star h_{KC},$$
$$h_{K'R'} \star h_{R'R} = h_{K'K} \star h_{KR}.$$

The square $K'KCC'$ and $K'R'RK$ are pullbacks, by Lemma 4.45. Furthermore, $K'C'XR'$ is a pushout by Lemma 4.46. Hence, the following commutative diagram holds:



□

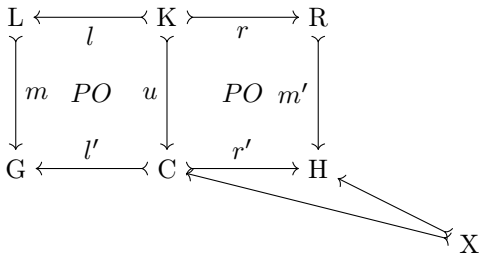
Lemma 5.4 Let $\rho = (L \overset{l}{\leftarrowtail} K \overset{r}{\rightarrowtail} R)$ be a rule and $\mathcal{X} = (X, \mathcal{C})$ be a ruler-graph with underlying graph X. Suppose that ρ^{-1} is F-non-increasing if $\mathcal{C} = \{f : X \rightarrowtail F\}$. For the DPO diagram:



the following inequality holds:

$$|\,\mathrm{Mono}(\mathcal{X}, G, \neg\, m, l')_{\mathrm{NF}}| \geq |\,\mathrm{Mono}(\mathcal{X}, H, \neg\, m', r')_{\mathrm{NF}}|.$$

Proof. There are two cases to consider: either $\mathcal{C} = \varnothing$ or $\mathcal{C} = \{f : X \nrightarrow F\}$. In the first case, since there is no forbidden context, the inequality is equivalent to $|\,\mathrm{Mono}(\mathcal{X}, G, \neg m, l')| \geq |\,\mathrm{Mono}(\mathcal{X}, H, \neg m', r')|$. To show this inequality, we are going to establish an injection from $\mathrm{Mono}(\mathcal{X}, H, \neg m', r')$ to $\mathrm{Mono}(\mathcal{X}, G, \neg m, l')$. Let $h_{XH} \in \mathrm{Mono}(\mathcal{X}, H, \neg m', r')$. By definition, there is a monomorphism $h_{XC} : X \nrightarrow C$ such that $h_{XH} = h_{XC} \star r'$. Thus we have the following commutative diagram:



We must show $h_{XC} \star l' \in \mathrm{Mono}(\mathcal{X}, G, \neg m, l')$ which is equivalent to showing that there is no h_{XL} such that $h_{XL} \star m = h_{XC} \star l'$. Suppose that the contrary holds. Since $KLGC$ is a pullback by Proposition 2.27, there is a morphism $h_{XK} : X \nrightarrow K$ such that

$$h_{XK} \star u = h_{XC}. \tag{5.3}$$

Therefore, we have

$$\begin{aligned} h_{XH} &= h_{XC} \star r' \\ &= (h_{XK} \star u) \star r' && \text{by (5.3)} \\ &= h_{XK} \star (r \star m') \\ &= (h_{XK} \star r) \star m'. \end{aligned}$$

This contradicts the fact that $h_{XH} \in \mathrm{Mono}(\mathcal{X}, H, \neg m', r')$. Therefore, we have $h_{XC} \star l' \in \mathrm{Mono}(\mathcal{X}, G, \neg m, l')$.

Furthermore, if $h_{XH}, h'_{XH} \in \mathrm{Mono}(\mathcal{X}, H, \neg m', r')$ with $h_{XH} \neq h'_{XH}$, then there are h_{XC}, h'_{XC} such that $h_{XH} = h_{XC} \star r'$ and $h'_{XH} = h'_{XC} \star r'$. Since r' is injective, we deduce that $h_{XC} \neq h'_{XC}$. Therefore, from the injectivity of l', we have $h_{XC} \star l' \neq h'_{XC} \star l'$. Thus, there is an injection from $\mathrm{Mono}(\mathcal{X}, H, \neg m', r')$ to $\mathrm{Mono}(\mathcal{X}, G, \neg m, l')$, which proves the inequality.

Now, let us focus on the second case, where $\mathcal{X} = (X, f : X \nrightarrow F)$. We have the following equalities:

$$
\begin{aligned}
&|\mathrm{Mono}(\mathcal{X}, G, \neg m, l')_{\mathrm{NF}}| - |\mathrm{Mono}(, H, \neg m', r')_{\mathrm{NF}}| \\
=&(|\mathrm{Mono}(X, G, \neg m, l')| - |\mathrm{Mono}(\mathcal{X}, G, \neg m, l')_{\mathrm{F}}|) - \\
&(|\mathrm{Mono}(X, H, \neg m', r')| - |\mathrm{Mono}(\mathcal{X}, H, \neg m', r')_{\mathrm{F}}|) \\
=&(|\mathrm{Mono}(X, G, \neg m, l')| - |\mathrm{Mono}(X, H, \neg m', r')|) + \\
&(|\mathrm{Mono}(\mathcal{X}, H, \neg m', r')_{\mathrm{F}}| - |\mathrm{Mono}(\mathcal{X}, G, \neg m, l')_{\mathrm{F}}|) \\
=&(|\mathrm{Mono}(X, C, \neg u)| - |\mathrm{Mono}(X, C, \neg u)|) + \\
&(|\mathrm{Mono}(\mathcal{X}, H, \neg m', r')_{\mathrm{F}}| - |\mathrm{Mono}(\mathcal{X}, G, \neg m, l')_{\mathrm{F}}|) \qquad \text{by Lemma 4.13} \\
=&|\mathrm{Mono}(\mathcal{X}, H, \neg m', r')_{\mathrm{F}}| - |\mathrm{Mono}(\mathcal{X}, G, \neg m, l')_{\mathrm{F}}|
\end{aligned}
$$

We prove this lemma by show that the function ϕ from $\mathrm{Mono}(\mathcal{X}, G, \neg m, l')_{\mathrm{F}}$ to $\mathrm{Mono}(\mathcal{X}, H, \neg m', r')_{\mathrm{NF}}$ defined by $\phi(h_{XG}) = h_{XC} \star r'$ for each element h_{XG} in $\mathrm{Mono}(\mathcal{X}, G, \neg m, l')_{\mathrm{F}}$, where $h_{XC} : X \nrightarrow C$ is the unique monomorphism such that $h_{XG} = h_{XC} \star l'$, is injective.

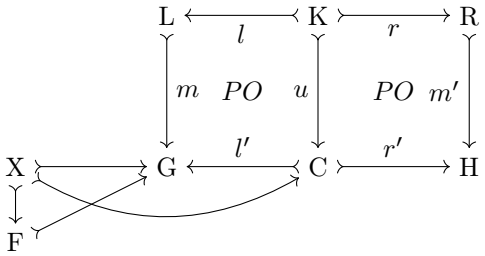
We first show that ϕ is well-defined.

Let $h_{XG} \in \mathrm{Mono}(\mathcal{X}, G, \neg m, l')_{\mathrm{F}}$. By definition, there are $h_{FG} : F \nrightarrow G$ and $h_{XC} : X \nrightarrow C$ such that

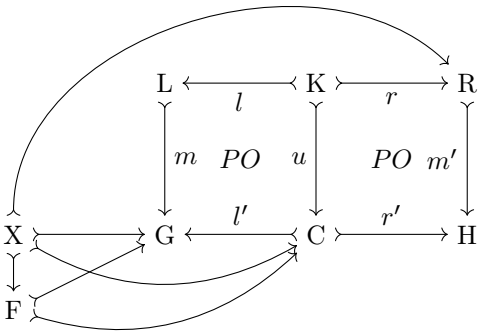
$$h_{XG} = h_{XC} \star l', \tag{5.4}$$

$$h_{XF} \star h_{FG} = H_{XG}. \tag{5.5}$$

Therefore, the following commutative diagram holds:



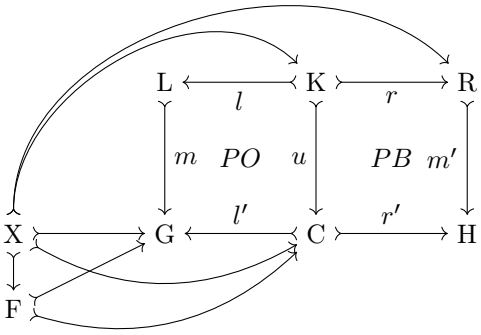
We prove that $\phi(h_{XG})$ cannot be factorized by m', by contradiction. Suppose that the contrary holds, i.e. there is $h_{XR} : X \to R$ such that $\phi(h_{XG}) \overset{def}{=} h_{XC} \star r' = h_{XR} \star m'$. The morphism h_{XR} is injective, because $\phi(h_{XG})$ is injective. Therefore, under this assumption, we have the following commutative diagram:



The square $KCHR$ is also a pullback by Proposition 2.27. By the universal property of pullback, there is a unique morphism $h_{XK} : X \to K$, which is injective because h_{XR} is injective, such that $h_{XK} \star r = h_{XR}$ and

$$h_{XK} \star u = h_{XC}. \tag{5.6}$$

Therefore, the following commutative diagram holds:

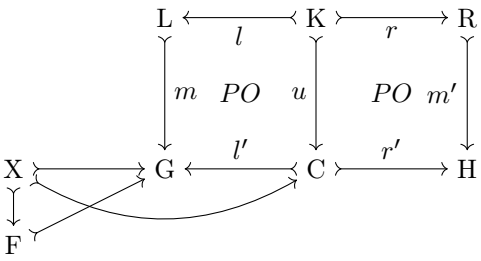


Thus, we have

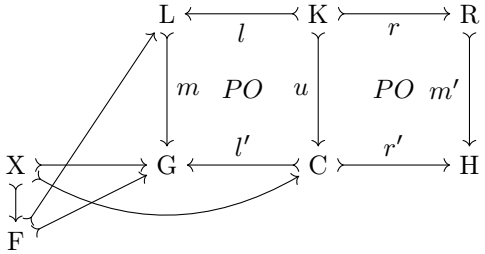
$$\begin{aligned} h_{XG} &= h_{XC} \star l' \\ &= (h_{XK} \star u) \star l' && \text{by (5.6)} \\ &= h_{XK} \star (l \star m). && \text{by commutativity of } KLGC \end{aligned}$$

This is a contradiction because h_{XG} is an element of $\mathrm{Mono}(\mathcal{X}, G, \neg m, l')_{\mathrm{F}}$ and, by definition, cannot be factorized by m. Thus, the assumption is false, and $\phi(h_{XG})$ cannot be factorized by m'.

Consider the commutative diagram below. It remains to show that $\phi(h_{XG})$ can be decomposed as $x \star y$ with $x : X \nrightarrow F$ and $y : F \nrightarrow H$. There are three mutually exclusive cases according to whether or not the morphism h_{FG} can be factorized by m or l'.



- Case 1 : there is $h_{FL} : F \to L$ such that $h_{FG} = h_{FL} \star m$. This case is impossible, because if it were true, then we would have the following commutative diagram:

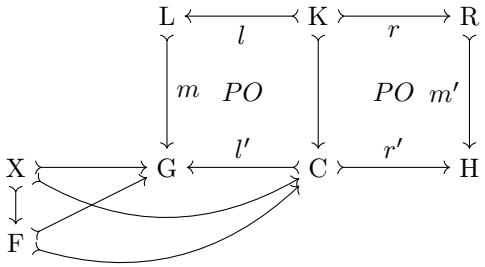


  Therefore, we would have

$$\begin{aligned} h_{XC} \star l' &= h_{XG} && \text{by (5.4)} \\ &= f \star h_{FG} && \text{by (5.5)} \\ &= f \star h_{FL} \star m. \end{aligned}$$

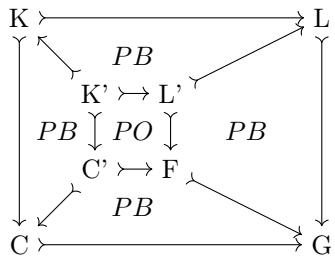
  This would imply that h_{XC} is not in $\mathrm{Mono}(X, G, \neg m, l')$.

- Case 2 : there is not any $h_{FL} : F \to L$ such that $h_{FG} = h_{FL} \star m$, but there is $h_{FC} : F \to C$ such that $h_{FG} = h_{FC} \star l'$. We have the following commutative diagram:



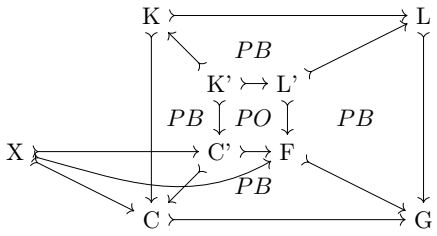
  We have therefore $\phi(h_{XG}) = h_{XC} \star r' = h_{XF} \star (h_{FC} \star r')$.

- Case 3 : there is no $h_{FL} : F \to L$ such that $h_{FG} = h_{FL} \star m$, and there is no $h_{FC} : F \to C$ such that $h_{FG} = h_{FC} \star l'$. By Lemma 5.14, we can construct the following commutative diagram:



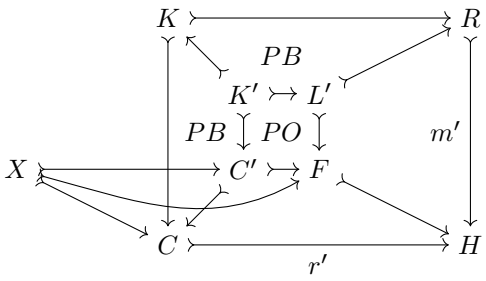
There is a unique monomorphism $h_{XC'} : X \rightarrowtail C'$ such that $h_{XC'} \star h_{C'C} = h_{XC}$ and $h_{XC'} \star h_{C'F} = h_{XF}$, because we have $h_{XF} \star h_{FG} = h_{XG}$, $h_{XC} \star h_{CG} = h_{XG}$ and $C'CGF$ is a pullback.

Thus, we have the following commutative diagram:



By the F-non-increasing property of ρ^{-1}, there is a morphism $h_{L'R} : L' \rightarrowtail R$ such that $K'L'RK$ is a pullback. By Lemma 4.48, there is a unique monomorphism $h_{FH} : F \rightarrowtail H$ such that $L'FHR$ and $C'FHC$ are commutative squares.

Thus, we have the following commutative diagram:



Finally, we have:

$$\begin{aligned}\phi(h_{XG}) &= h_{XC} \star r' \\ &= h_{XC'} \star h_{C'C} \star r' \\ &= (h_{XC'} \star h_{C'F}) \star h_{FH}.\end{aligned}$$

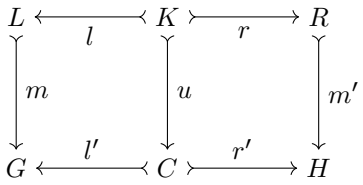
We have thus shown that ϕ is well defined.

Proving its injectivity is straightforward. Let $h_{XG}, h'_{XG} \in \mathrm{Mono}(\mathcal{X}, G, \neg m, l')_{\mathrm{F}}$ such that $h_{XG} \neq h'_{XG}$. There are morphisms h_{XC} and h'_{XC} such that

- $h_{XG} = h_{XC} \star l'$, and
- $h'_{XG} = h'_{XC} \star l'$.

By injectivity of l', we have $h_{XC} \neq h'_{XC}$. Therefore, since r' is injective, we have $\phi(h_{XG}) = h_{XC} \star r' \neq h'_{XC} \star r' = \phi(h'_{XG})$. □

Lemma 5.5 let $\rho = (L \overset{l}{\leftarrowtail} K \overset{r}{\rightarrowtail} R)$ be a rule, and let $\mathcal{X} = (X, \mathcal{C})$ be a ruler-graph. Suppose that ρ and ρ^{-1} are X-non-increasing. For the DPO diagram:



the following inequality holds:

$$|\mathrm{Mono}(\mathcal{X}, G, \neg m, \neg l')_{\mathrm{NF}}| \geq |\mathrm{Mono}(\mathcal{X}, H, \neg m', \neg r')_{\mathrm{NF}}|.$$

Proof. There are two cases to consider: $\mathcal{C} = \varnothing$ or $\mathcal{C} = \{f : X \rightarrowtail F\}$. For the first case, the inequality is equivalent to the inequality:

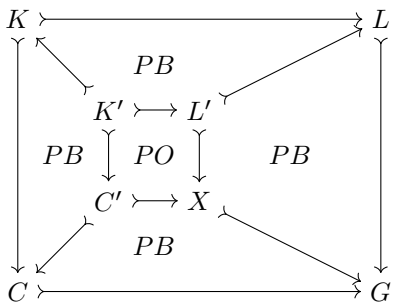
$$|\mathrm{Mono}(\mathcal{X}, G, \neg m, \neg l')| \geq |\mathrm{Mono}(\mathcal{X}, H, \neg m', \neg r')|,$$

and the result follows from Lemma 4.24. Let us now consider the second case. The following equalities and inequalities hold:

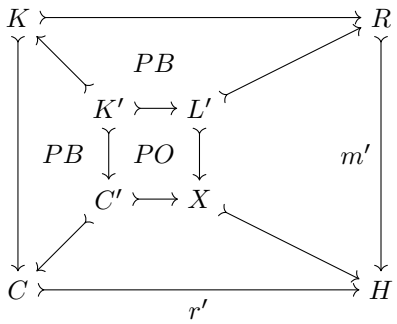
$$\begin{aligned}
&|\mathrm{Mono}(\mathcal{X}, G, \neg m, \neg l')_{\mathrm{NF}}| - |\mathrm{Mono}(\mathcal{X}, H, \neg m', \neg r')_{\mathrm{NF}}| \\
=&(|\mathrm{Mono}(X, G, \neg m, \neg l')| - |\mathrm{Mono}(\mathcal{X}, G, \neg m, \neg l')_{\mathrm{F}}|) - \\
&(|\mathrm{Mono}(X, H, \neg m', \neg r')| - |\mathrm{Mono}(\mathcal{X}, H, \neg m', \neg r')_{\mathrm{F}}|) \\
=&(|\mathrm{Mono}(X, G, \neg m, \neg l')| - |\mathrm{Mono}(X, H, \neg m', \neg r')|) + \\
&(|\mathrm{Mono}(\mathcal{X}, H, \neg m', \neg r')_{\mathrm{F}}| - |\mathrm{Mono}(\mathcal{X}, G, \neg m, \neg l')_{\mathrm{F}}|) \\
\geq&|\mathrm{Mono}(\mathcal{X}, H, \neg m', \neg r')_{\mathrm{F}}| - |\mathrm{Mono}(\mathcal{X}, G, \neg m, \neg l')_{\mathrm{F}}|. \qquad \text{by Lemma 4.24}
\end{aligned}$$

We prove $|\mathrm{Mono}(\mathcal{X}, H, \neg m', \neg r')_{\mathrm{F}}| \geq |\mathrm{Mono}(\mathcal{X}, G, \neg m, \neg l')_{\mathrm{F}}|$ by showing that we can construct an injection $\phi : \mathrm{Mono}(\mathcal{X}, G, \neg m, \neg l')_{\mathrm{F}} \to \mathrm{Mono}(\mathcal{X}, H, \neg m', \neg r')_{\mathrm{F}}$.

Let $h_{XG} \in \mathrm{Mono}(\mathcal{X}, G, \neg m, \neg l')_{\mathrm{F}}$. By Lemma 5.14, we can construct the following commutative diagram:



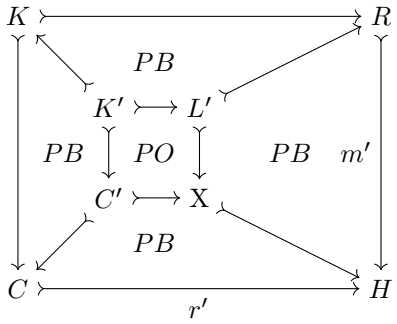
There is a monomorphism $h_{L'R} : L' \rightarrowtail R$ because ρ^{-1} is X-non-increasing by assumption. By Lemma 4.48, we can construct the following commutative diagram:



We define $\phi(h_{XG}) \stackrel{def}{=} h_{XH}$. The function ϕ is injective, by Lemma 4.49 . It remains to show that ϕ is well defined which requires to show that $\phi(h_{XG})$ is an element in $\text{Mono}(X, H, \neg m', \neg r')_{\text{F}}$, which is equivalent to show the following three conditions:

1. h_{XH} cannot be factorized by r',
2. h_{XH} cannot be factorized by m',
3. there is a monomorphism $h_{FH} : F \rightarrowtail H$ such that $h_{XH} = f \star h_{FH}$.

By Proposition 4.43, $KCHR$ is a VK-square. Therefore, we have the following commutative diagram:



We show that h_{XH} cannot be factorized by m' by contradiction. Assume the contrary, i.e. there is a morphism $h_{XR} : X \rightarrowtail R$ such that $h_{XH} = h_{XR} \star m'$.

There is a unique monomorphism $h_{XL'} : X \rightarrowtail L'$ such that $Id_X = h_{XL'} \star h_{L'X}$ and $h_{XR} = h_{XL'} \star h_{L'R}$ because $L'XHR$ is a pullback and $Id_X \star h_{XH} = h_{XR} \star h_{RH}$. Thus, we have the following equalities

$$\begin{aligned} h_{XG} &= Id_X \star h_{XG} \\ &= h_{XL'} \star (h_{L'X} \star h_{XG}) \\ &= (h_{XL'} \star h_{L'L}) \star m \end{aligned}$$

which contradicts the fact that h_{XG} cannot be factorized by m for it is an element of $\text{Mono}(\mathcal{X}, G, \neg m, \neg l')_{\text{F}}$.

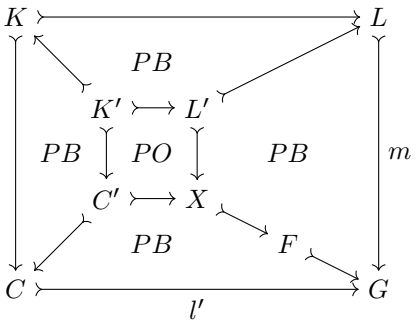
Symmetrically, $\phi(h_{XG})$ cannot be factorized by r'.

It remains to show that there is a monomorphism $h_{FH}: F \rightarrowtail H$ such that $h_{XH} = f \star h_{FH}$.

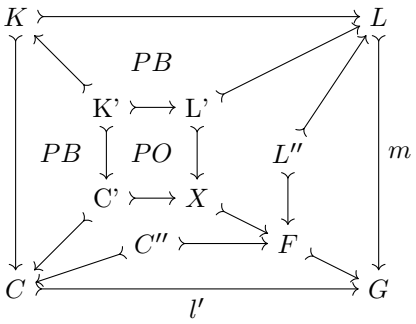
Since $h_{XG} \in \mathrm{Mono}(\mathcal{X}, G, \neg m, \neg l')_{\mathrm{F}}$, there is a morphism $h_{FG}: F \rightarrowtail G$ such that the following equality holds:

$$h_{XG} = f \star h_{FG}. \tag{5.7}$$

Thus, we have the following commutative diagram:



Since we work in **Graph**, we can construct the following commutative diagram where $L''LGF$ and $C''CGF$ are pullbacks:



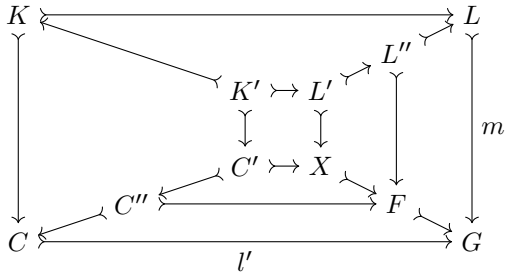
Since $L''LGF$ is pullback and $h_{L'X} \star h_{XF} \star h_{FG} = h_{L'L''} \star h_{L''L} \star h_{LG}$, by the universal property of pullback, there is a unique morphism $h_{L'L''}: L' \rightarrowtail L''$ such that

$$\begin{aligned} h_{L'L''} \star h_{L''L} &= h_{L'L}, \\ h_{L'X} \star h_{XF} &= h_{L'L''} \star h_{L''F}. \end{aligned} \tag{5.8}$$

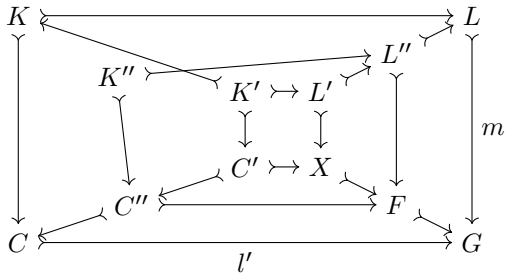
$L'L''L$ and $L'L''FX$ are commutative diagrams. Analogously, since $C''CGF$ is pullback and $C'CHFX$ is a commutative diagram, there is a unique morphism $h_{C'C''}: C' \rightarrowtail C''$ such that

$$\begin{aligned} h_{C'C''} \star h_{C''C} &= h_{C'C}, \\ h_{C'C''} \star h_{C''F} &= h_{C'X} \star h_{XF}. \end{aligned} \tag{5.9}$$

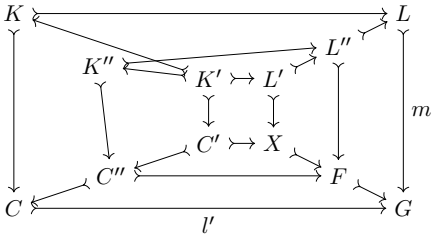
Thus, the following diagram holds:



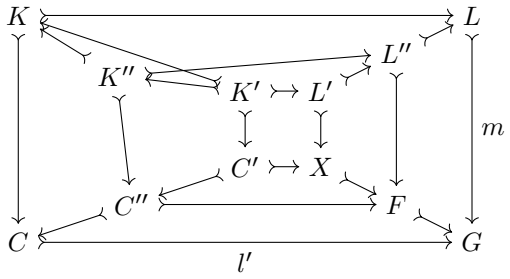
Since we work in **Graph**, we can construct $C'' \nleftarrow K'' \nrightarrow L''$ the pullback of $C'' \nrightarrow F \nleftarrow L''$, and the following commutative diagram holds:



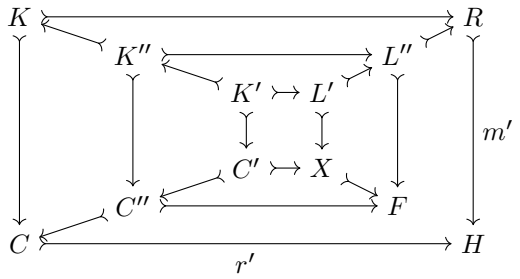
Since $K''C''FL''$ is pullback and $K'C'C''FL''L'$ is a commutative diagram, by the universal property of pullback, there is a unique morphism $h_{K'K''}: K' \nrightarrow K''$ such that $K'C'C''K''$ and $K'L'L''K''$ in the following diagram are commutative:



Similarly, since $KCGL$ is pullback and $K''C''CGLL''$ is a commutative diagram, by the universal property of pullback, there is a unique morphisms $h_{K''K}: K'' \nrightarrow K$ such that $K''C''CK$, $K''L''LK$ in the following diagram are commutative:



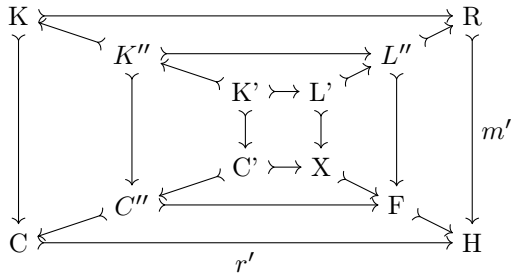
There is a monomorphism $h_{L''R} : L'' \nrightarrow R$ such that $K''L''RK$ is pullback, because ρ^{-1} is X-non-increasing by assumption. We have, thus, the following commutative diagram:



The square $K''C''FL''$ is pushout because it is pullback, by Proposition 2.27. Furthermore, we have

$$\begin{aligned}
&(h_{K''C''} \star h_{C''C}) \star h_{CH} \\
=&(h_{K''K} \star h_{KC}) \star h_{CH} \\
=&h_{K''K} \star (h_{KR} \star h_{RH}) \\
=&h_{K''L''} \star h_{L''R} \star h_{RH}.
\end{aligned}$$

Therefore, by the universal property of pushout, there is a unique morphism h_{FH} such that $L''FHR$ and $C''FHC$ are commutative diagrams. Thus, we have the following commutative diagram:



We have $h_{XH} = h_{XF} \star h_{FH}$ because $K'C'XL'$ is a pushout and h_{XH} is the unique morphism such that

$$\begin{aligned}
H_{C'X} \star h_{XH} &= h_{C'C} \star h_{CH} \\
&= h_{C'C''} \star h_{C''C} \star h_{CH} && \text{by (5.9)}
\end{aligned}$$

and

$$\begin{aligned}
h_{L'X} \star h_{XH} &= h_{L'R} \star h_{RH} \\
&= h_{L'L''} \star h_{L''R} \star h_{RH}, && \text{by (5.8)}
\end{aligned}$$

but we have

$$\begin{aligned}
&h_{C'X} \star (h_{XF} \star h_{FH}) \\
=&(h_{C'C''} \star h_{C''F}) \star h_{FH} \\
=&h_{C'C''} \star (h_{C''C} \star h_{CH})
\end{aligned}$$

and

$$\begin{aligned} & h_{L'X} \star (h_{XF} \star h_{FH}) \\ =& (h_{L'L''} \star h_{L''F}) \star h_{FH} \\ =& h_{L'L''} \star (h_{L''R} \star h_{RH}). \end{aligned}$$

This concludes our proof. □

Theorem **5.9**(Termination)

Let $\mathcal{A}$ and $\mathcal{B}$ be sets of injective DPO rewriting rules, $\mathbb{X}$ a set of ruler-graphs, and $s_{\mathbb{X}}$ a weight function. If the following conditions hold:

1. for every $\rho \in \mathcal{A} \cup \mathcal{B}$ and for every $\mathcal{X} = (X, \mathcal{C}) \in \mathbb{X}$, ρ is X-non-increasing, and if $\mathcal{C} = \{f : X \nrightarrow F\}$ then ρ^{-1} is X-non-increasing and F-non-increasing, and
2. for every $\rho \in \mathcal{A}$, we have $\sum_{\mathcal{X} \in \mathbb{X}} s_{\mathbb{X}}(\mathcal{X}) * \Lambda(\mathcal{X}, \rho) > 0$, and
3. for every $\rho \in \mathcal{B}$, we have $\sum_{\mathcal{X} \in \mathbb{X}} s_{\mathbb{X}}(\mathcal{X}) * \Lambda(\mathcal{X}, \rho) \geq 0$.

Then $\Rightarrow_{\mathcal{A},\mathcal{M}}$ terminates relative to $\Rightarrow_{\mathcal{B},\mathcal{M}}$.

Proof. By the assumption 1 and Lemma 5.8, the following inequality holds for all rewriting steps $G \Rightarrow_{\rho,\mathcal{M}} H$ with $\rho \in \mathcal{A} \cup \mathcal{B}$:

$$w_{s_{\mathbb{X}}}(G) - w_{s_{\mathbb{X}}}(H) \geq \sum_{\mathcal{X} \in \mathbb{X}} s_{\mathbb{X}}(\mathcal{X}) * (|\operatorname{Mono}(X, L)| - |\operatorname{Mono}(X, R)|)\,.$$

From the assumptions 2 and 3, we deduce

- $w_{s_{\mathbb{X}}}(G) > w_{s_{\mathbb{X}}}(H)$ for every rule $\rho \in \mathcal{A}$;
- $w_{s_{\mathbb{X}}}(G) \geq w_{s_{\mathbb{X}}}(H)$ for every rule $\rho \in \mathcal{B}$.

Since $w_{s_{\mathbb{X}}}(G) \in \mathbb{N}$ for all graph G. Every rewriting chain can only have a finite number of rewriting steps using rules from $\mathcal{A}$. □

Chapter 6

LyonParallel—A Tool for Relative Termination of DPO Graph Rewriting

LyonParallel is a prototype tool, written in OCaml, for the analysis of DPO graph rewriting systems. It implements the type graph method presented by Endrullis and Overbeek [39] and its extension in Chapter 3, and it also implements the morphism counting method in Chapter 4 and its extension in Chapter 5. For DPO graph rewriting systems with monic matches, the type graph method presented by Endrullis and Overbeek [39] is more powerful than the versions presented by Zantema et al. and Bruggink et al. [85, 19, 17], which are implemented in TORPAcyc [84] and Grez [20]. At the time of writing and to the best of our knowledge, the method presented by Overbeek and Endrullis has no publicly available implementation.

LyonParallel implements the notions of DPO rewriting systems on finite directed, edge-labeled multigraphs introduced in Chapter 2. Using the `REPL` (Read-Eval-Print Loop)—an interactive environment that reads user commands, evaluates them, prints results, and waits for the next input—users can select a set of rules encoded in the tool, then choose a method to partition the rules of the system into two sets $\mathcal{A}$ and $\mathcal{B}$ such that $\mathcal{A}$ terminates relative to $\mathcal{B}$. If such a partition is found, the tool replaces the current rule set with $\mathcal{B}$ and a new iteration begins; otherwise, users may try a different method. When the rule set becomes empty, the termination of the original system is proved.

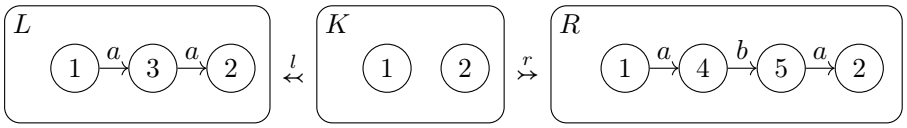
6.1 DPO graph rewriting systems

6.1.1 Labeled graphs

A labeled graph is encoded in the tool as a structure with five fields: a list of nodes, a list of edges, a mapping that associates to each edge a source node, a mapping that associates to each edge a target node, and a mapping that associates to each edge a label. The representation of a labeled graph is constructed using the function $MGraph.fromList$ given 2 parameters. The first

parameter is a list of integers representing the nodes of the graph. The second parameter is a list of four-tuples representing the labeled edges of the graph, where each tuple (s, lab, t, id) of type $int \times string \times int \times int$ represents an edge with source node s, label lab, target node t, and identifier id.

For example, consider the graph rewriting rule:



The representation of graphs K,L, and R in OCaml can be constructed by the following code:

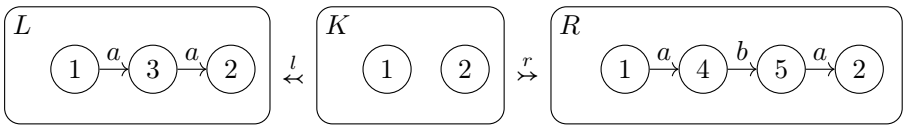
```
let graph_K = MGraph.fromList [1;2] []
let graph_L = Mgraph.fromList [1;2;3]
                [(1,"a",3,1);(3,"a",2,2)]
let graph_R = Mgraph.fromList [1;2;4;5]
                [(1,"a",4,1);(4,"b",5,2);(5,"a",2,3)]
```

6.1.2 Homomorphisms

A homomorphism between two labeled graphs is encoded as a mapping from the nodes of the source graph to the nodes of the target graph, which preserves the edge labels and the source and target nodes of each edge. Its representation can be constructed using the function $GraphHomomorphism.fromList$, given 4 arguments:

1. the first argument is the source graph,
2. the second argument is the target graph,
3. the third argument is a list of pairs, where each pair (s, t) means that node s of the source graph is mapped to node t of the target graph,
4. the fourth argument is a list of pairs similar to the third argument, but for the edges.

For example, consider morphisms l and r of the rewriting rule:



Their representation in OCaml can be constructed using the following code, where $morphism\_l$ represents l and $morphism\_r$ represents r, respectively, and $graph\_K$, $graph\_L$, and $graph\_R$ represent graphs K, L, and R, respectively (see Section 6.1.1):

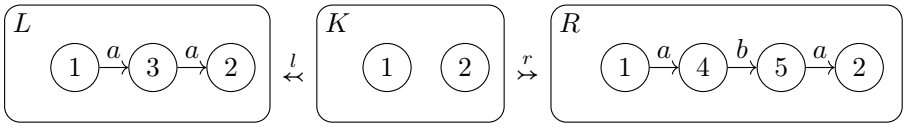
```
let morphism_l = GraphHomomorphism.fromList
                   graph_K graph_L [(1,1);(2,2)] []
let morphism_r = GraphHomomorphism.fromList
                   graph_K graph_R [(1,1);(2,2)] []
```

6.1.3 DPO rewriting rules and systems

A DPO rewriting rule is represented by a structure with two fields: a left-hand-side homomorphism and a right-hand side homomorphism. The representation of a rewriting rule can be constructed using the function

GraphRewritingSystem.DPOrule.fromHomos

given the representations of the left- and right-hand side homomorphisms. For example, consider the rewriting rule:



The representation of the rewriting rule is constructed using the following OCaml expression, which creates the representation of a DPO rule from the representations of the left and right morphisms. Here, *morphism_l* and *morphism_r* denote the representations of l and r, respectively (see Section 6.1.2):

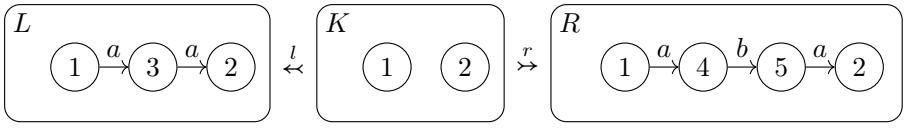
```
let rl = GraphRewritingSystem.DPOrule.fromHomos
           morphism_l morphism_r
```

A DPO graph rewriting system is represented by a structure with three fields: the set of rules, the name of the system, and a Boolean indicating whether the system is restricted to monic matches. A representation of the DPO graph rewriting system can be constructed using the function

GraphRewritingSystem.fromRulesListAndName

given three arguments: a list of rewriting rules and the name of the system; the third parameter is *false* by default.

For example, consider the DPO graph rewriting system that consists of the rule:



It is represented by the following OCaml expression that creates a system from the singleton list containing the representation of the rule. Here, *rl* denotes the representation of the rewriting rule.

```
let bruggink14_ex_4_and_6 =
    GraphRewritingSystem.fromRulesListAndName
      [rl] "bruggink14_ex_4_and_6"
```

The function *GraphRewritingSystem.fromRulesListAndName* produces a system named *bruggink14_ex_4_and_6* from the singleton list *[rl]*.

6.1.4 User-defined rewriting systems

User-defined rewriting systems should be placed in the file

lib/concretGraphRewritingSystems.ml

To expose them, add each instance to the list

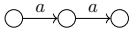
available_graph_rewriting_systems

in the same file.

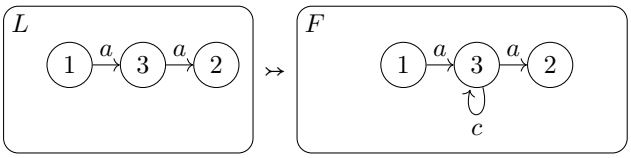
For example, for the rewriting rule named *bruggink14_ex_4_and_6*, the following code should be added to the end of the file:

```
let available_graph_rewriting_systems =
    bruggink14_ex_4_and_6 ::
        available_graph_rewriting_systems
```

6.2 Ruler-Graphs

A ruler-graph with one forbidden context as defined in Chapter 5 is represented by a structure with an underlying graph X and an injective graph homomorphism from X. Consider a ruler-graph (X, f) where X is the graph 

and the injective graph homomorphism is shown below:



Its representation in OCaml can be constructed using the following code:

```
let graph_X = MGraph.fromList
                [1;2;3] [(1,"a",3,1);(3,"a",2,2)]
let ruler_graph_aa_notin_aca =
    let graph_F = MGraph.fromList
            [1;2;3] [(1,"a",3,1);(3,"c",3,3);(3,"a"
                ,2,2)] in
    let f = Homo.fromList graph_X graph_F
                [(1,1);(2,2);(3,3)] [(1,1);(2,2)] in
    {x: graph_X; fx = Some f}
```

Figure 6.1

Instances of ruler-graphs can be defined in the file *lib/ruler_graph.ml* of **LyonParallel**. To expose these instances to the REPL, they should be added

to the list *ruler_graphs* in the same file. For example, for the instance of ruler-graph *ruler_graph_aa_notin_aca* shown in Figure 6.1, the following code should be added to the end of the file:

```
let ruler_graphs = ruler_graph_aa_notin_aca ::
    ruler_graphs
```

6.3 Installation and Execution of LyonParallel

This section shows the installation process using OPAM—a package manager for OCaml. To install LyonParallel, one needs to have OPAM installed on one's system, then run the following command

```
>> opam switch create lyonParallel 5.2.1
```

and follow the instructions to create and switch to a new OPAM switch. The dependencies of LyonParallel are listed in the file *LyonParallel.opam* in the root directory of the project. To install these dependencies, run the following command:

```
>> opam install .
```

The project can be built and installed by running the following command:

```
>> dune build && dune install
```

The following command uninstalls the tool:

```
>> opam remove LyonParallel
```

Finally, the following command launches the interactive REPL of the tool:

```
>> LyonParallel
```

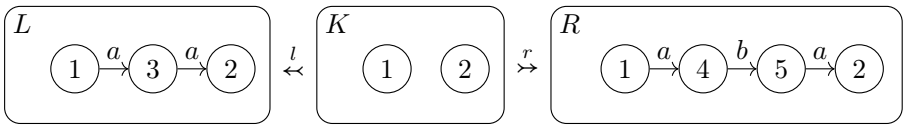
Upon launching, one should see:

```
>> Type "help" for a list of commands.
```

6.4 Relative Termination Analysis using LyonParallel

The LyonParallel REPL lets the user select a set of rewriting rules and run the tool's relative termination analysis techniques to test whether the set can be partitioned into two subsets $\mathcal{A}$ and $\mathcal{B}$ such that $\mathcal{A}$ terminates relative to $\mathcal{B}$. If such a partition is found, the REPL replaces the analyzed rule set with $\mathcal{B}$ for further analysis; otherwise, the user can try alternative methods or different partitions.

The REPL command *systems* lists the available DPO graph rewriting systems; user-defined systems can be added as described in § 6.1. The predefined system consisting of the rule shown below is named *bruggink14_ex_4_and_6*:

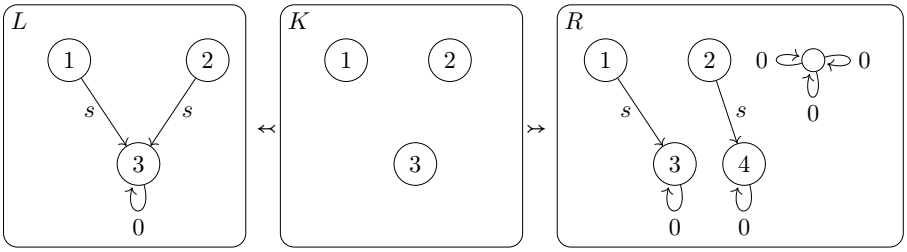


It appears in the list. Select it at the REPL by name:

```
>> select_system_by_name bruggink14_ex_4_and_6
```

6.4.1 Termination Analysis using the Morphism Counting Method

Consider the graph rewriting system presented in Example 4.28 with a single rule:



Its termination cannot be proved using the weighted type graph method presented in [85, 19, 17, 39] and in Chapter 3. However, its relative termination with respect to the empty set can be proved using the morphism counting method.

The following sequence of commands selects the system, then uses the morphism counting method without forbidden patterns presented in Chapter 4 to analyse the relative termination of the system.

```
>> select_system_by_name
    plump18_ex6_rule_copy_variant
>> subgraph_counting_no_forbidden_context
```

The following command summarizes the result of the termination analysis.

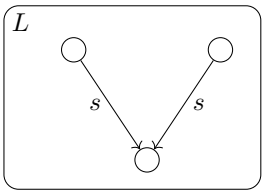
```
>> recap
```

The message displayed contains *Terminating: yes* when the method has proved the rule's relative termination with respect to the empty set; otherwise, the message contains *Terminating: unknown* and *Remaining Rules:*, followed by a list of rules to be analyzed.

Our implementation automatically tries every subgraph of the left-hand-side graphs of the system's rules as a candidate rule-graph until one succeeds. The chosen rule graph is then displayed in the analysis summary. For the system considered, the summary contains the information, shown below, which describes the employed rule graph.

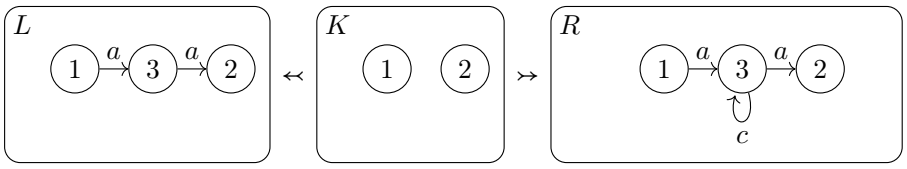
```
rule graph : nodes : [ 1;2;3 ]
arrows : [ (1,s,3,1);(2,s,3,3) ]
```

The ruler-graph is illustrated below.



6.4.2 Termination Analysis using the Morphism Counting with Antipattern

Consider the graph rewriting system shown in Example 5.11, whose termination cannot be proven either by the morphism counting method without forbidden patterns introduced in Chapter 4 or by the subgraph counting method presented in [66]. Its unique rule is illustrated below.



The rule is predefined in the tool and can be selected with the REPL command:

```
>> select_system_by_name bruggink14_ex1
```

The REPL provides the command *rulergraphs* to list available ruler-graphs. User-defined ruler-graphs can be added to this list using the method presented in the previous section. The ruler-graph shown in Figure 6.1 is predefined in the tool and appears in the list of available ruler-graphs.

The ruler-graph shown in Figure 6.1 is predefined and can be selected with the REPL command:

```
>> select_ruler_graph aa_not_in_aca
```

To analyse the system's relative termination using the morphism counting method with a forbidden pattern (see Chapter 5) with the ruler-graph, run:

```
>> subgraph_counting_one_forbidden_context
```

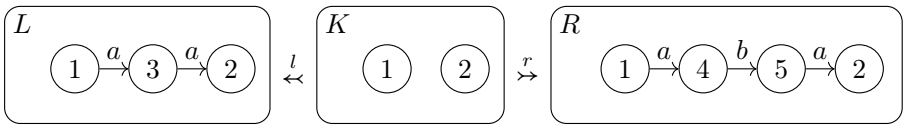
The following command displays a summary of the termination analysis.

```
>> recap
```

The message displayed contains *Terminating: yes*, which shows that the method applied successfully proved the relative termination of the rule with respect to the empty set of rules.

6.4.3 Termination Analysis using the Type Graph Method

Consider the rewriting rule shown below:



The rewriting system consisting of the rule is predefined in the tool and can be selected with the REPL command:

```
>> select_system_by_name bruggink14_ex_4_and_6
```

For a selected system, the following command analyses relative termination of some non-empty subset of rules of the system with respect to the set of the remaining rules using the type graph method with the natural arithmetic semiring. The timeout is set to 30 seconds.

```
>> type_graph 30 n
```

This command is an instance of the following command pattern

```
>> type_graph [timeout] [semiring_1 semiring_2 ...]
```

From the command pattern, we can see that more semirings can be selected. In fact, there are 6 semirings available in the tool, which can be selected by adding flags in the set $\{a, t, n, A, T, N\}$ where A, T, N stand for the arctic, tropical, and arithmetic semirings over the (extended) natural numbers; a, t, n stand for the arctic, tropical, and arithmetic semirings over the (extended) real numbers. If multiple semirings are selected, the tool runs the type graph method in parallel for each chosen semiring. When some rules are removed the tool automatically starts a new iteration on the remaining rules using the same selected semirings.

For example, we can have the following command which launches the type graph method with the timeout set to 30 seconds and all 6 available semirings.

```
>> type_graph 30 a n t A N T
```

Type graph methods with different semirings will be launched in parallel and cooperate with each other. When a non-empty subset A of rules is proved to be terminating relative to the set of remaining rules B, a new iteration starts with the set of rules B.

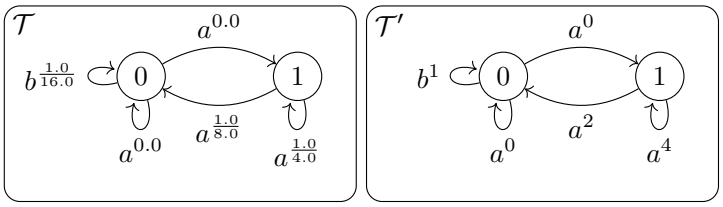
The command *recap* displays a summary of the termination analysis. The summary contains *Terminating: yes* and *Remaining Rules: none*, showing that the method successfully proved the rule's relative termination with respect to the empty rule set. The summary contains also the information shown below.

```
1 rule eliminated : rule 0
0 rule remaind : None

0--a->0 : 0.0
1--a->1 : 1.0/4.0
0--a->1 : 0.0
0--b->0 : 1.0/16.0
1--a->0 : 1.0/8.0
```

```
0—a—>0  :  0
1—a—>1  :  4
0—a—>1  :  0
0—b—>0  :  1
1—a—>0  :  2
```

The first block indicates that the rule whose representation is indexed by `0` (here the system's unique rule) can be eliminated using the type graph method over the real arctic semiring; no rules remain to be analyzed. The second block gives the information for constructing the type graph $\mathcal{T}$: two nodes and 5 edges with rational weights. The third block shows an equivalent type graph $\mathcal{T}'$ with integer weights obtained by multiplying the rational weights by their least common multiple. These type graphs are shown below, where the numbers inside the nodes are their identifiers.



6.5 Availability and license

LyonParallel is available at `https://github.com/Qi-tchi/LyonParallel/tree/thesis`. The project is distributed under the LGPL-2.1 License.

Chapter 7

Conclusion and Future Work

This dissertation focused on automated techniques for proving the relative termination of DPO graph rewriting systems.

We first extended an existing technique, the type graph method, to non-well-founded semirings, enabling more efficient implementation of the method. A new automated termination technique, called morphism counting, is also proposed for proving relative termination of injective DPO graph rewriting systems on edge-labeled directed multigraphs. It is based on counting monomorphisms from specific pattern-graphs to the graph before and after transformation. This technique can handle cases, such as Example 4.28, that existing interpretation-based approaches cannot. Finally, we extended our morphism counting method to count specific subgraphs that are not included in forbidden contexts. This extension successfully proves termination for systems like those presented in Example 5.11 and Example 5.12 which prior interpretation-based approaches [85, 19, 17, 38, 66] and the morphism counting method presented in Chapter 4 fail to handle.

An implementation of the type graph method and its extension, along with our morphism counting technique and its extension, was integrated into a unified tool, called LyonParallel, written in OCaml.

For future work, we outline directions that provide a roadmap from immediate improvements to broader theoretical generalizations and increased trustworthiness of the methods and tool presented in this dissertation.

In the short term (months), we will:

- Generalize our morphism counting method (Chapter 5) to count subgraphs with multiple forbidden contexts, enhancing its applicability to a broader range of graph rewriting systems.
- Investigate the differences between our morphism counting technique and the technique developed by Overbeek and Endrullis for PBPO+ [66], with the goal of understanding their relationship and moving toward a more general termination criterion for DPO rewriting systems or PBPO+.

In the mid term (2–5 years), we plan to:

- Develop a formal proof of correctness of the extended type graph method and the morphism counting technique using a proof assistant, to increase trustworthiness and prevent potential proof errors [23].

- Equip our tool with a certificate-generation mechanism that produces artifacts checkable by proof assistants, thereby mitigating implementation bugs and strengthening result reliability [25, 27].

In the long term (5+ years), we aim to:

- Extend the type graph method and the morphism counting technique to other algebraic graph rewriting formalisms such as PBPO+ [67].
- Extend the morphism counting technique to graph rewriting systems with negative application conditions. This could lead to a more practical termination tool, as it is very convenient to use negative application conditions to express constraints on the application of rules in graph transformation systems.

Bibliography

[1] Dimitri Ara, Albert Burroni, Yves Guiraud, Philippe Malbos, François Métayer, and Samuel Mimram. "Polygraphs: From Rewriting to Higher Categories". In: *CoRR* abs/2312.00429 (2023). DOI: 10.48550/ARXIV.2312.00429. arXiv: 2312.00429.

[2] Agnès Arnould, Hakim Belhaouari, Thomas Bellet, Pascale Le Gall, and Romain Pascual. "Preserving consistency in geometric modeling with graph transformations". In: *Math. Struct. Comput. Sci.* 32.3 (2022), pp. 300–347. DOI: 10.1017/S0960129522000226.

[3] Thomas Arts and Jürgen Giesl. "Termination of term rewriting using dependency pairs". In: *Theor. Comput. Sci.* 236.1-2 (2000), pp. 133–178. DOI: 10.1016/S0304-3975(99)00207-8.

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

[5] 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.

[6] Nicolas Behr, Russ Harmer, and Jean Krivine. "Concurrency Theorems for Non-linear Rewriting Theories". In: *CoRR* abs/2105.02842 (2021). arXiv: 2105.02842.

[7] Nicolas Behr, Russ Harmer, and Jean Krivine. "Fundamentals of compositional rewriting theory". In: *J. Log. Algebraic Methods Program.* 135 (2023), p. 100893. DOI: 10.1016/J.JLAMP.2023.100893.

[8] Nicolas Behr, Jean Krivine, Jakob L. Andersen, and Daniel Merkle. "Rewriting theory for the life sciences: A unifying theory of CTMC semantics". In: *Theor. Comput. Sci.* 884 (2021), pp. 68–115. DOI: 10.1016/J.TCS.2021.07.026.

[9] Hakim Belhaouari, Agnès Arnould, Pascale Le Gall, and Thomas Bellet. "Jerboa: A Graph Transformation Library for Topology-Based Geometric Modeling". In: *Graph Transformation - 7th International Conference, ICGT 2014, Held as Part of STAF 2014, York, UK, July 22-24, 2014. Proceedings*. Ed. by Holger Giese and Barbara König. Vol. 8571. Lecture Notes in Computer Science. Springer, 2014, pp. 269–284. DOI: 10.1007/978-3-319-09108-2_18.

[10] Thomas Bellet, Agnès Arnould, Hakim Belhaouari, and Pascale Le Gall. "Geometric Modeling: Consistency Preservation Using Two-Layered Variable Substitutions". In: *Graph Transformation*. Ed. by Juan de Lara and Detlef Plump. Cham: Springer International Publishing, 2017, pp. 36–53. ISBN: 978-3-319-61470-0.

[11] Clara Bertolissi, Maribel Fernández, and Bhavani Thuraisingham. "Category-Based Administrative Access Control Policies". In: *ACM Trans. Priv. Secur.* 28.1 (2025), 3:1–3:35. DOI: 10.1145/3698199.

[12] Clara Bertolissi, Maribel Fernández, and Bhavani Thuraisingham. "Graph-Based Specification of Admin-CBAC Policies". In: *CODASPY '21: Eleventh ACM Conference on Data and Application Security and Privacy, Virtual Event, USA, April 26-28, 2021*. Ed. by Anupam Joshi, Barbara Carminati, and Rakesh M. Verma. ACM, 2021, pp. 173–184. DOI: 10.1145/3422337.3447850.

[13] Nikolaj Bjorner and Lev Nachmanson. "Arithmetic Solving in Z3". In: *Computer Aided Verification*. Ed. by Arie Gurfinkel and Vijay Ganesh. Cham: Springer Nature Switzerland, 2024, pp. 26–41. ISBN: 978-3-031-65627-9.

[14] Paolo Bottoni, Manuel Koch, Francesco Parisi-Presicce, and Gabriele Taentzer. "Termination of High-Level Replacement Units with Application to Model Transformation". In: *Proceedings of the Workshop on Visual Languages and Formal Methods, VLFM 2004, Rome, Italy, September 30, 2004*. Ed. by Mark Minas. Vol. 127. Electronic Notes in Theoretical Computer Science 4. Elsevier, 2004, pp. 71–86. DOI: 10.1016/J.ENTCS.2004.08.048.

[15] Paolo Bottoni and Francesco Parisi-Presicce. "A Termination Criterion for Graph Transformations with Negative Application Conditions". In: *Electron. Commun. Eur. Assoc. Softw. Sci. Technol.* 30 (2010). DOI: 10.14279/TUJ.ECEASST.30.419.

[16] Benjamin Braatz, Ulrike Golas, and Thomas Soboll. "How to delete categorically - Two pushout complement constructions". In: *J. Symb. Comput.* 46.3 (Mar. 2011), pp. 246–271. ISSN: 0747-7171. DOI: 10.1016/j.jsc.2010.09.007.

[17] H. J. Sander Bruggink, Barbara König, Dennis Nolte, and Hans Zantema. "Proving Termination of Graph Transformation Systems using Weighted Type Graphs over Semirings". In: *CoRR* abs/1505.01695 (2015). arXiv: 1505.01695.

[18] H. J. Sander Bruggink, Barbara König, Dennis Nolte, and Hans Zantema. *Proving Termination of Graph Transformation Systems using Weighted Type Graphs over Semirings*. 2023. arXiv: 1505.01695v3 [cs.LO].

[19] 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 Díaz, 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. URL: https://doi.org/10.1007/978-3-662-44602-7_15.

[20] H.J. Sander Bruggink. *Grez*. https://www.ti.inf.uni-due.de/research/tools/grez/. Accessed: 2024-11-07. 2024.

[21] Jessica Carter. "Categories for the working mathematician: making the impossible possible". In: *Synth.* 162.1 (2008), pp. 1–13. DOI: 10.1007/S11229-007-9166-9.

[22] George E. Collins. "Quantifier elimination for real closed fields by cylindrical algebraic decomposition-preliminary report". In: *SIGSAM Bull.* 8.3 (1974), pp. 80–90. DOI: 10.1145/1086837.1086852.

[23] Evelyne Contejean. "A Certified AC Matching Algorithm". In: *Rewriting Techniques and Applications, 15th International Conference, RTA 2004, Aachen, Germany, June 3-5, 2004, Proceedings*. Ed. by Vincent van Oostrom. Vol. 3091. Lecture Notes in Computer Science. Springer, 2004, pp. 70–84. DOI: 10.1007/978-3-540-25979-4_5.

[24] Evelyne Contejean, Pierre Courtieu, Julien Forest, Olivier Pons, and Xavier Urbain. "Automated Certified Proofs with CiME3". In: *Proceedings of the 22nd International Conference on Rewriting Techniques and Applications, RTA 2011, May 30 - June 1, 2011, Novi Sad, Serbia*. Ed. by Manfred Schmidt-Schauß. Vol. 10. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fur Informatik, 2011, pp. 21–30. DOI: 10.4230/LIPICS.RTA.2011.21.

[25] Evelyne Contejean, Pierre Courtieu, Julien Forest, Olivier Pons, and Xavier Urbain. "Certification of Automated Termination Proofs". In: *Frontiers of Combining Systems, 6th International Symposium, FroCoS 2007, Liverpool, UK, September 10-12, 2007, Proceedings*. Ed. by Boris Konev and Frank Wolter. Vol. 4720. Lecture Notes in Computer Science. Springer, 2007, pp. 148–162. DOI: 10.1007/978-3-540-74621-8_10.

[26] Evelyne Contejean, Claude Marché, Ana Paula Tomás, and Xavier Urbain. "Mechanically Proving Termination Using Polynomial Interpretations". In: *J. Autom. Reason.* 34.4 (2005), pp. 325–363. DOI: 10.1007/S10817-005-9022-X.

[27] Evelyne Contejean, Andrey Paskevich, Xavier Urbain, Pierre Courtieu, Olivier Pons, and Julien Forest. "A3PAT, an approach for certified automated termination proofs". In: *Proceedings of the 2010 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation, PEPM 2010, Madrid, Spain, January 18-19, 2010*. Ed. by John P. Gallagher and Janis Voigtländer. ACM, 2010, pp. 63–72. DOI: 10.1145/1706356.1706370.

[28] Andrea Corradini, Dominique Duval, Rachid Echahed, Frederic Prost, and Leila Ribeiro. "AGREE - Algebraic Graph Rewriting with Controlled Embedding". In: *Graph Transformation - 8th International Conference, ICGT 2015, Held as Part of STAF 2015, L'Aquila, Italy, July 21-23, 2015. Proceedings*. Ed. by Francesco Parisi-Presicce and Bernhard Westfechtel. Vol. 9151. Lecture Notes in Computer Science. Springer, 2015, pp. 35–51. DOI: 10.1007/978-3-319-21145-9_3.

[29] Andrea Corradini, Dominique Duval, Rachid Echahed, Frederic Prost, and Leila Ribeiro. "The PBPO graph transformation approach". In: *J. Log. Algebraic Methods Program.* 103 (2019), pp. 213–231. DOI: 10.1016/J.JLAMP.2018.12.003.

[30] Andrea Corradini, Ugo Montanari, Francesca Rossi, Hartmut Ehrig, Reiko Heckel, and Michael Lowe. "Algebraic Approaches to Graph Transformation - Part I: Basic Concepts and Double Pushout Approach". In: *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*. Ed. by Grzegorz Rozenberg. World Scientific, 1997, pp. 163–246.

[31] Nachum Dershowitz. "Orderings for Term-Rewriting Systems". In: *Theor. Comput. Sci.* 17 (1982), pp. 279–301. DOI: 10.1016/0304-3975(82)90026-3.

[32] Bruno Dutertre. "Yices 2.2". In: *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*. Ed. by Armin Biere and Roderick Bloem. Vol. 8559. Lecture Notes in Computer Science. Springer, 2014, pp. 737–744. DOI: 10.1007/978-3-319-08867-9_49.

[33] H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, eds. *Handbook of graph grammars and computing by graph transformation: vol. 2: applications, languages, and tools*. USA: World Scientific Publishing Co., Inc., 1999. ISBN: 9810240201.

[34] H. Ehrig, H.-J. Kreowski, U. Montanari, and G. Rozenberg, eds. *Handbook of graph grammars and computing by graph transformation: vol. 3: concurrency, parallelism, and distribution*. USA: World Scientific Publishing Co., Inc., 1999. ISBN: 981024021X.

[35] Hartmut Ehrig, Karsten Ehrig, Ulrike Prange, and Gabriele Taentzer. *Fundamentals of Algebraic Graph Transformation*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2006. ISBN: 978-3-540-31187-4. DOI: 10.1007/3-540-31188-2.

[36] Hartmut Ehrig, Reiko Heckel, Martin Korff, Michael Lowe, Leila Ribeiro, Annika Wagner, and Andrea Corradini. "Algebraic Approaches to Graph Transformation - Part II: Single Pushout Approach and Comparison with Double Pushout Approach". In: *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*. Ed. by Grzegorz Rozenberg. World Scientific, 1997, pp. 247–312.

[37] 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.

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

[39] 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.

[40] Jorg Endrullis and Roy Overbeek. *Termination of Graph Transformation Systems via Generalized Weighted Type Graphs*. 2024. arXiv: 2307.07601v3 [cs.LO].

[41] Alfons Geser. "Relative Termination". PhD thesis. University of Passau, Germany, 1990.

[42] Jurgen Giesl, Marc Brockschmidt, Fabian Emmes, Florian Frohn, Carsten Fuhs, Carsten Otto, Martin Plucker, Peter Schneider-Kamp, Thomas Stroder, Stephanie Swiderski, and Rene Thiemann. "Proving Termination of Programs Automatically with AProVE". In: *Automated Reasoning - 7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 19-22, 2014. Proceedings*. Ed. by Stephane Demri, Deepak Kapur, and Christoph Weidenbach. Vol. 8562. Lecture Notes in Computer Science. Springer, 2014, pp. 184–191. DOI: 10.1007/978-3-319-08587-6_13.

[43] Annegret Habel, Jurgen Muller, and Detlef Plump. "Double-pushout graph transformation revisited". In: *Math. Struct. Comput. Sci.* 11.5 (2001), pp. 637–688. DOI: 10.1017/S0960129501003425.

[44] Tero Harju, Ion Petre, and Grzegorz Rozenberg. "Tutorial on DNA Computing and Graph Transformation". In: *Graph Transformations, Second International Conference, ICGT 2004, Rome, Italy, September 28 - October 2, 2004, Proceedings*. Ed. by Hartmut Ehrig, Gregor Engels, Francesco Parisi-Presicce, and Grzegorz Rozenberg. Vol. 3256. Lecture Notes in Computer Science. Springer, 2004, pp. 434–436. DOI: 10.1007/978-3-540-30203-2_31.

[45] Reiko Heckel and Gabriele Taentzer. *Graph Transformation for Software Engineers - With Applications to Model-Based Development and Domain-Specific Language Engineering*. Springer, 2020. ISBN: 978-3-030-43915-6. DOI: 10.1007/978-3-030-43916-3.

[46] Gernot Heiser, June Andronick, Kevin Elphinstone, Gerwin Klein, Ihor Kuz, and Leonid Ryzhyk. "The road to trustworthy systems". In: *Proceedings of the fifth ACM workshop on Scalable trusted computing, STC@CCS 2010, Chicago, IL, USA, October 4, 2010*. Ed. by Shouhuai Xu, N. Asokan, and Ahmad-Reza Sadeghi. ACM, 2010, pp. 3–10.

[47] Dejan Jovanovic and Leonardo de Moura. "Cutting to the Chase - Solving Linear Integer Arithmetic". In: *J. Autom. Reason.* 51.1 (2013), pp. 79–108. DOI: 10.1007/S10817-013-9281-X.

[48] Dejan Jovanovic and Leonardo de Moura. "Solving Non-linear Arithmetic". In: *Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings*. Ed. by Bernhard Gramlich, Dale Miller, and Uli Sattler. Vol. 7364. Lecture Notes in Computer Science. Springer, 2012, pp. 339–354. DOI: 10.1007/978-3-642-31365-3_27.

[49] Narendra Karmarkar. "A new polynomial-time algorithm for linear programming". In: *Comb.* 4.4 (1984), pp. 373–396. DOI: 10.1007/BF02579150. URL: https://doi.org/10.1007/BF02579150.

[50] Jan-Christoph Kassing, Grigory Vartanyan, and Jürgen Giesl. "A Dependency Pair Framework for Relative Termination of Term Rewriting". In: *Automated Reasoning - 12th International Joint Conference, IJCAR 2024, Nancy, France, July 3-6, 2024, Proceedings, Part II*. Ed. by Christoph Benzmüller, Marijn J. H. Heule, and Renate A. Schmidt. Vol. 14740. Lecture Notes in Computer Science. Springer, 2024, pp. 360–380. DOI: 10.1007/978-3-031-63501-4_19.

[51] Jan Willem Klop. "Term Rewriting Systems: A Tutorial". In: *Bulletin of the European Association for Theoretical Computer Science* (1987).

[52] Manuel Koch, Luigi V. Mancini, and Francesco Parisi-Presicce. "On the specification and evolution of access control policies". In: *6th ACM Symposium on Access Control Models and Technologies, SACMAT 2001, Litton-TASC, Chantilly, Virginia, USA, May 3-4, 2001*. Ed. by Ravi S. Sandhu and Trent Jaeger. ACM, 2001, pp. 121–130. DOI: 10.1145/373256.373280.

[53] Barbara König, Dennis Nolte, Julia Padberg, and Arend Rensink. "A Tutorial on 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. 83–104. DOI: 10.1007/978-3-319-75396-6_5.

[54] Stephen Lack and Pawel Sobocinski. "Adhesive Categories". In: *Foundations of Software Science and Computation Structures, 7th International Conference, FOSSACS 2004, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2004, Barcelona, Spain, March 29 - April 2, 2004, Proceedings*. Ed. by Igor Walukiewicz. Vol. 2987. Lecture Notes in Computer Science. Springer, 2004, pp. 273–288. DOI: 10.1007/978-3-540-24727-2_20.

[55] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. "The Byzantine generals problem". In: *Concurrency: the Works of Leslie Lamport*. Ed. by Dahlia Malkhi. ACM, 2019, pp. 203–226. DOI: 10.1145/3335772.3335936.

[56] Tihamer Levendovszky, Ulrike Prange, and Hartmut Ehrig. "Termination Criteria for DPO Transformations with Injective Matches". In: *Proceedings of the Workshop on Graph Transformation for Concurrency and Verification, GT-VC@CONCUR 2006, Bonn, Germany, August 31, 2006*. Ed. by Arend Rensink, Reiko Heckel, and Barbara König. Vol. 175. Electronic Notes in Theoretical Computer Science 4. Elsevier, 2006, pp. 87–100. DOI: 10.1016/J.ENTCS.2007.04.019.

[57] Gavin Lowe. "Breaking and fixing the Needham-Schroeder Public-Key Protocol using FDR". In: *Tools and Algorithms for the Construction and Analysis of Systems*. Ed. by Tiziana Margaria and Bernhard Steffen. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 147–166. ISBN: 978-3-540-49874-2.

[58] Michael Lowe. "Graph Rewriting in Span-Categories". In: *Graph Transformations - 5th International Conference, ICGT 2010, Enschede, The Netherlands, September 27 - - October 2, 2010. Proceedings*. Ed. by Hartmut Ehrig, Arend Rensink, Grzegorz Rozenberg, and Andy Schurr. Vol. 6372.

Lecture Notes in Computer Science. Springer, 2010, pp. 218–233. DOI: 10.1007/978-3-642-15928-2_15.

[59] 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.

[60] Claude Marché and Xavier Urbain. "Modular and incremental proofs of AC-termination". In: *J. Symb. Comput.* 38.1 (2004), pp. 873–897. DOI: 10.1016/J.JSC.2004.02.003.

[61] Yuri Matiyasevich. "Enumerable sets are diophantine". In: *Mathematical logic in the 20th century* (2003), pp. 269–273.

[62] Aart Middeldorp and Hans Zantema. "Simple Termination of Rewrite Systems". In: *Theor. Comput. Sci.* 175.1 (1997), pp. 127–158. DOI: 10.1016/S0304-3975(96)00172-7.

[63] Leonardo de Moura and Nikolaj S. Bjorner. "Z3: An Efficient SMT Solver". In: *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*. Ed. by C. R. Ramakrishnan and Jakob Rehof. Vol. 4963. Lecture Notes in Computer Science. Springer, 2008, pp. 337–340. DOI: 10.1007/978-3-540-78800-3_24.

[64] Roger M. Needham and Michael D. Schroeder. "Using encryption for authentication in large networks of computers". In: *Commun. ACM* 21.12 (Dec. 1978), pp. 993–999. ISSN: 0001-0782. DOI: 10.1145/359657.359659.

[65] Roy Overbeek and Jorg Endrullis. "A PBPO+ Graph Rewriting Tutorial". In: *Proceedings Twelfth International Workshop on Computing with Terms and Graphs, TERMGRAPH@FSCD 2022, Technion, Haifa, Israel, 1st August 2022*. Ed. by Clemens Grabmayer. Vol. 377. EPTCS. 2022, pp. 45–63. DOI: 10.4204/EPTCS.377.3.

[66] Roy Overbeek and Jorg Endrullis. "Termination of Graph Transformation Systems Using Weighted Subgraph Counting". In: *Logical Methods in Computer Science* Volume 20, Issue 4, 12 (Nov. 2024). ISSN: 1860-5974. DOI: 10.46298/lmcs-20(4:12)2024.

[67] Roy Overbeek, Jorg Endrullis, and Alois Rosset. "Graph rewriting and relabeling with PBPO$^+$: A unifying theory for quasitoposes". In: *J. Log. Algebraic Methods Program.* 133 (2023), p. 100873. DOI: 10.1016/J.JLAMP.2023.100873.

[68] Romain Pascual, Pascale Le Gall, Agnès Arnould, and Hakim Belhaouari. "Topological consistency preservation with graph transformation schemes". In: *Sci. Comput. Program.* 214 (2022), p. 102728. DOI: 10.1016/J.SCICO.2021.102728.

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

[70] Detlef Plump. "Hypergraph rewriting: critical pairs and undecidability of confluence". In: *Term Graph Rewriting: Theory and Practice*. GBR: John Wiley and Sons Ltd., 1993, pp. 201–213. ISBN: 0471935670.

[71] 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.

[72] 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.

[73] Detlef Plump. "Termination of Graph Rewriting is Undecidable". In: *Fundam. Informaticae* 33.2 (1998), pp. 201–209. DOI: 10.3233/FI-1998-33204.

[74] Detlef Plump. "The Graph Programming Language GP". In: *Algebraic Informatics, Third International Conference, CAI 2009, Thessaloniki, Greece, May 19-22, 2009, Proceedings*. Ed. by Symeon Bozapalidis and George Rahonis. Vol. 5725. Lecture Notes in Computer Science. Springer, 2009, pp. 99–122. DOI: 10.1007/978-3-642-03564-7_6.

[75] Mathieu Poudret, Agnès Arnould, Jean-Paul Comet, and Pascale Le Gall. "Graph Transformation for Topology Modelling". In: *Graph Transformations, 4th International Conference, ICGT 2008, Leicester, United Kingdom, September 7-13, 2008. Proceedings*. Ed. by Hartmut Ehrig, Reiko Heckel, Grzegorz Rozenberg, and Gabriele Taentzer. Vol. 5214. Lecture Notes in Computer Science. Springer, 2008, pp. 147–161. DOI: 10.1007/978-3-540-87405-8_11.

[76] Mathieu Poudret, Jean-Paul Comet, Pascale Le Gall, Agnès Arnould, and Philippe Meseure. "Topology-based Geometric Modelling for Biological Cellular Processes". In: *LATA 2007. Proceedings of the 1st International Conference on Language and Automata Theory and Applications*. Ed. by Remco Loos, Szilárd Zsolt Fazekas, and Carlos Martín-Vide. Vol. Report 35/07. Research Group on Mathematical Linguistics, Universitat Rovira i Virgili, Tarragona, 2007, pp. 497–508.

[77] Qi Qiu. *Termination of Graph Rewriting using Weighted Type Graphs over Non-well-founded Semirings*. Tech. rep. Universite Claude Bernard Lyon 1 (UCBL), Lyon, FRA., Feb. 2025. URL: https://hal.science/hal-04954960v2.

[78] 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.

[79] Grzegorz Rozenberg, ed. *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*. World Scientific, 1997. ISBN: 9810228848.

[80] Joachim Steinbach. “Simplification Orderings: Histrory of Results”. In: *Fundam. Informaticae* 24.1/2 (1995), pp. 47–87. DOI: 10.3233/FI-1995-24123.

[81] Alfred Tarski. “A Decision Method for Elementary Algebra and Geometry”. In: *Quantifier Elimination and Cylindrical Algebraic Decomposition*. Ed. by Bob F. Caviness and Jeremy R. Johnson. Vienna: Springer Vienna, 1998, pp. 24–84. ISBN: 978-3-7091-9459-1.

[82] Xavier Urbain. “Modular and Incremental Automated Termination Proofs”. In: *Journal of Automated Reasoning* 32.4 (2004). Received 16 January 2002; Accepted 15 April 2004, p. 315. DOI: 10.1007/BF03177743.

[83] Hans Zantema. “Termination of Term Rewriting by Semantic Labelling”. In: *Fundam. Informaticae* 24.1/2 (1995), pp. 89–105. DOI: 10.3233/FI-1995-24124.

[84] Hans Zantema. *TORPAcyc*. https://hzantema.win.tue.nl/torpa.html. Accessed: 2024-11-07. 2024.

[85] 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.

Index