master thesis in computer science

by

Maria A Schett

submitted to the Faculty of Mathematics, Computer Science and Physics of the University of Innsbruck

in partial fulfillment of the requirements for the degree of Master of Science

supervisor: Assoc. Prof. Dr. Georg Moser, Department of Computer Science

Innsbruck, 1 October 2016

Master Thesis

From Trees to Graphs: On the Influence of Collapsing on Rewriting and on Termination

Maria A Schett (0716526)
maria.schett@student.uibk.ac.at

1 October 2016

Supervisor: Assoc. Prof. Dr. Georg Moser

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt durch meine eigenhändige Unterschrift, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle Stellen, die wörtlich oder inhaltlich den angegebenen Quellen entnommen wurden, sind als solche kenntlich gemacht.
Die vorliegende Arbeit wurde bisher in gleicher oder ähnlicher Form noch nicht als Magister-/Master-/Diplomarbeit/Dissertation eingereicht.

Datum

Unterschrift

Abstract

When moving from the tree representation of terms to the graph representation of term graphs, we change the potential rewrite steps: every graph rewrite step can be simulated by one or more term rewrite steps—but not vice versa. In this work we are interested in the effects of collapsing equal sub-terms/graphs on the rewrite relation. Therefore we first comprehensively compare combinations of different collapsing relations with the rewrite relation. We conduct this comparison with respect to simulation of steps and normal forms. Next we study the effect of collapsing on termination. A straight-forward consequence of simulation is that infinitely many term graph rewrite steps imply infinitely many term rewrite steps—but again not vice versa. Hence sometimes term graph rewriting terminates, where term rewriting does not. We are interested in this gap and developed a termination technique to show termination for term graph rewriting: a lexicographic path order which we proved well-founded by adapting Kruskal's Tree Theorem for term graphs.

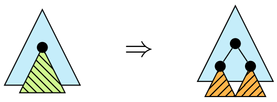
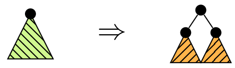
Acknowledgments

First and foremost I want to thank my advisor Georg Moser for his support throughout my master's programme and this thesis. I also want to thank my all colleagues at the CBR and CL research group. Last but not least my thanks go to my family, and of course, Julian.

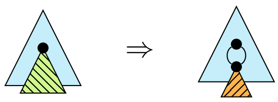
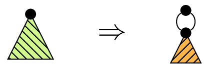
Contents

1 Introduction

Rewriting is the transformation of objects based on a set of directed equations, namely rewrite rules. If these objects are terms, we talk of term rewriting—a Turing-complete, yet simple, abstract model of computation [6, 34]. We rewrite:

 with the rule  $(\star_1)$.

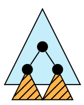
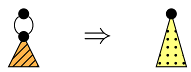
Inherent to terms is their tree structure. As in the step above, a sub-term can get duplicated by a rewrite step. Thereby rewrite steps can cause an exponential blow-up in size—even when this is not necessary. To avoid this unnecessary blow-up we move from the tree structure of terms to a graph structure: term graphs. Term graphs allow to share equal sub-graphs. When sharing the sub-graph  in the above rule and step, we rewrite:

 with the rule  $(\star_2)$.

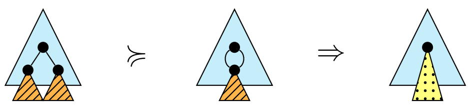
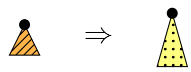
But not only can we share equal sub-graphs through a rewrite step—we can also add an explicit collapsing relation:

 .

To apply a rule sometimes we need collapsing to find the exact pattern of the rule in a term graph. Consider for example the following:

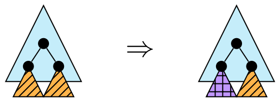
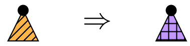
 does not match the rule  .

But if we collapse first we can find exactly the same structure and can apply the rule:

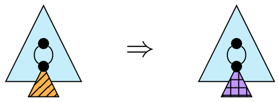
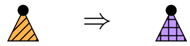
 with the rule  .

This raises several questions: How to sensibly combine these two relations: $\Rightarrow$ and $\succcurlyeq$? Through union like $\Rightarrow \cup \succcurlyeq$ or through concatenation like $\Rightarrow \cdot \succcurlyeq$? Do we first collapse and then rewrite or the other way around? Do we always collapse as much as possible? And what are the consequences of each of these decisions? To find answers to these questions is the first part of this thesis: to investigate the influence of collapsing on the rewrite relation.

We can easily see that sharing equal sub-graphs influences the potential rewrite steps. In $(\star_1)$ we can choose to rewrite only the first argument:

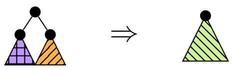
 with the rule  $(\star_3)$.

But if we try to simulate this in $(\star_2)$ we rewrite in parallel.

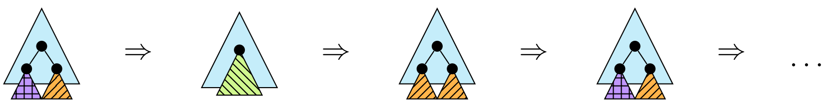
 with the rule  $(\star_4)$.

Term graph rewriting with collapsing *and* the reverse operation can simulate term rewriting with polynomial overhead and linear size growth [18, 2]. But simulation requires uncollapsing, which seems to annihilate the advantage of the efficient graph representation. Thus our aim is to investigate term graph rewriting with collapsing only. Unsurprisingly every term graph rewrite step can be simulated by n term rewrite steps but not vice versa.

As a direct consequence infinitely many term rewrite steps imply infinitely many graph rewrite steps, but again not vice versa. Hence we may have only finitely many rewrite steps, that is termination for term graphs, where we have infinitely many for terms. Suppose we add the following rule to $(\star_3)$ and $(\star_4)$:

 .

Now an infinite number of rewrite steps is possible combining $(\star_1)$ and $(\star_3)$:



But we cannot apply the same rule in $(\star_4)$. Hence, we have termination for term graphs in $(\star_2)$ but not for terms in $(\star_1)$. We are interested in this gap and want to find techniques to prove termination of term graph rewriting—in the absence of termination of term rewriting. This is the second part of this thesis: finding a termination technique directly for term graph rewriting.

After this introduction in Chapter 1, this thesis is structured as follows: In Chapter 2 are the preliminaries, where we fix notations for sets, relations, orders, and functions in Section 2.1, and give a brief introduction to term rewriting in Section 2.2. In Chapter 3 we present term graph rewriting by first introducing the underlying data structure in Section 3.1, then rewriting in Section 3.2, and in the final Section 3.3 we investigate the transfer between the term graph rewriting and the term rewriting setting. In Chapter 4 we combine the term graph rewrite relation with collapsing in different variations. In Section 4.1 we start with presenting results on why collapsing and its inverse are needed to simulate term rewriting. Then we drop the requirement of simulation and compare different combinations with respect to one-step-simulation and normal forms. We have three blocks, which we present in Section 4.2: in Section 4.3 we compare the combinations of the term graph rewrite relation and the collapsing relation through concatenation, in Section 4.4 through union, and in Section 4.5 we compare them with each other. The next Chapter 5 presents our results on Kruskal's Tree Theorem for term graphs, where we present our motivation and notion of argument graph in Section 5.1, develop our notion of embedding in Section 5.2, and give the proof in Section 5.3. In Chapter 6 we use the result of the previous chapter to present a termination order, more specifically a simplification order, in Section 6.1. In Section 6.2 we give some straight-forward non-termination results. In Chapter 7 we give an overview over the term graph rewriting literature. We presents different representations of term graphs in Section 7.1, results on termination in Section 7.2, confluence in Section 7.3, modularity in Section 7.4, and discuss the difference between collapsing and memoisation in Section 7.5. We conclude the thesis in Chapter 8 with a conclusion on the combination between collapsing and rewriting in Section 8.1, a conclusion on our results on termination in Section 8.2, and a conclusion on the term graph rewriting literature in Section 8.3.

2 Preliminaries

The goal of this chapter is to introduce and fix notation. In Section 2.1 we define sets, relations, orders, and functions, and in Section 2.2 we introduce term rewriting.

2.1 Sets, Relations, Orders, and Functions

Let A and B be sets, and let $\varnothing$ denote the empty set. With $|A|$ we denote the cardinality of A. We write $A \setminus B$ for set difference, $A \cap B$ for intersection, $A \cup B$ for union, $A \times B$ for the cartesian product, $A \subseteq B$ for the sub-set relation, $A \subsetneq B$ if $A \subseteq B$ and $A \neq B$, AB for set concatenation, and $\mathcal{P}(A)$ for the power set of A. The set A^* is defined as $\{a_0 \cdot a_1 \cdots a_n \mid n \geqslant 0 \text{ and } a_i \in A, 1 \leqslant i \leqslant n\}$ and $A^+ = AA^*$.

Next we define some standard notions on binary relations. We write $\rhd$ for an arbitrary binary relation, and define reflexivity, irreflexivity, symmetry, anti- and asymmetry, and transitivity.

Definition 2.1. A binary relation $\rhd$ over a set A is called

- *reflexive* if $a \rhd a$ holds for all $a \in A$,
- *irreflexive* if $a \rhd a$ holds for no $a \in A$,
- *symmetric* if $a \rhd b$ implies $b \rhd a$ for all $a, b \in A$,
- *anti-symmetric* if $a \rhd b$ and $b \rhd a$ implies $a = b$ for all $a, b \in A$,
- *asymmetric* if $a \rhd b$ implies $b \not\rhd a$ for all $a, b \in A$,
- *transitive* if $a \rhd b$ and $b \rhd c$ implies $a \rhd c$ for all $a, b, c \in A$.

We write $\rhd^=$ for the reflexive closure, and $\rhd^+$ for the transitive closure of $\rhd$. The transitive and reflexive closure of $\rhd$ is denoted by $\rhd^*$. Next we are interested in $\rhd$ with respect to its normal forms.

Definition 2.2. An element a is in *normal form* with respect to $\rhd$ if there is no element b with $a \rhd b$. We write $a \rhd^! b$, if $a \rhd^* b$ and b is in normal form. The set of normal forms for a relation $\rhd$ is denoted by $\mathsf{NF}(\rhd)$.

A binary relation can be strongly or weakly normalising, have unique normal forms, can be confluent, complete or semi-complete.

Definition 2.3. We say a binary relation $\rhd$ over A

- is *strongly normalising*, *well-founded*, or *terminating* if there are no infinite sequences $a_1 \triangleright a_2 \triangleright a_3 \dots$ with $a_i \in A$ for $i \geqslant 1$.
- is *weakly normalising* if for each element a there is a b such that $a \triangleright^! b$.
- has *unique normal forms*, if for all elements a where $a \triangleright^! b$ and $a \triangleright^! c$, $b = c$ holds.
- is *confluent*, if for elements a_1, a_2, and a_3 with $a_1 \triangleright^* a_2$ and $a_1 \triangleright^* a_3$ there exists an element a_4 such that $a_2 \triangleright^* a_4$ and $a_3 \triangleright^* a_4$.
- is *semi-complete* if it is weakly normalising and confluent.
- is *complete* if it is strongly normalising and confluent.

Based on binary relations we can define some standard notions on orders. We look at the definition of pre-order, partial order, and proper order.

Definition 2.4. Let A be a set.

- A *pre-order* is a reflexive, and transitive binary relation over A.
- A *partial order* on A is a reflexive, anti-symmetric, and transitive relation.
- A *proper order* on A is an irreflexive, and transitive binary relation over A.

We next look at the concept of good and bad sequences and define well-quasi orders and chains.

Definition 2.5. Let $\preccurlyeq$ be a pre-order on a set A. An infinite sequence $\mathbf{a}$ over A is called *good*, if there are indices i, j with $1 \leqslant i < j$ such that $a_i \preccurlyeq a_j$. Otherwise $\mathbf{a}$ is called *bad*. If every infinite sequence is good, $\preccurlyeq$ is a *well-quasi order (wqo)*. An infinite sequence $\mathbf{a}$ is a *chain* if $a_i \preccurlyeq a_{i+1}$ holds for all $i \geqslant 1$.

We can find a chain in every infinite sequence, as stated by the following lemma and shown in the proof.

Lemma 2.6. *If $\preccurlyeq$ is a wqo then every infinite sequence contains a chain.*

Proof. We follow the proof in [22]. Let $\mathbf{a}$ be an infinite sequence. By assumption, and definition of wqo, every infinite sequence is good. If for a sub-sequence with elements a_i we cannot find $a_i \preccurlyeq a_j$ for $j > i$, this sub-sequence is bad. If this sub-sequence is also infinite, this contradicts the assumption. Hence this sub-sequence is finite and we can find an index $N \geqslant 1$ such that for all $i \geqslant N$ we can find $a_i \preccurlyeq a_j$ for $j > i$. We construct a chain $\mathbf{a}_\phi$ where $\phi(1) := N$ and $\phi(i) := \min\{j \mid j > \phi(i-1) \text{ and } a_{\phi(i-1)} \not\succcurlyeq a_j\}$. ☐

Finally we consider functions. Let $f : A \to B$ be a function from a set A to a set B. We call A the *domain*, denoted by $\mathsf{dom}(f)$, and B the *co-domain*, of f. We write $f|_{A'}$ for restricting $\mathsf{dom}(f)$ to $A' \subseteq A$. The *inverse* of f is denoted by $f^{-1} : B \to A$.

Definition 2.7. For functions $f : A \to B$ and $g : B \to C$ we write $g \circ f : A \to C$ for *function composition*, where $g \circ f(x) = g(f(x))$.

This concludes the first part of the preliminaries. In the next section we introduce a simple, yet Turing-complete model of computation: *term rewriting*.

2.2 Term Rewriting

Term rewriting has its roots in equational reasoning. We reason by replacing equal terms by equal terms following directed equations, namely rewrite rules. Applying such a rewrite rule is a purely syntactical manipulation. This makes term rewriting an easy to understand model of computation close to first-order functional programming.

This section gives only a brief introduction to term rewriting. For a detailed introduction the kind reader is referred to [6], [34], or [22]. The notation is based on [22].

Definition 2.8. Let $\mathcal{F}$ be a finite set of function symbols, a *signature*, where every $f \in \mathcal{F}$ has an associated arity $\mathsf{ar}(f) \in \mathbb{N}$. Let $\mathcal{V}$ be a (countably infinite) set of variables disjoint from $\mathcal{F}$. Then $\mathcal{T}(\mathcal{F}, \mathcal{V})$ denotes the set of terms over $\mathcal{F}$ and $\mathcal{V}$.

We usually write $x, y, z \ldots$ for variables. For function symbols we write $f, g, h \ldots$, but when $\mathsf{ar}(f) = 0$, then f is called a *constant* and we denote constants by $a, b, c \ldots$ In $\mathcal{V}\mathsf{ar}(t)$ we collect all variables of a term t. If $\mathcal{V}\mathsf{ar}(t) = \varnothing$, then t is *ground*, and we write $t \in \mathcal{T}(\mathcal{F})$.

Example 2.9. Given a signature $\mathcal{F} = \{\mathsf{f}, \mathsf{g}, \mathsf{h}, \mathsf{a}\}$ with $\mathsf{ar}(\mathsf{f}) = 2$, $\mathsf{ar}(\mathsf{g}) = \mathsf{ar}(\mathsf{h}) = 1$, and a constant a. Then we have the following terms in $\mathcal{T}(\mathcal{F}, \mathcal{V})$:

$$\mathsf{f}(x, x) \quad , \quad \mathsf{f}(\mathsf{g}(\mathsf{a}), \mathsf{g}(\mathsf{a})) \quad , \quad x \quad , \quad \mathsf{h}(\mathsf{f}(\mathsf{g}(\mathsf{a}), \mathsf{a})) \quad , \quad \ldots$$

The set of positions of a term t is a sequence of natural numbers and is defined recursively by

$$\mathsf{Pos}(t) := \begin{cases} \{\epsilon\} & \text{if } t \in \mathcal{V} \\ \{\epsilon\} \cup \{i \cdot p \mid p \in \mathsf{Pos}(t_i) \text{ for } 1 \leqslant i \leqslant k\} & \text{if } t = f(t_1, \ldots, t_k) \end{cases} \quad .$$

Here, $\cdot$ denotes concatenation of sequences. The size of a term, denoted by $|t|$, is defined as $|\mathsf{Pos}(t)|$. The sub-term at position p of a term t is defined as

$$t|_p := \begin{cases} t & \text{if } p = \epsilon \\ t_i|_q & \text{if } t = f(t_1, \ldots, t_k) \text{ and } p = i \cdot q \end{cases} \quad .$$

The set of sub-terms of a term t is defined as $\{t|_p \mid p \in \mathsf{Pos}(t)\}$. The root of a term t, $\mathsf{rt}(t)$, is the symbol at position ϵ.

Definition 2.10. Let ℓ and r be terms. A *rewrite rule* is a pair $\ell \to r$ such that

1. $\ell \notin \mathcal{V}$, and
2. $\mathcal{V}\mathsf{ar}(r) \subseteq \mathcal{V}\mathsf{ar}(\ell)$.

We call ℓ the left-hand side (lhs) and r the right-hand side (rhs) of the rule. A *term rewrite system (TRS)* $\mathcal{R}$ is a set of rewrite rules.

A term t is *linear* if every variable $x \in \mathcal{V}$ occurs at most once in t. A TRS is *left-linear* (*right-linear*), if the lhs (rhs) of every rule is linear.

To apply a rewrite rule to a term we need the concept of substitution and context. A *substitution* is a mapping $\sigma : \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V})$ such that for finitely many x, $\sigma(x) \neq x$ holds. This extends to a term t in the obvious way and is written as $t\sigma$. Let $\square$ be a fresh constant symbol, i.e. $\square \notin \mathcal{F}$. A term $C \in \mathcal{T}(\mathcal{F} \cup \{\square\}, \mathcal{V})$ is a *context*, if $\square$ occurs exactly once in C. Let $C[t]_p$ be the term obtained by replacing $\square$ at position p in C by term t.

Definition 2.11. A term s *rewrites* to a term t, denoted as $s \to_{\mathcal{R}} t$, if there is a rule $\ell \to r \in \mathcal{R}$, a substitution σ, and a context C such that $s = C[\ell\sigma]_p$ and $t = C[r\sigma]_p$.

Example 2.12. Given a rule $\mathsf{f}(\mathsf{g}(x)) \to \mathsf{g}(x) \in \mathcal{R}$ and a term $\mathsf{h}(\mathsf{f}(\mathsf{g}(\mathsf{a}), \mathsf{a}))$. Then

$$\mathsf{h}(\mathsf{f}(\mathsf{g}(\mathsf{a}), \mathsf{a})) \to_{\mathcal{R}} \mathsf{h}(\mathsf{g}(\mathsf{a}), \mathsf{a}) \quad .$$

Here the substitution is $\sigma : x \mapsto \mathsf{a}$, and the context is $C = \mathsf{h}(\square, \mathsf{a})$.

Note that $\to_{\mathcal{R}}$ is a binary relation over $\mathcal{T}(\mathcal{F}, \mathcal{V})$. If $\to_{\mathcal{R}}$ is terminating, we say that $\mathcal{R}$ is *terminating.* We say a binary relation $\triangleright$ over terms is *closed under context*, if for all contexts C, $s \triangleright t$ implies $C[s] \triangleright C[t]$. Further, $\triangleright$ is *closed under substitution* if for all substitutions σ, $s \triangleright t$ implies $s\sigma \triangleright t\sigma$. By definition $\to_{\mathcal{R}}$ is closed under substitution and context.

Definition 2.13. A proper order $\succ$ is a *rewrite order* if $\succ$ is closed under context and substitution. A *reduction order* is a well-founded rewrite order.

Let $\succ$ be an reduction order on $\mathcal{T}(\mathcal{F}, \mathcal{V})$. We say $\succ$ is *compatible* with $\mathcal{R}$ if $s \succ t$ whenever $s \to_{\mathcal{R}} t$, i.e. $\to_{\mathcal{R}} \subseteq \succ$. We define a lexicographic path order on terms as e.g. in [13]. We call a proper order $>$ on $\mathcal{F}$ a *precedence.*

Definition 2.14. Let s and t be terms and $>$ a precedence on $\mathcal{F}$. Then $s >_{\mathsf{lpo}} t$ for $s = f(s_1, \dots, s_n)$ if one of the following holds

1. $s_i \geqslant_{\mathsf{lpo}} t$ for $1 \leqslant i \leqslant n$,
2. $t = g(t_1, \dots, t_m)$, $g < f$, and $s >_{\mathsf{lpo}} t_i$ for $1 \leqslant i \leqslant m$,
3. $t = f(t_1, \dots, t_n)$, and for some $1 \leqslant i \leqslant n$ for all $1 \leqslant j < i$, $s_j = t_j$, and $s_i >_{\mathsf{lpo}} t_i$, and $s >_{\mathsf{lpo}} t_k$ for all $i < k \leqslant n$.

Example 2.15. Given a precedence $\mathsf{f} > \mathsf{g} > \mathsf{a} > \mathsf{b}$. Then $\mathsf{f}(x, y) >_{\mathsf{lpo}} x$, $\mathsf{f}(x, y) >_{\mathsf{lpo}} \mathsf{g}(x)$, and $\mathsf{f}(\mathsf{a}, \mathsf{a}) >_{\mathsf{lpo}} \mathsf{f}(\mathsf{a}, \mathsf{b})$.

A TRS $\mathcal{R}$ is terminating if $\ell >_{\mathsf{lpo}} r$ for all $\ell \to r \in \mathcal{R}$. Put differently, if the rules in a rewrite system are compatible with $>_{\mathsf{lpo}}$, there are no infinite rewrite sequences. This statement can be proved by defining an embedding relation $\sqsubseteq_{\mathsf{emb}}$ on terms, and showing $\sqsupset_{\mathsf{emb}} \subseteq >_{\mathsf{lpo}}$. If $\sqsupset_{\mathsf{emb}} \subseteq >_{\mathsf{lpo}}$, $>_{\mathsf{lpo}}$ is a *simplification order* [21]. Well-foundedness of the $\sqsubseteq_{\mathsf{emb}}$, and all simplification orders, can be shown by Kruskal's Tree theorem [19]. We define embedding of terms as follows.

Definition 2.16. Let s, t be terms and $\leqslant$ a proper order on $\mathcal{F}$. A term t is embedded in a term s, denoted by $s \sqsupseteq_{\mathsf{emb}} t$, if for $s = f(s_1, \ldots, s_n)$ one of the following holds

1. $s_i \sqsupseteq_{\mathsf{emb}} t$ for $1 \leqslant i \leqslant m$, or
2. for $t = g(t_1, \ldots, t_m)$, $g \leqslant f$, we have $s_{i_1} \sqsupseteq_{\mathsf{emb}} t_1, s_{i_2} \sqsupseteq_{\mathsf{emb}} t_2, \cdots, s_{i_m} \sqsupseteq_{\mathsf{emb}} t_m$ where $1 \leqslant i_1 < i_2 < \cdots < i_m \leqslant n$.

With this we conclude the preliminaries. In the next chapter we move from the tree structure of terms to a graph structure and introduce term graphs.

3 Term Graph Rewriting

Term graph rewriting comes in many different flavours. Some differences are subtle. For example, nodes representing variables may have labels like x, y, and z, e.g. [5], or the label of any variable node is $\perp$, e.g. [7, 8]. Some differences are not so subtle. For example, some formalisms allow for cycles and hence infinite structures, e.g. [7, 8, 18] and some do not, e.g. [28, 2]. These different flavours are presented in Chapter 7.

In this thesis, we chose the term graph rewriting formalism defined in [2, 4, 5], which is very close to term rewriting. In fact, it is shown to be adequate to simulate term rewriting [2]—provided we have an appropriate mechanism to collapse and uncollapse sub-graphs. Thus it is quite intuitive from the background of term rewriting. Finally these works have clear and well developed definitions.

In Section 3.1 we introduce term graphs and operations on them. Then in Section 3.2 we define term graph rewriting. Finally in Section 3.3 we transfer from term graph rewriting to term rewriting and back again.

3.1 Term Graphs

The definition of term graphs is based on directed and ordered graphs.

Definition 3.1. Let $\mathcal{N}$ be a countable infinite set of nodes. A *directed and ordered graph* G over a set of *labels* $\mathcal{L}$ is a triple $G = (N_G, \mathsf{succ}_G, \mathsf{lab}_G)$, where $N_G \subseteq \mathcal{N}$ and N_G is finite, $\mathsf{succ}_G : N_G \to [N_G, \dots, N_G]$ is a mapping from a node to an ordered sequence of *successors*, and lab_G is a mapping $N_G \to \mathcal{L}$.

We call a directed and ordered graph simply graph. In the following, graphs will be denoted by G and H. Nodes in a graph will usually be n, or u and v, and we write $n \in G$ instead of $n \in N_G$. Subscripts are dropped where the context is clear. By default, $\mathcal{N} = \mathbb{N}$, but we may use roman numerals or (Greek) letters on occasion. Still we usually refer to elements of $\mathcal{N}$ as node numbers.

Example 3.2. In examples we will draw graphs as depicted below. Shown is the graph $G = (\{\textcircled{1}, \textcircled{2}\}, \mathsf{succ}_G, \mathsf{lab}_G)$, where $\mathsf{succ}_G : \textcircled{1} \mapsto [\textcircled{2}, \textcircled{2}], \textcircled{2} \mapsto [\,]$, and $\mathsf{lab} : \textcircled{1} \mapsto \mathsf{f}, \textcircled{2} \mapsto x$. As can be seen in the right graph, we omit node numbers when convenient:

$$G : \begin{tikzcd} \mathsf{f} : \textcircled{1} \arrow[d, bend right] \arrow[d, bend left] \\ x : \textcircled{2} \end{tikzcd} \quad \text{or simply} \quad \begin{tikzcd} \mathsf{f} \arrow[d, bend right] \arrow[d, bend left] \\ x \end{tikzcd} .$$

The next definition introduces some standard notions on graphs, i.e. size, successors, paths, roots, and acyclicicity.

Definition 3.3. Let $G = (N_G, \mathsf{succ}_G, \mathsf{lab}_G)$ be a graph.

- The *size* of G is defined, and denoted, as $|G| := |N_G|$.
- Suppose for some node $n \in G$ we have $\mathsf{succ}_G(n) = [n_1, \dots, n_i, \dots, n_k]$. Then n_i is the i^{th} successor of n denoted by $\mathsf{succ}^i_G(n) = n_i$, or, $n \xrightharpoonup{i}_G n_i$. We write $n \rightharpoonup_G n_i$, when $n \xrightharpoonup{i}_G n_i$ for some i. We also say there is an *edge* from n to n_i.
- For a non-empty sequence $n_1, \dots, n_{k+1} \in G$ with $n_1 \rightharpoonup_G \cdots \rightharpoonup_G n_{k+1}$ we have a *path* of length k. We also say, n_{k+1} is *reachable* from n_1.
- For $n_1 \rightharpoonup^*_S n_2$ we say that n_1 is *above* n_2, or, alternatively n_2 is *below* n_1. If $n_1 \rightharpoonup^+_S n_2$, then n_1 is *strictly above* of n_2, or n_2 is *strictly below* of n_1.
- Two nodes n_1, n_2 are *parallel* if neither $n_1 \rightharpoonup^*_G n_2$ nor $n_2 \rightharpoonup^*_G n_1$.
- The graph G is *rooted*, if there exists a unique node, the *root* of G, denoted by $\mathsf{rt}(G)$, such that for all nodes $n \in G$, $\mathsf{rt}(G) \rightharpoonup^* n$ holds.
- A graph G is *acyclic*, if $n_1 \rightharpoonup^+ n_2$ implies $n_1 \neq n_2$ for all nodes $n_1, n_2 \in G$.

Example 3.4. In Example 3.2 the size of G is $|G| = 2$, the successors are $① \xrightharpoonup{1} ②$ and $① \xrightharpoonup{2} ②$, and the root of G is $\mathsf{rt}(G) = ①$. Also, G is acyclic.

Next we will introduce some basic operations on directed and ordered graphs: the sub-graph operation, the union of two graphs, and the redirection of edges.

Definition 3.5. The *sub-graph* of a graph $G = (N_G, \mathsf{succ}_G, \mathsf{lab}_G)$ reachable from a node $n \in G$ is defined as $G' = (N_{G'}, \mathsf{succ}_{G'}, \mathsf{lab}_{G'})$, where $N_{G'} = \{n' \mid n \rightharpoonup^*_G n'\}$ and the domains of $\mathsf{succ}_{G'}$ and $\mathsf{lab}_{G'}$ are restricted to $N_{G'}$. We write $G' = G{\restriction}n$.

In the next definition we will consider the union of two graphs.

Definition 3.6. For two graphs G and H, their (left-biased) *union*, denoted by $G \oplus H$, is defined as $(N_G \cup N_H, \mathsf{succ}_G \oplus \mathsf{succ}_H, \mathsf{lab}_G \oplus \mathsf{lab}_H)$, where for $f \in \{\mathsf{succ}, \mathsf{lab}\}$ we define

$$f_G \oplus f_H(n) := \begin{cases} f_G(n) & \text{if } n \in G \\ f_H(n) & \text{if } n \notin G \text{ and } n \in H. \end{cases}$$

Note, that we do not require $N_G \cap N_H = \varnothing$. In particular for a node n where $n \in G$ and $n \in H$ we favour the graph G, hence the union is left-biased.

The next definition allows the redirection of all in-coming edges of a node to another node in the graph.

Definition 3.7. Let u, v be two distinct nodes in G. By $G[v \leftarrow u]$ we *redirect* edges pointing at u to v. That is, $G[v \leftarrow u] = (N_G, \mathsf{succ}_{G[v\leftarrow u]}, \mathsf{lab}_G)$, where for all nodes $n \in G$

$$\mathsf{succ}^i_{G[v\leftarrow u]}(n) := \begin{cases} v & \text{if } n = u \\ n & \text{otherwise.} \end{cases}$$

Note, that for $G[v \leftarrow u]$ we still have $u \in G$. The introduced definitions and operations so far work on directed and ordered graphs. There are no restrictions with respect to cycles and roots yet.

Definition 3.8. Let G be a directed, ordered, acyclic graph, i.e. a *dag*. Then G is a *term dag* over a signature $\mathcal{F}$ and variables $\mathcal{V}$ when

- the set of labels $\mathcal{L}$ is $\mathcal{F} \cup \mathcal{V}$,
- for a node $n \in G$ with $\mathsf{lab}_G(n) = f \in \mathcal{F}$ and $\mathsf{ar}(f) = k$ exist $n_1, \dots, n_k \in G$ such that $\mathsf{succ}_G(n) = [n_1, \dots, n_k]$,
- for every $n \in G$ with $\mathsf{lab}_G(n) \in \mathcal{V}$, $\mathsf{succ}_G(n) = [\,]$ holds.

A term dag G is a *term graph* if it is rooted. We collect all term graphs over $\mathcal{F}$ and $\mathcal{V}$ in $\mathcal{TG}(\mathcal{F}, \mathcal{V})$.

Example 3.9. The graph in Example 3.2 is a term graph.

In [4, 5] it is additionally required that all variable nodes with the same label are represented by the same node. Formally that is, for all $n_1 \in G$ with $\mathsf{lab}_G(n_1) = x \in \mathcal{V}$, if $\mathsf{lab}_G(n_2) = x$ then $n_1 = n_2$. We drop this requirement here, as in the later [2]—as distinguishing between nodes representing variables and nodes representing function symbols is not necessary for a term graph. It will be important for graph rewrite rules though.

The set of variable nodes in a term dag G is denoted by $\mathcal{V}\mathsf{ar}(G) \subseteq N_G$. Let $f \in \mathcal{F} \cup \mathcal{V}$. If for a node n, $\mathsf{lab}(n) = f$, then n is called a *f-node*. In the following S and T denote term graphs.

Finally we consider the essential benefit of the graph representation: The possibility to share equal sub-graphs by identifying them by a single node.

Example 3.10. The three term graphs below employ different degrees of sharing.

$$\begin{tikzcd}[column sep=small] & \mathsf{f} \arrow[dl] \arrow[dr] & \\ \mathsf{g} \arrow[d] & & \mathsf{g} \arrow[d] \\ \mathsf{a} & & \mathsf{a} \end{tikzcd} \quad \text{shares fewer nodes then} \quad \begin{tikzcd}[column sep=small] & \mathsf{f} \arrow[dl] \arrow[dr] & \\ \mathsf{g} \arrow[dr] & & \mathsf{g} \arrow[dl] \\ & \mathsf{a} & \end{tikzcd} \quad \text{shares fewer nodes then} \quad \begin{tikzcd} \mathsf{f} \arrow[d, bend right] \arrow[d, bend left] \\ \mathsf{g} \arrow[d] \\ \mathsf{a} \end{tikzcd} .$$

We define shared nodes based on the notion of *positions*.

Definition 3.11. The set of positions of a node $u \in S$ is defined as follows:

$$\mathsf{Pos}_S(u) := \begin{cases} \{\epsilon\} & \text{if } u = \mathsf{rt}(S) \\ \{p \cdot i \mid \exists v \in S \text{ with } v \xrightarrow{i}_S u \text{ and } p \in \mathsf{Pos}_S(v)\} & \text{otherwise.} \end{cases}$$

Then we can determine for a node if it is shared or unshared by the set of its positions.

Definition 3.12. A node $n \in S$ is *shared*, if $\mathsf{Pos}_S(n)$ is not a singleton. Otherwise n is *unshared*.

Positions are also interesting because they give rise to a *canonical* representation of term graphs. For a canonical representation each node in the term graph is referenced by the set of its positions.

Definition 3.13. For a term graph T its *canonical representation* is $C = (N_C, \mathsf{succ}_C, \mathsf{lab}_C)$, where $N_C = \{\mathsf{Pos}_T(n) \mid n \in T\}$, $\mathsf{lab}_C(\mathsf{Pos}_T(n)) = \mathsf{lab}_T(n)$, and for $\mathsf{succ}_C(n) = [n_1, \ldots, n_k]$ we have $\mathsf{succ}_C(\mathsf{Pos}_T(n)) = [\mathsf{Pos}(n_1), \ldots, \mathsf{Pos}(n_k)]$.

Example 3.14. Recall the graphs from Example 3.10. Their canonical representations are shown below:

$$
\begin{tikzcd}[column sep=tiny]
& \mathsf{f} : \{\epsilon\} \arrow[dl] \arrow[dr] & \\
\mathsf{g} : \{1\} \arrow[d] & & \mathsf{g} : \{2\} \arrow[d] \\
\mathsf{a} : \{1 \cdot 1\} & & \mathsf{a} : \{2 \cdot 1\}
\end{tikzcd}
\quad , \quad
\begin{tikzcd}[column sep=tiny]
& \mathsf{f} : \{\epsilon\} \arrow[dl] \arrow[dr] & \\
\mathsf{g} : \{1\} \arrow[dr] & & \mathsf{g} : \{2\} \arrow[dl] \\
& \mathsf{a} : \{1 \cdot 1, 2 \cdot 1\} &
\end{tikzcd}
\quad , \quad
\begin{tikzcd}
\mathsf{f} : \{\epsilon\} \arrow[d, bend right] \arrow[d, bend left] \\
\mathsf{g} : \{1, 2\} \arrow[d] \\
\mathsf{a} : \{1 \cdot 1, 2 \cdot 1\}
\end{tikzcd}
\quad .
$$

We next demonstrate a way to compute nodes, which can be shared. To find a common structure between two graphs consider the following definition of *morphism*.

Definition 3.15. Let S and T be term graphs, and $\Delta \subseteq \mathcal{F} \cup \mathcal{V}$. A function $m : S \to T$ is *morphic* if for a node $n \in S$

1. $\mathsf{lab}_S(n) = \mathsf{lab}_T(m(n))$, and
2. if $n \xrightarrow{i}_S n_i$ then $m(n) \xrightarrow{i}_T m(n_i)$ for all appropriate i.

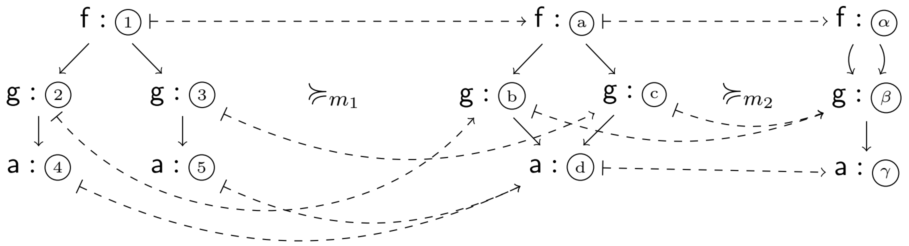
A Δ-*morphism* from S to T is a mapping $m : S \to_\Delta T$, which is morphic in all nodes $n \in S$ with $\mathsf{lab}(n) \notin \Delta$, where additionally $m(\mathsf{rt}(S)) = \mathsf{rt}(T)$ holds.

This definition of Δ-morphism allows to detect sub-graphs that can be shared. Later on a morphism will also indicate, when a rule matches. But first we consider the definition of *collapsing*. As an intuition: we set the set Δ to $\varnothing$ and thereby capture all nodes.

Definition 3.16. Let S and T be term graphs with a morphism $m : S \to_\varnothing T$. Then S *collapses* to T denoted by $S \succcurlyeq_m T$. If $S \succcurlyeq_m T$ and $T \succcurlyeq_m S$, then S is *isomorphic* to T, denoted by $S \cong_m T$. If $S \succcurlyeq_m T$ and $S \ncong_m T$, the relation is *strict* and we write $S \succ_m T$.

We usually drop m if it is clear from context. That is we write $S \succcurlyeq T$ if $S \succcurlyeq_m T$ for some $m : S \to_\varnothing T$. In the remainder of this thesis we distinguish between collapsing and shared nodes and sub-graphs. We talk of collapsing when we mean an explicit operation, which results in a term graph that may contain more shared nodes. Shared nodes and sub-graphs describe a state or a property.

Example 3.17. Recall the term graphs in Example 3.10. Below we can see the underlying morphisms m_1 and m_2 indicated by dashed lines.



.

For m_1 we have ① ↦ ⓐ, ② ↦ ⓑ, ③ ↦ ⓒ, ④ ↦ ⓓ, and ⑤ ↦ ⓓ. For m_2 we have ⓐ ↦ (α), ⓑ ↦ (β), ⓒ ↦ (β), and ⓓ ↦ (γ).

Our notion of Δ-morphism is similar to the concept of *bisimilar graphs*. Two graphs are bisimilar, if they are indistinguishable with respect to their symbol structure. That is, each path from the root in one graph corresponds to a path in the other graph with the same sequence of labels. Ariola et al [1] and Barendsen [8] give the formal definition:

Definition 3.18. Given two term graphs S and T with $n_S \in S$ and $n_T \in T$, and a relation $\rhd \subseteq N_S \times N_T$. Then S and T are *bisimilar* if

- $n_S \rhd n_T$,
- $n_S \rhd n_T$ implies $\mathsf{lab}(n_S) = \mathsf{lab}(n_T)$ and for appropriate i, $\mathsf{succ}^i(n_S) \rhd \mathsf{succ}^i(n_T)$.

If $\rhd$ is a function, then $\rhd$ is a homomorphism.

Comparing with our definition of collapsing and isomorphism in Chapter 3 both relations are a bisimulation for ground term graphs.

We investigate collapsing a bit further and show that $\succ$ defines a proper order and $\cong$ an equivalence on $\mathcal{TG}(\mathcal{F}, \mathcal{V})$.

Lemma 3.19. *The relation $\succcurlyeq$ is a pre-order on $\mathcal{TG}(\mathcal{F}, \mathcal{V})$.*

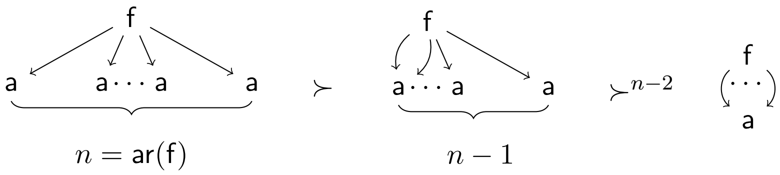
Proof. We follow the proof in [2, Lemma 4.16]. By the identity morphism $S \rightarrow_{\varnothing} S$, $\succcurlyeq$ is reflexive. For transitivity assume $S \succcurlyeq_{m_1} T$ and $T \succcurlyeq_{m_2} U$, we then set $m = m_1 \circ m_2$ and observe that both conditions in Definition 3.15 hold, showing $S \succcurlyeq_m U$. ☐

If all equal sub-terms are represented by the same node, we call a term graph *maximally shared*. Maximally shared term graphs are minimal in the order $\succ$ and are the most space efficient representation.

Next we investigate the influence of collapsing on size, and find an upper bound on the potential collapsing steps. Hereby, the number of *strict* collapsing steps is bounded by the size of the graph.

Lemma 3.20. *$S \succ T$ implies $|S| > |T|$ and $\mathsf{max}\{k \mid S \succ^k T\} \leqslant |S| - 1$.*

Proof. The first statement follows from the definition of $\succ$. We prove the second statement by induction on the size of S with the first statement. The intuition is:



□

So far we presented graphs and term graphs with different operations on them—most notably union, redirection, and finding a morphism which gives rise to collapsing. Next we want to use these operations to rewrite term graphs.

3.2 Term Graph Rewriting

To introduce term graph rewriting we must first consider graph rewrite rules and systems.

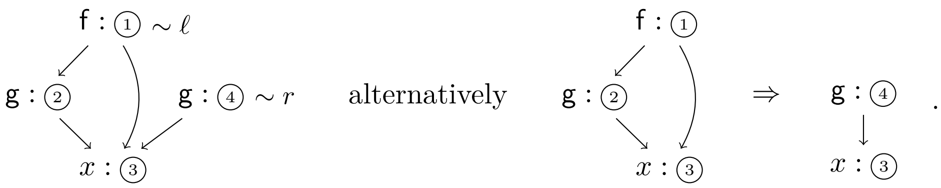
Definition 3.21. A *graph rewrite rule* is a triple (G, ℓ, r), where G is a graph, ℓ is the root node of the left hand side (lhs), and r is the root node of right hand side (rhs). That is $G\restriction\ell = L \in \mathcal{TG}(\mathcal{F}, \mathcal{V})$ and $G\restriction r = R \in \mathcal{TG}(\mathcal{F}, \mathcal{V})$. Additionally the following holds:

1. $\mathsf{lab}(\ell) \notin \mathcal{V}$, and
2. $\mathcal{V}\mathsf{ar}(R) \subseteq \mathcal{V}\mathsf{ar}(L)$, and
3. for all nodes $n, m \in G$, if $\mathsf{lab}(m) = \mathsf{lab}(n) \in \mathcal{V}$ then $n = m$.

A *graph rewrite system* (GRS) $\mathcal{G}$ is a set of graph rewrite rules.

Restriction 1 avoids a rule which matches every term graph, restriction 2 avoids free variables on the rhs, and restriction 3 ensures, that all occurrences of a variable are mapped to the same sub-graph. When all occurrences of a variable are represented by a single node, we call a graph *variable sharing*.

Example 3.22. The following picture shows a graph rewrite rule. On the left is a single graph as described in Definition 3.21 with distinguished root nodes ℓ and r. On the right is a graph rewrite rule with separated graphs for the left and the right hand side. The connection between the two sides is indicated through identical node numbers, here ③.

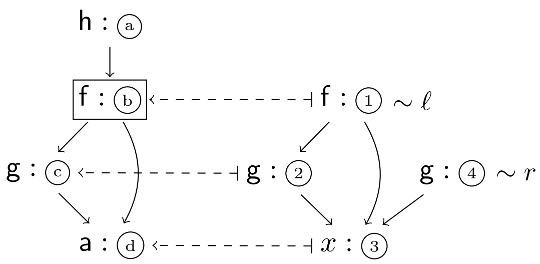


To apply a graph rewrite rule to a term graph S we need to *match* the lhs with some sub-graph of S. As mentioned before, we employ a morphism to find this matching.

Definition 3.23. Let $L \Rightarrow R$ be a graph rewrite rule and S be a term graph, where $S \cap R = \varnothing$. The term graph L *matches* the term graph S at *redex node* n, if there is a morphism $m : L \to_{\mathcal{V}} S\restriction n$.

The morphism suspends the condition for equal labels on variable nodes. The condition $S \cap R = \varnothing$ is required as to not accidentally get duplicate node numbers later on.

Example 3.24. We match a term graph with the lhs of the graph rewrite rule from Example 3.22. Therefore we find the morphism from redex node (b) with the (1) $\mapsto$ (b), (2) $\mapsto$ (c), and (3) $\mapsto$ (d). We indicate the redex node by a frame around it.



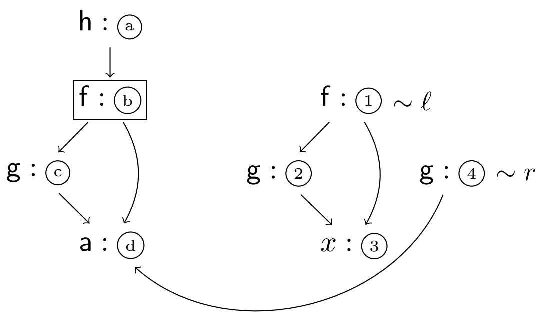
For defining a graph rewrite step we need to *apply* this graph morphism.

Definition 3.25. Let $m : L \to S\restriction n$ for a rule $L \Rightarrow R$. The morphism m is *applied* to R, denoted by $m(R)$, by redirecting all variable nodes in R to their image. That is, for all variable nodes $n_1, \dots, n_k \in \mathcal{V}\mathsf{ar}(R)$ we define

$$m(R) = ((R \oplus S)[m(n_1) \leftarrow n_1]) \dots [m(n_k) \leftarrow n_k].$$

Definition 3.21 requires that all variables are represented by the same node. Thus we know that every node in $\mathcal{V}\mathsf{ar}(R)$ is in the domain of m.

Example 3.26. Here we apply the morphism computed in Example 3.24, i.e. (d) $\leftarrow$ (4):

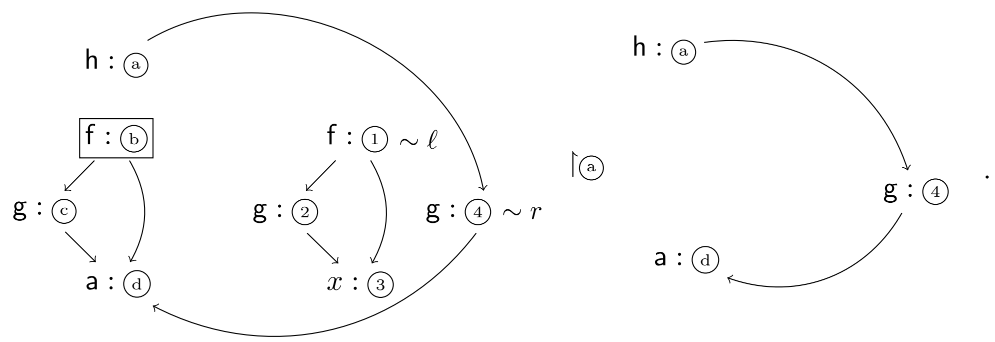


We apply the graph rewrite rule in a context, i.e. the nodes above the redex node n. Note that n may be referenced by several edges.

Definition 3.27. Let S and T be term graphs, $n \in S$, and $N_S \cap N_T = \varnothing$. The replacement of the sub-graph $S{\restriction}n$ by T is defined as follows:

$$S[T]_n := \begin{cases} T & \text{if } n = \mathsf{rt}(S) \\ (S \oplus T)[\mathsf{rt}(T) \leftarrow n]{\restriction}\mathsf{rt}(S) & \text{otherwise.} \end{cases}$$

Example 3.28. Continuing with Example 3.26 first we redirect the incoming edge of node b to node 4. Then we collect all reachable nodes from the root a.

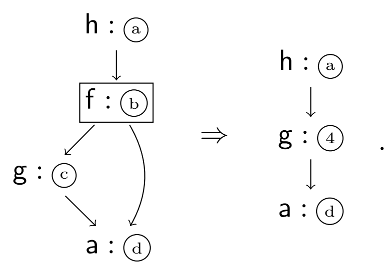


Finally we can define a term graph rewrite step.

Definition 3.29. Let $\mathcal{G}$ be a GRS. A term graph S *rewrites* to a term graph T, denoted by $S \Rightarrow_{L \Rightarrow R,n} T$, if there is a graph rewrite rule $L \Rightarrow R \in \mathcal{G}$ with $N_R \cap N_S = \varnothing$ and a morphism $m : L \to S{\restriction}n$ such that $S[m(R)]_n = T$.

We sometimes write $S \Rightarrow_{\mathcal{G}} T$, if $S \Rightarrow_{L \Rightarrow R,n} T$ and $L \Rightarrow R \in \mathcal{G}$.

Example 3.30. Combining the graph rewrite rule from Example 3.22, the morphism from Example 3.24 starting from redex node b, and the application from Example 3.26 with the redirection from Example 3.28, we get the following graph rewrite step:



We briefly touch upon the consequences of not restricting the sharing within graph rewrite rules. A detailed investigation is out of scope for this work. Definition 3.21 only requires variable nodes to be shared. Apart from variable sharing, no restriction is placed upon how L and R are shared. This is contrary to [2, 4, 5], where the graph rewrite rules share only variable nodes.

Sharing within L seems of little benefit and interest. To find a morphism it is best if a lhs shares only variable nodes. Shared rhs's may induce a loss of reachable term graphs. Consider first the GRS $\mathcal{G}$ below, which shares only variables.

Example 3.31. Consider the GRS $\mathcal{G}$, which shares only variable nodes:

$$\begin{tikzcd}[row sep=small, column sep=small]
 & & & \mathsf{h} \arrow[dl] \arrow[dr] & & & & & \\
\mathsf{f} \arrow[d] & \Rightarrow & \mathsf{g} \arrow[dr] & & \mathsf{g} \arrow[dl] & , & \mathsf{g} \arrow[d] & \Rightarrow & x \quad . \\
x & & & x & & & x & &
\end{tikzcd}$$

Then $\mathcal{G}$ admits the rewrite sequence, where no form of collapsing is employed:

$$\begin{tikzcd}[row sep=small, column sep=small]
 & & & \mathsf{h} \arrow[dl] \arrow[dr] & & & & \boxed{\mathsf{h}} \arrow[dl] \arrow[dr] & \\
\boxed{\mathsf{f}} \arrow[d] & \Rightarrow_{\mathcal{G}} & \boxed{\mathsf{g}} \arrow[dr] & & \mathsf{g} \arrow[dl] & \Rightarrow_{\mathcal{G}} & \mathsf{g} \arrow[d] & & \mathsf{a} \quad . \\
\mathsf{a} & & & \mathsf{a} & & & \mathsf{a} & &
\end{tikzcd}$$

If $\mathcal{G}$ is changed so that the rhs's of the rules share maximally, this rewrite sequence is not possible any more. Consider first the modified rewrite system $\mathcal{G}_{\succ!}$, where the first rule is maximally shared:

$$\begin{tikzcd}[row sep=small, column sep=small]
 & & \mathsf{h} \arrow[d, bend right] \arrow[d, bend left] & & & & \\
\mathsf{f} \arrow[d] & \Rightarrow & \mathsf{g} \arrow[d] & , & \mathsf{g} \arrow[d] & \Rightarrow & x \quad . \\
x & & x & & x & &
\end{tikzcd}$$

Then the above rewrite sequence is not possible any more:

$$\begin{tikzcd}[row sep=small, column sep=small]
 & & \mathsf{h} \arrow[d, bend right] \arrow[d, bend left] & & \mathsf{h} \arrow[d, bend right] \arrow[d, bend left] \\
\boxed{\mathsf{f}} \arrow[d] & \Rightarrow_{\mathcal{G}_{\succ!}} & \boxed{\mathsf{g}} \arrow[d] & \Rightarrow_{\mathcal{G}_{\succ!}} & \mathsf{a} \quad . \\
\mathsf{a} & & \mathsf{a} & &
\end{tikzcd}$$

Throughout the remainder of this thesis we assume our graph rewrite systems to be variable sharing—but only sharing variable nodes.

In the remaining part of this section we investigate how collapsing and sharing influences the potential rewrite steps.

Lemma 3.32. *If $S \Rightarrow_{L \Rightarrow R, n} T$ for $L \Rightarrow R \in \mathcal{G}$ and $S \succcurlyeq S'$, then $S' \Rightarrow_{L \Rightarrow R, n'} T'$.*

Proof. By the first assumption we have a morphism $m_1 : L \to S{\upharpoonright}n$ and $S[m_1(R)]_n = T$. By the second assumption we have a morphism $m_2 : S \to S'$. Combining them we get a morphism $m_3 : L \to S'{\upharpoonright}m_2(n)$ with $S[m_1(R)]_n \succcurlyeq_{m_2} S'[m_3(R)]_{m_2(n)}$. □

We conclude the section with one of the greatest advantages of the graph representation: the bound on size growth with each rewrite step from [2] Lemma 6.1.

Lemma 3.33. *If $S \Rightarrow T$ then $|T| \leqslant |S| + C$ where $C = \mathsf{max}\{|R| \mid L \Rightarrow R \in \mathcal{G}\}$.*

Proof. For $S \Rightarrow T$, we know that every node $n \in T$ occurs either in S or R, hence $|T| \leqslant |S| + |R|$ and $|R| \leqslant |C|$. □

In the last section of this chapter we consider the translations from terms to term graphs and vice versa.

3.3 From Terms to Term Graphs and Back Again

To investigate the relationship between term and term graph rewriting in the next Chapter 4 we need to transfer from term graphs to terms and vice versa. We start by computing a term from a term graph.

Definition 3.34. Let T be a term graph. The mapping $\mathsf{term} : \mathcal{TG}(\mathcal{F}, \mathcal{V}) \to \mathcal{T}(\mathcal{F}, \mathcal{V})$ computes the term representation of T and is defined as

$$\mathsf{term}(T) := \begin{cases} x & \text{if } \mathsf{lab}(\mathsf{rt}(T)) = x \in \mathcal{V} \\ f(\mathsf{term}(T{\upharpoonright}n_1), \dots, \mathsf{term}(T{\upharpoonright}n_k)) & \text{if } \mathsf{lab}(\mathsf{rt}(T)) = f \in \mathcal{F} \\ & \text{and } \mathsf{succ}(n) = [n_1, \dots, n_k]. \end{cases}$$

Note, that term is surjective but not injective in general. This is due to two reasons: Reason (1) is the set $\mathcal{N}$ may be different, or the individual nodes may be organised differently. Here injectivity can be restored with a canonical representation of the term graph as presented in Definition 3.13. Reason (2) are shared nodes. An alternative to Definition 3.12 is given next.

Definition 3.35. A node n in a term graph graph S is *shared*, if $S{\upharpoonright}n$ represents more than one sub-term in $\mathsf{term}(S)$.

Now we define the opposite and compute term graphs from terms. We hereby make use of a canonical representation.

Definition 3.36. Let t be a term. The function $\mathsf{tree}^{\mathcal{T}} : \mathcal{T}(\mathcal{F}, \mathcal{V}) \to \mathcal{TG}(\mathcal{F}, \mathcal{V})$ computes the canonical tree representation of t—a term graph $T = (N_T, \mathsf{lab}_T, \mathsf{succ}_T)$, where

- $N_T = \mathsf{Pos}(t)$
- for all $n \in N_T$, $\mathsf{lab}_T(n) = \mathsf{rt}(t|_n)$ and $\mathsf{succ}_T(n) = \{n \cdot i \mid i \in \{1, \dots, \mathsf{ar}(\mathsf{lab}_T(n))\}\}$.

Note, that $\mathsf{tree}^{\mathcal{T}}$ is injective, but not surjective. In particular, $\mathsf{term} = \mathsf{tree}^{\mathcal{T}^{-1}}$. We use the function $\mathsf{tree}^{\mathcal{T}}$ to define the $\mathsf{tree}^{\mathcal{G}}$ of a term graph—which is its tree representation of a term graph and maximal with respect to $\succcurlyeq$.

Definition 3.37. For a term graph S, let $\mathsf{tree}^{\mathcal{G}}(S) := \mathsf{tree}^{\mathcal{T}}(\mathsf{term}(S))$.

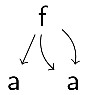
An explicit collapsing step does not change a term graph's term representation.

Lemma 3.38. *If $S \succcurlyeq T$ then $\mathsf{term}(S) = \mathsf{term}(T)$.*

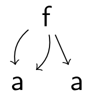
Proof. The proof in [2, Lemma 4.17] is by structural induction on S. Alternatively, we can construct $\mathsf{tree}^{\mathcal{G}}(S)$ and $\mathsf{tree}^{\mathcal{G}}(T)$, and obtain $m_1 : \mathsf{tree}^{\mathcal{G}}(S) \to S$ and $m_2 : \mathsf{tree}^{\mathcal{G}}(T) \to T$. From $S \succcurlyeq T$ we obtain $m_3 : S \to T$ and can define $m_4 := m_3 \circ m_1$. Hence we have $\mathsf{tree}^{\mathcal{G}}(S) \succcurlyeq T \preccurlyeq \mathsf{tree}^{\mathcal{G}}(T)$, where $\mathsf{tree}^{\mathcal{G}}(T)$ and $\mathsf{tree}^{\mathcal{G}}(S)$ are maximal wrt. $\succcurlyeq$ and hence $\mathsf{tree}^{\mathcal{G}}(T) \cong \mathsf{tree}^{\mathcal{G}}(S)$. By definition of $\mathsf{tree}^{\mathcal{G}}$ we have to show $\mathsf{tree}^{\mathcal{T}}(\mathsf{term}(T)) \cong \mathsf{tree}^{\mathcal{T}}(\mathsf{term}(S))$ implies $\mathsf{term}(S) = \mathsf{term}(T)$. It suffices to show that $\mathsf{tree}^{\mathcal{T}}(s) \cong \mathsf{tree}^{\mathcal{T}}(t)$ implies $s = t$, which is easy to prove by, e.g. contra-position. $\square$

The reverse of Lemma 3.38 does not hold, in particular $\mathsf{term}(S) = \mathsf{term}(T)$ does imply neither $S \succcurlyeq T$ nor $T \preccurlyeq S$. To emphasize this consider the following example.

Example 3.39. For the term graphs S and T below we have $\mathsf{term}(S) = \mathsf{term}(T)$, but they are incomparable:



$\not\succcurlyeq$ and $\not\preccurlyeq$



.

The translation between terms and term graphs naturally extends to rewrite systems.

Definition 3.40. We write $\mathcal{R}(\mathcal{G})$ for the TRS constructed from a GRS $\mathcal{G}$, i.e. $\mathcal{R}(\mathcal{G}) := \{\mathsf{term}(L) \to \mathsf{term}(R) \mid L \Rightarrow R \in \mathcal{G}\}$.

Conversely, we construct a graph rewrite system from a term rewrite system.

Definition 3.41. We write $\mathcal{G}(\mathcal{R})$ for a GRS constructed from TRS $\mathcal{R}$. For every $\ell \to r \in \mathcal{R}$ we compute $\mathsf{tree}^{\mathcal{T}}(\ell) = L$ and $\mathsf{tree}^{\mathcal{T}}(r) = R$. Then we compute $L \succcurlyeq L'$ and $R \succcurlyeq R'$ such that all, but only, variable nodes are shared, i.e. $N_{L'} \cap N_{R'} = \mathcal{V}\mathsf{ar}(L') \cup \mathcal{V}\mathsf{ar}(R')$. A graph rewrite rule is then $(L' \oplus R', \mathsf{rt}(L'), \mathsf{rt}(R'))$. We collect in $\mathcal{G}(\mathcal{R})$ all rules created from $\mathcal{R}$.

GRSs created in that way are by construction unique up to isomorphism. They are variable sharing, but do not share any other node.

This chapter introduced the underlying formalism for the remainder of this thesis. We continue now with an investigation of the potential combinations of collapsing and term graph rewriting.

4 Collapsing and Rewriting

The ability to share equal sub-graphs is the essence of term graph rewriting. But how do sharing and collapsing influence the potential graph rewrite steps? To answer this question we investigate term graph rewriting with respect to term rewriting.

We start with a crucial result: every term graph rewrite step can be simulated by one or more term rewrite steps.

Lemma 4.1. *If* $S \Rightarrow_{L \Rightarrow R, n} T$ *for* $L \Rightarrow R \in \mathcal{G}$*, then* $\mathsf{term}(S) \rightarrow^k_{\mathcal{R}(\mathcal{G})} \mathsf{term}(T)$*, where* $k = |\mathsf{Pos}(n)|$*.*

This lemma can be shown by induction over the nodes in S. In e.g. [17] the authors employ a case analysis on whether n is the root node and then apply the induction hypothesis k times, i.e. the number of paths from the root to n. Different to our setting, term graphs in [17] are hyper graphs (cf. Chapter 7). A proof in our setting of term graph rewriting in [2] relies on the definition of *multi-hole contexts* in terms, which are simultaneously replaced.

Note that Lemma 4.1 does not rely on any particular form of how nodes are shared in S and T nor does it incorporate any collapsing. Hence we stress: *any* graph rewrite step, independent of how the nodes in the term graphs are shared, can be simulated by one or more term rewrite steps. This naturally brings up the question: Can we simulate term rewriting by term graph rewriting?

4.1 Adequacy

We want to know whether term graph rewriting is *adequate* for term rewriting. This question was answered by Kennaway et al [18] and refined by Avanzini [2], as well as Plump [29, 30] who investigates soundness and completeness.

Different notions of adequacy exist in literature. Informally speaking we say that a graph rewrite relation $\triangleright_{\mathcal{G}} \subseteq \mathcal{G}(\mathcal{F},\mathcal{V}) \times \mathcal{G}(\mathcal{F},\mathcal{V})$ is adequate for a term rewrite relation $\triangleright_{\mathcal{R}} \subseteq \mathcal{T}(\mathcal{F},\mathcal{V}) \times \mathcal{T}(\mathcal{F},\mathcal{V})$ if we can simulate $\triangleright_{\mathcal{R}}$ by $\triangleright_{\mathcal{G}}$ and $\triangleright_{\mathcal{G}}$ by $\triangleright_{\mathcal{R}}$. In [18] adequacy is defined along four dimensions:

1. *surjectivity*, i.e. for every term t there exists a term graph T such that $\mathsf{term}(T) = t$,
2. *closure under reduction*, i.e. if $S \triangleright_{\mathcal{G}} T$ then $S \in \mathcal{G}(\mathcal{F},\mathcal{V})$ and $T \in \mathcal{G}(\mathcal{F},\mathcal{V})$.
3. *preservation of reduction*, i.e. if $S \triangleright_{\mathcal{G}} T$ then $\mathsf{term}(S) \triangleright_{\mathcal{R}} \mathsf{term}(T)$.
4. *simulation of reduction*, i.e. if $\mathsf{term}(S) \triangleright_{\mathcal{R}} t$ then $S \triangleright_{\mathcal{G}} T$ where $\mathsf{term}(T) = t$.

The main difference between [18] and [2] is that [18] works on cyclic term graphs where $\rhd_{\mathcal{R}}$ is a restricted form of infinitary term rewriting. Furthermore [18] only considers left-linear rules, which makes collapsing not a requirement. On the other hand [2] seeks to precisely relate steps, and thus formulates Condition 3 and 4 for single steps. Therefore [2] needs an explicit collapsing and uncollapsing relation.

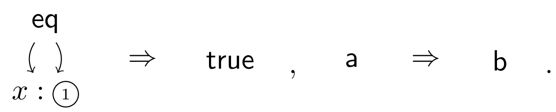
From Lemma 4.1 we know that for a graph rewrite step $S \Rightarrow_{\mathcal{G}} T$ we can find corresponding term rewrite steps $\mathsf{term}(S) \to^{+}_{\mathcal{R}(\mathcal{G})} \mathsf{term}(T)$. This is also referred to as *soundness*. On the other hand *completeness* is formulated as follows and proved in [29, 30]. In the proof the combination of $\Rightarrow$ and $\succ$ by $\cup$ is necessary to allow the base case of the induction.

Lemma 4.2. *Let $\mathcal{R}$ be a TRS and $\mathcal{G}(\mathcal{R})$ the corresponding GRS. For all term graphs S and T,*

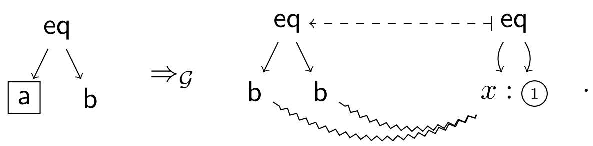
$$\mathsf{term}(S)\ (\to \cup \leftarrow)^*\ \mathsf{term}(T)\ \textit{iff}\ S\ (\Rightarrow \cup \succ \cup \Leftarrow \cup \prec)^*\ T \quad .$$

In course of this work we only give an intuition why collapsing and uncollapsing are needed. An explicit collapsing relation is needed, because in term graph rewriting equality is expressed through equal nodes. So even if two sub-graphs are the "same", i.e. isomorphic, this "same-ness" is not reflected if these sub-graphs are not represented by the same node. To remedy this we need collapsing. To clarify—an example:

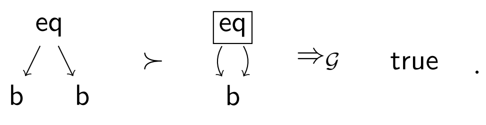
Example 4.3. The following unique representation of GRS $\mathcal{G}$ checks whether two arguments are equal—by checking whether they are represented by the same node:



Then $\mathcal{G}$ admits the following derivation starting from the unique graph representation of $\mathsf{eq}(\mathsf{a}, \mathsf{b})$:



After rewriting a to b, the eq-rule is not applicable because there is no morphism from the lhs to the term graph. As indicated by the zig-zag line the node ① cannot be mapped to two different nodes, even if they are the represent the same term. To make the eq-rule applicable we need an explicit collapse step:



In the lhs of a rule all nodes representing the same variable have to be shared by definition. Thus collapsing is crucial for for non-left linear systems. Indeed we can simulate term rewriting with term graph rewriting with only uncollapsing if $\mathcal{R}$ is left-linear [2, p. 58].

But why is uncollapsing required? This is illustrated in the following Example 4.4 which also shows the reverse of Lemma 4.1 to not hold in general: not every term rewrite step can be simulated by term graph rewrite steps.

Example 4.4. Consider the TRS $\mathcal{R}$:

$$\mathsf{f}(x) \to \mathsf{g}(x, x) \quad , \quad \mathsf{a} \to \mathsf{b} \quad .$$

Then the following term rewriting sequence is possible:

$$\mathsf{f}(\mathsf{a}) \to_{\mathcal{R}} \mathsf{g}(\mathsf{a}, \mathsf{a}) \to_{\mathcal{R}} \mathsf{g}(\mathsf{a}, \mathsf{b}) \quad .$$

This derivation cannot be simulated with term graph rewriting. Consider first the unique representation of GRS $\mathcal{G}(\mathcal{R})$:

$$\begin{tikzcd} \mathsf{f} \arrow[d] \\ x \end{tikzcd} \Rightarrow \begin{tikzcd} \mathsf{g} \arrow[d, bend right] \arrow[d, bend left] \\ x \end{tikzcd} \quad , \quad \mathsf{a} \Rightarrow \mathsf{b} \quad .$$

The above derivation does not yield the same result:

$$\begin{tikzcd} \boxed{\mathsf{f}} \arrow[d] \\ \mathsf{a} \end{tikzcd} \Rightarrow_{\mathcal{G}(\mathcal{R})} \begin{tikzcd} \mathsf{g} \arrow[d, bend right] \arrow[d, bend left] \\ \mathsf{a} \end{tikzcd} \quad .$$

The step $\mathsf{g}(\mathsf{a}, \mathsf{a}) \to_{\mathcal{R}} \mathsf{g}(\mathsf{a}, \mathsf{b})$ cannot be simulated. This is due to the fact, that there is no inverse operation of collapsing. To simulate the step we need the following explicit uncollapsing step:

$$\begin{tikzcd} \mathsf{g} \arrow[d, bend right] \arrow[d, bend left] \\ \mathsf{a} \end{tikzcd} \prec \begin{tikzcd} & \mathsf{g} \arrow[dl] \arrow[dr] & \\ \mathsf{a} & & \boxed{\mathsf{a}} \end{tikzcd} \Rightarrow_{\mathcal{G}(\mathcal{R})} \begin{tikzcd} & \mathsf{g} \arrow[dl] \arrow[dr] & \\ \mathsf{a} & & \mathsf{b} \end{tikzcd} \quad .$$

Hence for adequacy we need to be able to collapse a term graph, but also the reverse: we need to be able to uncollapse. But—why do we insist on adequacy? What if we do not demand adequacy and treat term graphs as first-order citizens?

4.2 Combine Rewriting and Collapsing

The aim of this section is to investigate what happens if we disregard adequacy of term graph rewriting for term rewriting. As we have seen in the previous section, adequacy demands collapsing and the reverse operation: uncollapse.

We argue that uncollapsing is counter-intuitive for term graph rewriting. It is in contrast to the idea of sharing equal sub-graphs. Thus we argue to drop uncollapsing. However for collapsing we argue in a different direction. We want to reap the fruits of term graph rewriting, i.e. the explicit option of sharing equal sub-graphs. Indeed when defining embedding of term graphs in Chapter 5 and a termination order in Chapter 6 we take sharing into account.

So we aim to integrate collapsing with the rewrite relation. But how to combine the collapsing relation with the rewrite relation? We have different ways to combine the graph rewrite relation $\Rrightarrow$ with a collapsing relation: by concatenation $(\cdot)$ or by union $(\cup)$. For the collapsing relation we can choose strict collapsing $(\succ)$, collapsing with equality $(\succcurlyeq)$, or collapsing to normal form $(\succ^!)$. We can immediately discard some combinations based on obvious observations.

- $\Rrightarrow \cdot \succ$ and $\succ \cdot \Rrightarrow$ do not allow rewrite steps without collapsing, e.g. with the rule $\mathsf{a} \Rrightarrow \mathsf{b}$ we cannot rewrite a, because $\mathsf{b} \not\succ \mathsf{b}$.
- $\Rrightarrow \cup \succcurlyeq$ immediately introduces non-termination as witnessed by $T \succcurlyeq T \succcurlyeq T \cdots$
- $\Rrightarrow \cup \succ^!$ also introduces non-termination: $T \succ^! T \succ^! T \cdots$

The non-termination observed in the last item can be countered by eliminating reflexivity from $\succ^!$. We thus introduce a new relation $\succ^{!^+} := \succ \cdot \succ^!$ to avoid the encountered problem. This leaves several potential combinations, which we now want to investigate systematically in turn with respect to

1. inclusion, and
2. normal forms.

For the former, inclusion, our aim is to find out to what extend we can simulate one relation with the other and where and why we fail. We investigate the latter, normal forms, as they correspond to our notion of a result. These two aspects, inclusion and normal forms, are also important when comparing rewrite strategies [36].

We list the potential combinations and comparisons in Figure 4.1, where we refer to those lemmata which state facts about the relationship between the two relations. Roughly we can divide the combinations in three sections: Section 4.3 deals with different combinations of collapsing through concatenation, Section 4.4 with different combinations of collapsing through union, and Section 4.5 compares combinations through concatenation with combinations through union.

We start by stating and showing some general lemmata about normal forms, which will prove useful throughout our analysis. Recall that $\triangleright_1$ and $\triangleright_2$ are arbitrary binary relations, and $\triangleright_2^=$ denotes the reflexive closure of $\triangleright_2$.

Lemma 4.5. $\mathsf{NF}(\triangleright_1) \subseteq \mathsf{NF}(\triangleright_1 \cdot \triangleright_2)$.

Proof. We need to show that $a \in \mathsf{NF}(\triangleright_1)$ implies $a \in \mathsf{NF}(\triangleright_1 \cdot \triangleright_2)$. By contra-position we have to show $a \notin \mathsf{NF}(\triangleright_1 \cdot \triangleright_2)$ implies $a \notin \mathsf{NF}(\triangleright_1)$. Then by $a \notin \mathsf{NF}(\triangleright_1 \cdot \triangleright_2)$ we have $\exists b.\ a \triangleright_1 c \triangleright_2 b$. But then $\exists c.\ a \triangleright_1 c$, hence $a \notin \mathsf{NF}(\triangleright_1)$. $\square$

We inspect the reverse direction to discover:

Lemma 4.6. $\mathsf{NF}(\triangleright_1 \cdot \triangleright_2^=) \subseteq \mathsf{NF}(\triangleright_1)$.

$\Rightarrow \cdot \succcurlyeq / \succcurlyeq \cdot \Rightarrow$	$\Rightarrow \cdot \succ^! / \succ^! \cdot \Rightarrow$	$\Rightarrow \cup \succ$	$\Rightarrow \cup \succ^{!+}$	
Lem. 4.10 & 4.12 Section 4.3	Lem. 4.14 & 4.16 Section 4.3	Lemma 4.21 Section 4.4	Lemma 4.22 Section 4.4	$\Rightarrow$
	Lem. 4.19 & 4.20 Section 4.3	Lem. 4.25 & 4.28 Section 4.5	Lem. 4.30 & 4.31 Section 4.5	$\Rightarrow \cdot \succcurlyeq / \succcurlyeq \cdot \Rightarrow$
		Lem. 4.32 & 4.33 Section 4.5	Lem. 4.34 & 4.35 Section 4.5	$\Rightarrow \cdot \succ^! / \succ^! \cdot \Rightarrow$
			Lemma 4.23 Section 4.4	$\Rightarrow \cup \succ$

Figure 4.1: Combinations between the collapsing and the rewriting relation.

Proof. We need to show that $a \in \mathsf{NF}(\rhd_1 \cdot \rhd_2^{=})$ implies $a \in \mathsf{NF}(\rhd_1)$. By contra-position we have to show that $a \notin \mathsf{NF}(\rhd_1)$ implies $a \notin \mathsf{NF}(\rhd_1 \cdot \rhd_2^{=})$. Then by $a \notin \mathsf{NF}(\rhd_1)$ we have $\exists b.\ a \rhd_1 b$. By reflexivity of $\rhd_2^{=}$ we have $\exists b.\ a \rhd_1 b \rhd_2^{=} b$ and hence $a \notin \mathsf{NF}(\rhd_1 \cdot \rhd_2)$. $\square$

Combining Lemma 4.5 and Lemma 4.6 we establish the following equality:

Lemma 4.7. $\mathsf{NF}(\rhd_1) = \mathsf{NF}(\rhd_1 \cdot \rhd_2^{=})$.

If one relation is included in another also their normal forms are included—but in reverse order.

Lemma 4.8. *If* $\rhd_1 \subseteq \rhd_2$ *then* $\mathsf{NF}(\rhd_2) \subseteq \mathsf{NF}(\rhd_1)$.

Proof. By contra-position we have to show $a \notin \mathsf{NF}(\rhd_2)$ implies $a \notin \mathsf{NF}(\rhd_1)$. Then by $a \notin \mathsf{NF}(\rhd_2)$ we have $\exists b.\ a \rhd_2 b$. By $\rhd_1 \subseteq \rhd_2$ we now know $a \rhd_1 b$, hence $a \notin \mathsf{NF}(\rhd_1)$. $\square$

Finally we investigate how the union of two relations affects their normal forms.

Lemma 4.9. $\mathsf{NF}(\rhd_1 \cup \rhd_2) = \mathsf{NF}(\rhd_2) \cap \mathsf{NF}(\rhd_1)$.

Proof. By $a \in \mathsf{NF}(\rhd_1 \cup \rhd_2)$ we have $\mathsf{a} \in \mathsf{NF}(\rhd_1)$ and $\mathsf{a} \in \mathsf{NF}(\rhd_2)$ hence $a \in \mathsf{NF}(\rhd_1) \cap \mathsf{NF}(\rhd_2)$. $\square$

With these lemmata we start investigating the combination between $\Rightarrow$ and collapsing through concatenation.

4.3 Concatenating Collapse

In this section we first cover $\Rightarrow$ versus $\Rightarrow \cdot \succcurlyeq$ and the reversed $\succcurlyeq \cdot \Rightarrow$, then $\Rightarrow$ versus $\Rightarrow \cdot \succ^!$ and the reversed $\succ^! \cdot \Rightarrow$, and finally $\Rightarrow \cdot \succcurlyeq$ versus $\Rightarrow \cdot \succ^!$, as well as the reversed $\succcurlyeq \cdot \Rightarrow$ versus $\succ^! \cdot \Rightarrow$. Throughout this section we will present (notorious) counter-examples to inclusion. The examples show steps which are possible with one combination but not the other. First we compare the graph rewrite relation without any collapsing ($\Rightarrow$) with the graph rewrite relation concatenated with collapsing ($\Rightarrow \cdot \succcurlyeq$).

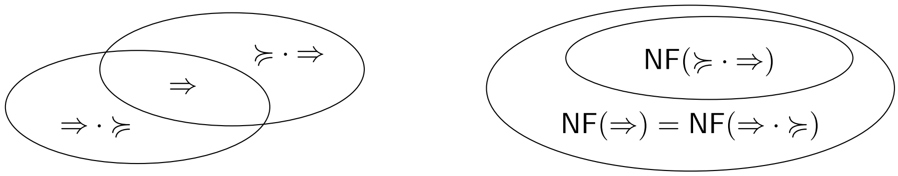
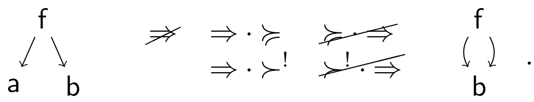


Figure 4.2: Between $\Rightarrow$, $\Rightarrow \cdot \succcurlyeq$, and $\succcurlyeq \cdot \Rightarrow$.

Lemma 4.10. $\Rightarrow \subsetneq \Rightarrow \cdot \succcurlyeq$ *and* $\mathsf{NF}(\Rightarrow) = \mathsf{NF}(\Rightarrow \cdot \succcurlyeq)$*.*

Proof. By reflexivity of $\succcurlyeq$ we have $\Rightarrow \subseteq \Rightarrow \cdot \succcurlyeq$, and Example 4.11 below refutes the reverse direction. The second statement follows from Lemma 4.7. $\square$

Example 4.11. Consider the rule $\mathsf{a} \Rightarrow \mathsf{b}$ and the following rewrite step:



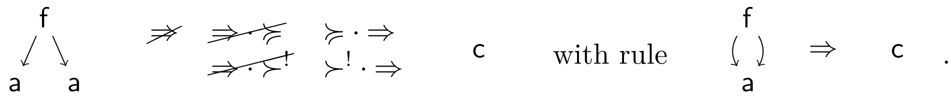
The term graph on the right is not reachable by $\Rightarrow$, neither by $\succcurlyeq \cdot \Rightarrow$ nor $\succ^! \cdot \Rightarrow$, as we lack the ability to share the two bs after performing the rewrite step.

Next, we compare the graph rewrite relation without any collapsing ($\Rightarrow$) with the graph rewrite relation where now collapsing is invoked first ($\succcurlyeq \cdot \Rightarrow$).

Lemma 4.12. $\Rightarrow \subsetneq \succcurlyeq \cdot \Rightarrow$ *and* $\mathsf{NF}(\succcurlyeq \cdot \Rightarrow) \subsetneq \mathsf{NF}(\Rightarrow)$*.*

Proof. By reflexivity of $\succcurlyeq$ we have $\Rightarrow \subseteq \succcurlyeq \cdot \Rightarrow$, and Example 4.13 below refutes the reverse direction. Further $\mathsf{NF}(\succcurlyeq \cdot \Rightarrow) \subseteq \mathsf{NF}(\Rightarrow)$ follows from the reflexivity of $\succcurlyeq$. The reverse is refuted by Example 4.13 too, as the term graph on the left is in $\mathsf{NF}(\Rightarrow)$. $\square$

Example 4.13. Consider the rewrite rule below on the right and the rewrite step:



We cannot apply the rule without a preceding collapsing step. Thus we cannot rewrite with $\Rightarrow$, and neither with $\Rightarrow \cdot \succcurlyeq$ nor with $\Rightarrow \cdot \succ^!$.

Figure 4.2 shows Venn-diagrams of $\Rightarrow$, $\Rightarrow \cdot \succcurlyeq$, and $\succcurlyeq \cdot \Rightarrow$, and their respective normal forms. For $\succcurlyeq \cdot \Rightarrow$ strictly more steps than for $\Rightarrow$ are possible, i.e. more rules are applicable. But consequently we have less normal forms. On the other hand $\Rightarrow$ and $\Rightarrow \cdot \succcurlyeq$ have more normal forms. From a term rewriting perspective these normal forms are un-intuitive as they stem from non-applicability of a rule based on a structural mismatch.

Next we compare the graph rewrite relation ($\Rightarrow$) with the graph rewrite relation followed by collapsing to normal form ($\Rightarrow \cdot \succ^!$).

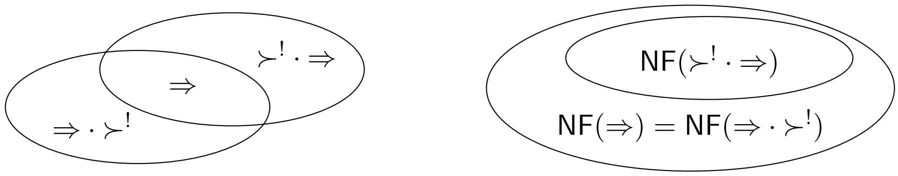


Figure 4.3: Between $\Rightarrow$, $\Rightarrow \cdot \succ^!$, and $\succ^! \cdot \Rightarrow$.

Lemma 4.14. *$\Rightarrow \not\subseteq \Rightarrow \cdot \succ^!$ and $\Rightarrow \cdot \succ^! \not\subseteq \Rightarrow$, but $\mathsf{NF}(\Rightarrow) = \mathsf{NF}(\Rightarrow \cdot \succ^!)$.*

Proof. The first statement is witnessed by Example 4.15 below, and the second statement is witnessed by Example 4.11. For the last statement we know $\mathsf{NF}(\Rightarrow) \subseteq \mathsf{NF}(\Rightarrow \cdot \succ^!)$ from Lemma 4.5. To show the opposite direction we use contra-position to show $S \notin \mathsf{NF}(\Rightarrow)$ implies $S \notin \mathsf{NF}(\Rightarrow \cdot \succ^!)$. By assumption $\exists T.\, S \Rightarrow T$ but then $T \succ^! T'$, where potentially $T = T'$, and thus $S \notin \mathsf{NF}(\Rightarrow \cdot \succ^!)$. $\square$

Example 4.15. Given rule $\mathsf{a} \Rightarrow \mathsf{b}$ and the following rewrite step:

$$\begin{array}{c} \mathsf{f} \\ \swarrow\ \searrow \\ \mathsf{a}\qquad \mathsf{b} \end{array} \quad \Rightarrow \quad \begin{array}{cc} \Rightarrow \cdot \not\succ & \not\succ \cdot \Rightarrow \\ \cancel{\Rightarrow \cdot \succ^!} & \succ^! \cdot \Rightarrow \end{array} \quad \begin{array}{c} \mathsf{f} \\ \swarrow\ \searrow \\ \mathsf{b}\qquad \mathsf{b} \end{array} \quad .$$

Enforcing maximal sharing after the rewrite step forbids to reach the term graph on the right.

Again we compare the graph rewrite relation without any collapsing ($\Rightarrow$) with the graph rewrite relation where now collapsing comes first ($\succ^! \cdot \Rightarrow$).

Lemma 4.16. *$\Rightarrow \not\subseteq \succ^! \cdot \Rightarrow$ and $\succ^! \cdot \Rightarrow \not\subseteq \Rightarrow$, but $\mathsf{NF}(\succ^! \cdot \Rightarrow) \subsetneq \mathsf{NF}(\Rightarrow)$.*

Proof. The first statement is witnessed by Example 4.17 below, the second by Example 4.13. For $\mathsf{NF}(\succ^! \cdot \Rightarrow) \subseteq \mathsf{NF}(\Rightarrow)$, by contra-position we have to show $S \notin \mathsf{NF}(\Rightarrow)$ implies $S \notin \mathsf{NF}(\succ^! \cdot \Rightarrow)$. As $S \notin \mathsf{NF}(\Rightarrow)$, $\exists T.\, S \Rightarrow T$. By Lemma 3.32 $S \Rightarrow T$ implies $S' \succ^! \cdot \Rightarrow T'$ for some $S \succ^! S'$. The opposite direction is refuted by Example 4.13. $\square$

Example 4.17. Given rule $\mathsf{a} \Rightarrow \mathsf{b}$ and the following rewrite step:

$$\begin{array}{c} \mathsf{f} \\ \swarrow\ \searrow \\ \mathsf{a}\qquad \mathsf{a} \end{array} \quad \Rightarrow \quad \begin{array}{cc} \Rightarrow \cdot \not\succ & \not\succ \cdot \Rightarrow \\ \Rightarrow \cdot \succ^! & \cancel{\succ^! \cdot \Rightarrow} \end{array} \quad \begin{array}{c} \mathsf{f} \\ \swarrow\ \searrow \\ \mathsf{a}\qquad \mathsf{b} \end{array} \quad .$$

Enforcing maximal sharing before the rewrite step forbids to reach the term graph on the right.

The Venn-diagrams of $\Rightarrow$, $\Rightarrow \cdot \succ^!$, and $\succ^! \cdot \Rightarrow$, and their respective normal forms are shown in Figure 4.3—and show the same relationships as in Figure 4.2.

At this point we start a brief interlude and investigate the difference between collapsing before and after the rewrite step for more than one step. If we expand a rewrite sequence with $\triangleright \in \{\succsim, \succ^!\}$ we get:

$$T_1 \triangleright T_2 \Rightarrow T_3 \triangleright \cdots \triangleright T_{n-1} \Rightarrow T_n \triangleright T_{n+1} \quad .$$

So only the first or the last collapsing step really differ between $\triangleright \cdot \Rightarrow$ and $\Rightarrow \cdot \triangleright$. For $\triangleright \cdot \Rightarrow$ we have to include the final step $T_n \triangleright T_{n+1}$ for T_{n+1} to be in a more, or the most, space efficient representation. Similar to what we observed in Example 4.11. That is, the last step with rule $\mathsf{a} \Rightarrow \mathsf{b}$ gives rise to one more collapsing step:

$$\ldots \quad \Rightarrow \cdot \triangleright \quad \begin{array}{c} \mathsf{f} \\ \swarrow \; \searrow \\ \mathsf{a} \quad \mathsf{b} \end{array} \quad \Rightarrow \cdot \triangleright \quad \begin{array}{c} \mathsf{f} \\ \swarrow \; \searrow \\ \mathsf{b} \quad \mathsf{b} \end{array} \quad \triangleright \quad \begin{array}{c} \mathsf{f} \\ (\;) \\ \mathsf{b} \end{array} \quad .$$

For the reverse $\Rightarrow \cdot \triangleright$ one has to start with $T_1 \triangleright T_2$, i.e. the initial term graph has to be shared appropriately. Again we observed this already—in Example 4.13. This observation gives raise to the following straight-forward lemma:

Lemma 4.18. *For $\triangleright = \succsim$ or $\triangleright = \succ^!$ we have*

$$\triangleright \cdot (\Rightarrow \cdot \triangleright)^n = (\triangleright \cdot \Rightarrow)^n \cdot \triangleright \quad .$$

Proof. Easy induction on n. □

Finally we investigate the difference between the graph rewrite relation concatenated with collapsing ($\Rightarrow \cdot \succsim$) and the graph rewrite relation concatenated with collapsing to normal form ($\Rightarrow \cdot \succ^!$).

Lemma 4.19. $\Rightarrow \cdot \succ^! \subsetneq \Rightarrow \cdot \succsim$ *and* $\mathsf{NF}(\Rightarrow \cdot \succ^!) = \mathsf{NF}(\Rightarrow \cdot \succsim)$.

Proof. By $\succ^! \subseteq \succsim$ we have $\Rightarrow \cdot \succ^! \subseteq \Rightarrow \cdot \succsim$. We refute the other direction in Example 4.15. By Lemmas 4.10 and 4.14 we have $\mathsf{NF}(\Rightarrow \cdot \succsim) = \mathsf{NF}(\Rightarrow) = \mathsf{NF}(\Rightarrow \cdot \succ^!)$. □

As before we also compare the situation where collapsing comes before the rewrite step.

Lemma 4.20. $\succ^! \cdot \Rightarrow \subsetneq \succsim \cdot \Rightarrow$ *and* $\mathsf{NF}(\succ^! \cdot \Rightarrow) = \mathsf{NF}(\succsim \cdot \Rightarrow)$.

Proof. By $\succ^! \subseteq \succsim$ we have $\succ^! \cdot \Rightarrow \subseteq \succsim \cdot \Rightarrow$. We refute the other direction by Example 4.17. From Lemma 4.8 and the first statement we get $\mathsf{NF}(\succsim \cdot \Rightarrow) \subseteq \mathsf{NF}(\succ^! \cdot \Rightarrow)$. The opposite direction we show by contra-position: $S \notin \mathsf{NF}(\succsim \cdot \Rightarrow)$ implies $S \notin \mathsf{NF}(\succ^! \cdot \Rightarrow)$. By assumption we know $\exists T.\, S \succsim S' \Rightarrow T$, and $S \succsim S' \succ^! S''$. By Lemma 3.32 we know that $S'' \Rightarrow T$, hence $S \notin \mathsf{NF}(\succ^! \cdot \Rightarrow)$. □

Finally Figure 4.4 shows Venn-diagrams of $\Rightarrow \cdot \succsim$ and $\Rightarrow \cdot \succ^!$, of $\succsim \cdot \Rightarrow$ and $\succ^! \cdot \Rightarrow$, and their respective normal forms. After investigating the combinations through concatenation in the next section we investigate the combinations through union.

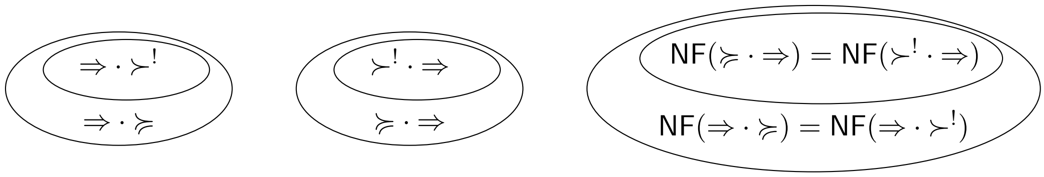


Figure 4.4: Between $\Rightarrow \cdot \not\succcurlyeq$ and $\Rightarrow \cdot \succ^!$, as well as $\not\succcurlyeq \cdot \Rightarrow$ and $\succ^! \cdot \Rightarrow$.

4.4 Union Collapse

In this rather short section we investigate the graph rewrite relation ($\Rightarrow$) with the graph rewrite relation union collapse ($\Rightarrow \cup \succ$) and union collapse to normal form ($\Rightarrow \cup \succ^{!+}$). Finally we compare also the latter two with each other. As in the previous section, we first compare the graph rewrite relation without any collapsing ($\Rightarrow$) with the graph rewrite relation combined with the collapsing relation by union ($\Rightarrow \cup \succ$). The following statement follows directly from definition and Lemma 4.9.

Lemma 4.21. $\Rightarrow \subsetneq \Rightarrow \cup \succ$ *and* $\mathsf{NF}(\Rightarrow \cup \succ) = \mathsf{NF}(\Rightarrow) \cap \mathsf{NF}(\succ)$.

Next we compare the graph rewrite relation without any collapsing ($\Rightarrow$) with the graph rewrite relation now combined with the full collapsing relation ($\Rightarrow \cup \succ^!$).

Lemma 4.22. $\Rightarrow \subsetneq \Rightarrow \cup \succ^{!+}$ *and* $\mathsf{NF}(\Rightarrow \cup \succ^{!+}) = \mathsf{NF}(\Rightarrow) \cap \mathsf{NF}(\succ)$.

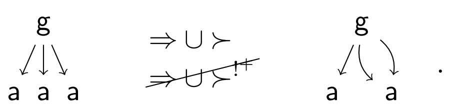
Proof. Again the first statement follows directly from definition. For the second statement by Lemma 4.9 we have $\mathsf{NF}(\Rightarrow \cup \succ^{!+}) = \mathsf{NF}(\Rightarrow) \cap \mathsf{NF}(\succ^{!+})$ from $\succ^{!+}$, defined as $\succ \cdot \succ^!$, we get $\mathsf{NF}(\succ^{!+}) = \mathsf{NF}(\succ)$. □

Finally we compare the difference between collapsing ($\succ$) and collapsing to normal form ($\succ^{!+}$) when in union with graph rewrite relation ($\Rightarrow$).

Lemma 4.23. $\Rightarrow \cup \succ^{!+} \subsetneq \Rightarrow \cup \succ$ *and* $\mathsf{NF}(\Rightarrow \cup \succ^{!+}) = \mathsf{NF}(\Rightarrow \cup \succ)$.

Proof. By $\succ^{!+} \subseteq \succ$ we have $\Rightarrow \cup \succ^{!+} \subseteq \Rightarrow \cup \succ$. The reverse is refuted by Example 4.24 below. The second statement follows directly from Lemma 4.21 and Lemma 4.22. □

Example 4.24. Independent of $\Rightarrow$ we cannot simulate the following step:



Clearly collapsing to normal form forbids any intermediate form of collapsing.

Again we show Venn-diagrams in Figure 4.5 now of the different versions of combining $\Rightarrow$ by union with $\succ$ and $\succ^{!+}$.

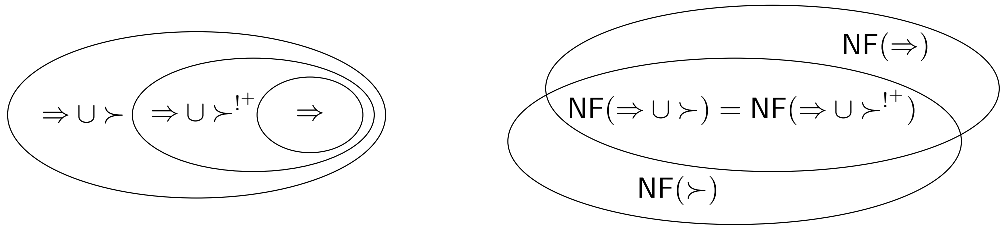


Figure 4.5: Between $\Rightarrow$, $\Rightarrow \cup \succ$, and $\Rightarrow \cup \succ^{!+}$.

4.5 Between Concatenation and Union

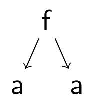
Finally we compare the difference between the combinations based on concatenation and based on union. We compare first $\Rightarrow \cup \succ$ with $\Rightarrow \cdot \succcurlyeq$ and $\succcurlyeq \cdot \Rightarrow$, then $\Rightarrow \cup \succ^{!+}$ with $\Rightarrow \cdot \succcurlyeq$ and $\succcurlyeq \cdot \Rightarrow$. Afterwards we look at $\Rightarrow \cup \succ$ versus $\Rightarrow \cdot \succ^{!}$ and $\succ^{!} \cdot \Rightarrow$, and finally $\Rightarrow \cup \succ^{!+}$ versus $\Rightarrow \cdot \succ^{!}$ and $\succ^{!} \cdot \Rightarrow$.

We start by comparing $\Rightarrow \cup \succ$ with $\Rightarrow \cdot \succcurlyeq$. Clearly they are incomparable for a single step—for one because the former allows a single collapsing step and the latter does not. On the other hand, the latter is a concatenation of two relations, $\Rightarrow$ *and* $\succcurlyeq$, and the former is only a choice of one.

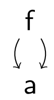
Lemma 4.25. $\Rightarrow \cup \succ \not\subseteq \Rightarrow \cdot \succcurlyeq$ *and* $\Rightarrow \cdot \succcurlyeq \not\subseteq \Rightarrow \cup \succ$*, but* $\mathsf{NF}(\Rightarrow \cup \succ) \subsetneq \mathsf{NF}(\Rightarrow \cdot \succcurlyeq)$.

Proof. The first two statements are shown by Example 4.26 and Example 4.27 below. For $\mathsf{NF}(\Rightarrow \cup \succ) \subsetneq \mathsf{NF}(\Rightarrow \cdot \succ)$ we know that by Lemma 4.9 $\mathsf{NF}(\Rightarrow \cup \succ) = \mathsf{NF}(\Rightarrow) \cap \mathsf{NF}(\succ)$ and by Lemma 4.10 $\mathsf{NF}(\Rightarrow) = \mathsf{NF}(\Rightarrow \cdot \succcurlyeq)$, and thus $\mathsf{NF}(\Rightarrow \cup \succ) = \mathsf{NF}(\Rightarrow \cdot \succcurlyeq) \cap \mathsf{NF}(\succ)$. Now as in general $\mathsf{NF}(\succ) \neq \varnothing$, and clearly $\mathsf{NF}(\succ) \neq \mathsf{NF}(\Rightarrow \cdot \succcurlyeq)$, the statement holds. $\square$

Example 4.26. In general a single collapsing step cannot be achieved when combining the graph rewrite relation and collapsing through concatenation:



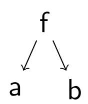
$$\begin{array}{ccc} \cancel{\Rightarrow \cdot \succcurlyeq} & \cancel{\succcurlyeq \cdot \Rightarrow} & \Rightarrow \cup \succ \\ \cancel{\Rightarrow \cdot \succ^{!}} & \cancel{\succ^{!} \cdot \Rightarrow} & \Rightarrow \cup \succ^{!+} \end{array}$$

 .

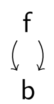
There is no possibility to simulate this step with either $\Rightarrow \cdot \succ$, or $\Rightarrow \cdot \succ^{!}$, $\succcurlyeq \cdot \Rightarrow$, and $\succ^{!} \cdot \Rightarrow$—even for $\mathcal{G} = \varnothing$ with some $f \in \mathcal{F}$ with $\mathsf{ar}(f) \geqslant 2$.

On the other hand we cannot simulate a step with incorporates two relations, e.g. $\Rightarrow \cdot \succcurlyeq$ with $\Rightarrow$ *and* $\succcurlyeq$ with just a choice of one relation from e.g. $\Rightarrow \cup \succ$.

Example 4.27. We repeat Example 4.11. Consider rule $\mathsf{a} \Rightarrow \mathsf{b}$ for the following rewrite step:



$$\begin{array}{cc} \Rightarrow \cdot \succcurlyeq & \cancel{\Rightarrow \cup \succ} \\ \Rightarrow \cdot \succ^{!} & \cancel{\Rightarrow \cup \succ^{!+}} \end{array}$$

 .

It is not possible to simulate $\Rightarrow \cdot \succ$, nor $\Rightarrow \cdot \succ^{!}$, with only one step in $\Rightarrow \cup \succ$, or $\Rightarrow \cup \succ^{!+}$.

As before we now investigate the difference if collapsing comes first ($\succcurlyeq \cdot \Rightarrow$) and compare it to the graph rewrite relation union collapsing ($\Rightarrow \cup \succ$).

Lemma 4.28. $\Rightarrow \cup \succ \not\subseteq \succcurlyeq \cdot \Rightarrow$ *and* $\succcurlyeq \cdot \Rightarrow \not\subseteq \Rightarrow \cup \succ$, *but* $\mathsf{NF}(\Rightarrow \cup \succ) \subsetneq \mathsf{NF}(\succcurlyeq \cdot \Rightarrow)$.

Proof. Again the first statement follows from Example 4.26. The second statement is justified by Example 4.29 below. To show the last statement we show by contra-position $S \notin \mathsf{NF}(\succcurlyeq \cdot \Rightarrow)$ implies $S \notin \mathsf{NF}(\Rightarrow \cup \succ)$. By assumption $\exists S'\,T.\, S \succcurlyeq S' \Rightarrow T$, but then if $S \succ S'$ then $S \notin \mathsf{NF}(\succ)$, and if $S = S'$ then $S' \Rightarrow T$ and $S \notin \mathsf{NF}(\Rightarrow)$. The reverse is refuted by Example 4.15, which is in $\mathsf{NF}(\succcurlyeq \cdot \Rightarrow)$, but not in $\mathsf{NF}(\Rightarrow \cup \succ)$. $\square$

Example 4.29. We recall Example 4.13. Consider the rewrite rule below on the right and the rewrite step:

$$\begin{array}{c} \mathsf{f} \\ \swarrow\ \searrow \\ \mathsf{a}\quad\ \mathsf{a} \end{array} \qquad \begin{array}{cc} \succcurlyeq \cdot \Rightarrow & \cancel{\Rightarrow \cup \succ} \\ \succ^{!} \cdot \Rightarrow & \cancel{\Rightarrow \cup \succ^{!+}} \end{array} \qquad \mathsf{c} \qquad \text{with rule} \qquad \begin{array}{c} \mathsf{f} \\ \downdownarrows \\ \mathsf{a} \end{array} \quad \Rightarrow \quad \mathsf{c} \quad .$$

Again it is not possible to simulate two relations with just a choice of one.

Next we compare the graph rewrite relation concatenated with collapsing ($\Rightarrow \cdot \succcurlyeq$) with the graph rewrite relation union collapsing to normal form ($\Rightarrow \cup \succ^{!+}$).

Lemma 4.30. $\Rightarrow \cup \succ^{!+} \not\subseteq \Rightarrow \cdot \succcurlyeq$ *and* $\Rightarrow \cdot \succcurlyeq \not\subseteq \Rightarrow \cup \succ^{!+}$, *but* $\mathsf{NF}(\Rightarrow \cup \succ^{!+}) \subsetneq \mathsf{NF}(\Rightarrow \cdot \succcurlyeq)$.

Proof. The first two statements follow from Example 4.26 and Example 4.27. The last statement follows from Lemma 4.25 and $\mathsf{NF}(\Rightarrow \cup \succ^{!+}) = \mathsf{NF}(\Rightarrow \cup \succ)$. $\square$

Again we check the case when collapsing comes first ($\succcurlyeq \cdot \Rightarrow$) and compare it to the graph rewrite relation union collapsing to normal form ($\Rightarrow \cup \succ^{!+}$).

Lemma 4.31. $\Rightarrow \cup \succ^{!+} \not\subseteq \succcurlyeq \cdot \Rightarrow$ *and* $\succcurlyeq \cdot \Rightarrow \not\subseteq \Rightarrow \cup \succ^{!+}$, *but* $\mathsf{NF}(\Rightarrow \cup \succ^{!+}) \subsetneq \mathsf{NF}(\succcurlyeq \cdot \Rightarrow)$.

Proof. The first two statements follow from again Example 4.26 and Example 4.29. The last statement follows from Lemma 4.28 and $\mathsf{NF}(\Rightarrow \cup \succ^{!+}) = \mathsf{NF}(\Rightarrow \cup \succ)$. $\square$

Next we compare the concatenation of the graph rewrite relation with collapsing to normal form ($\Rightarrow \cdot \succ^{!}$) with the union of the graph rewrite relation with collapsing ($\Rightarrow \cup \succ$).

Lemma 4.32. $\Rightarrow \cup \succ \not\subseteq \Rightarrow \cdot \succ^{!}$ *and* $\Rightarrow \cdot \succ^{!} \not\subseteq \Rightarrow \cup \succ$, *but* $\mathsf{NF}(\Rightarrow \cup \succ) \subsetneq \mathsf{NF}(\Rightarrow \cdot \succ^{!})$.

Proof. The first two statements follow from Example 4.26 and Example 4.27. The last statement follows from Lemma 4.14, $\mathsf{NF}(\Rightarrow \cdot \succcurlyeq) = \mathsf{NF}(\Rightarrow)$ and Lemma 4.9 as in general $\mathsf{NF}(\succ) \neq \varnothing$. $\square$

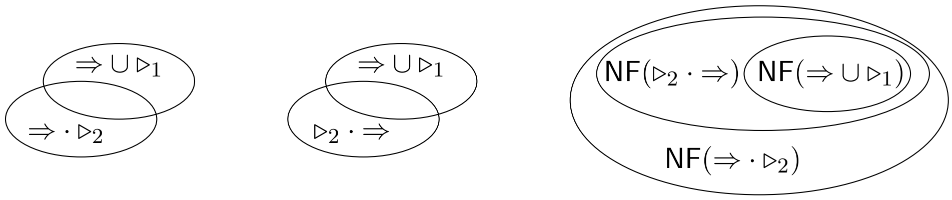


Figure 4.6: Comparing $\Rightarrow \cup \rhd_1$ with $\Rightarrow \cdot \rhd_2$ and $\rhd_2 \cdot \Rightarrow$, where we have $\rhd_1 \in \{\succ, \succ^{!+}\}$ and $\rhd_2 \in \{\not\succeq, \succ^!\}$.

Now we compare again the union of the graph rewrite relation with collapsing ($\Rightarrow \cup \succ$), with the concatenation of collapsing to normal form, but now with collapsing first ($\succ^! \cdot \Rightarrow$).

Lemma 4.33. *$\Rightarrow \cup \succ \not\subseteq \succ^! \cdot \Rightarrow$ and $\succ^! \cdot \Rightarrow \not\subseteq \Rightarrow \cup \succ$, but $\mathsf{NF}(\Rightarrow \cup \succ) \subsetneq \mathsf{NF}(\succ^! \cdot \Rightarrow)$.*

Proof. The first two statements again follow from Example 4.26 and Example 4.29. The last statement follows from Lemma 4.19, $\mathsf{NF}(\succ^! \cdot \Rightarrow) = \mathsf{NF}(\not\succeq \cdot \Rightarrow)$ and Lemma 4.28. □

Then we compare the concatenation of the graph rewrite relation with collapsing to normal form ($\Rightarrow \cdot \succ^!$) with the union of the graph rewrite relation with collapsing to normal form ($\Rightarrow \cup \succ^{!+}$).

Lemma 4.34. *$\Rightarrow \cup \succ^{!+} \not\subseteq \Rightarrow \cdot \succ^!$ and $\Rightarrow \cdot \succ^! \not\subseteq \Rightarrow \cup \succ^{!+}$, but $\mathsf{NF}(\Rightarrow \cup \succ^{!+}) \subsetneq \mathsf{NF}(\Rightarrow \cdot \succ^!)$.*

Proof. The first two statements follow from Example 4.26 and Example 4.27. The last statement follows from Lemma 4.28 and $\mathsf{NF}(\Rightarrow \cup \succ^{!+}) = \mathsf{NF}(\Rightarrow \cup \succ)$. □

Now we compare again the union of the graph rewrite relation with collapsing to normal form ($\Rightarrow \cup \succ^{!+}$), with the union of the graph rewrite relation with collapsing to normal form, but now collapsing comes first ($\succ^! \cdot \Rightarrow$).

Lemma 4.35. *$\Rightarrow \cup \succ^{!+} \not\subseteq \succ^! \cdot \Rightarrow$ and $\succ^! \cdot \Rightarrow \not\subseteq \Rightarrow \cup \succ^{!+}$, but $\mathsf{NF}(\Rightarrow \cup \succ^{!+}) \subsetneq \mathsf{NF}(\succ^! \cdot \Rightarrow)$.*

Proof. The first two statements follow from again Example 4.26 and Example 4.29. The last statement follows from Lemma 4.33 $\mathsf{NF}(\Rightarrow \cup \succ^{!+}) = \mathsf{NF}(\Rightarrow \cup \succ)$. □

As before we summarise the results in a Venn-diagrams in Figure 4.6. Here we use the generic $\rhd_1 \in \{\succ, \succ^{!+}\}$ and $\rhd_2 \in \{\not\succeq, \succ^!\}$ because the relationship holds for any combination of those collapsing relations.

We observed that there is no one-to-one correspondence between rewriting combined with collapsing through concatenation and rewriting combined with collapsing through union. But what about an m-to-n correspondence? This we will inspect next. As we can see from Example 4.29 we require a preceding collapsing step.

Lemma 4.36. *For $\rhd_1 = \succneq$ and $\rhd_2 = \succ$, or $\rhd_1 = \succ^!$ and $\rhd_2 = \succ^{!+}$, and for all $n \in \mathbb{N}$ we have a $m \in \mathbb{N}$ such that*

$$S \rhd_1 \cdot (\Rightarrow \cdot \rhd_1)^n\ T \subseteq S\ (\Rightarrow \cup \rhd_2)^m\ T \quad .$$

Here $m \leqslant (n+1) \times (|S| + n \times C)$ with $C = \mathsf{max}\{|R| \mid L \Rightarrow R \in \mathcal{G}\}$ of $\mathcal{G}$ underlying $\Rightarrow$.

Proof. We have to prove that $S\ \rhd_1\ S'\ (\Rightarrow \cdot \rhd_1)^n\ T$ implies $S\ (\Rightarrow \cup \rhd_2)^m\ T$ for $m \leqslant (n+1) \times (|S| + n \times C)$ by induction over n.

Base Case. $n = 0$: hence $S \rhd_1 S' = T$. For $S \rhd_2 S'$ by Lemma 3.20 if $\rhd_1 = \succneq$ we have $m \leqslant |S| - 1$ hence $m \leqslant (n+1) \times (|S| + n \times C)$. If $\rhd_1 = \succ^!$ we have $m \leqslant 1$ hence also $m \leqslant (n+1) \times (|S| + n \times C)$.

Step Case. $S\ \rhd_1\ S'\ (\Rightarrow \cdot \rhd_1)^n\ T\ \Rightarrow \cdot \rhd_1\ U$ implies $S\ (\Rightarrow \cup \rhd_2)^m\ T\ (\Rightarrow \cup \rhd_2)^k\ U$ for some $m + k \leqslant (n+2) \times (|S| + (n+1) \times C)$. By induction hypothesis we have $m \leqslant (n+1) \times (|S| + n \times C)$. We analyse the last step, $T \Rightarrow T' \rhd_1 U$ simulated by $T \Rightarrow T' \rhd_2^k U$:

Case. $T' \cong U$ Then $k = 0$ and the bound on m holds.

Case. $T' \succ U$ Then $k \leqslant |T'| - 1$ and $|T'| \leqslant |S| + n \times C$ by Lemma 3.33. Then $m + k \leqslant m + |S| + n \times C \leqslant (n+2) \times (|S| + n \times C)$ by induction hypothesis.

□

The above bound is a slight over-approximation. Each performed collapsing step $\succ$ and $\succ^!$ results in an actual decrease of the size of the term graph. This is not reflected in the bound. For $\succ^!$ and $\succ^{!+}$ the opposite direction does not hold as witnessed by the following lemma.

Lemma 4.37. $\succ^! \cdot (\Rightarrow \cdot \succ^!)^n \subsetneq (\Rightarrow \cup \succ^{!+})^m$.

Proof. Lemma 4.36 shows $\succ^! \cdot (\Rightarrow \cdot \succ^!)^n \subseteq (\Rightarrow \cup \succ^{!+})^m$. This inclusion is strict, as $\succ^!$ is not reflexive. With the rule $\mathsf{a} \Rightarrow \mathsf{b}$ the following step is possible:

$$\begin{array}{c} \mathsf{f} \\ \swarrow\ \ \searrow \\ \mathsf{a}\qquad\mathsf{a}\end{array} \qquad \begin{array}{c}(\Rightarrow \cup \succ^{!+})^m \\ \cancel{\succ^! \cdot (\Rightarrow \cdot \succ^!)^n}\end{array} \qquad \begin{array}{c} \mathsf{f} \\ \swarrow\ \ \searrow \\ \mathsf{a}\qquad\mathsf{b}\end{array} \quad .$$

□

For the other direction we try not to collapse to normal form, that is we employ only $\succneq$ and $\succ$.

Lemma 4.38. *For all $m \in \mathbb{N}$ there is an $n \leqslant m$ such that*

$$S\ (\Rightarrow \cup \succ)^m\ T \subseteq S \succneq \cdot (\Rightarrow \cdot \succneq)^n\ T \quad .$$

Proof. We prove $S\ (\Rightarrow \cup \succ)^m\ T$ implies $S \succneq S'\ (\Rightarrow \cdot \succneq)^n\ T$ by induction over m.

Base Case. $m = 0$: hence $S = T$ and by reflexivity of $\succcurlyeq$ also $S \succcurlyeq S = T$ holds for $n = 0$.

Step Case. $S\ (\Rightarrow \cup \succ)^m\ T\ (\Rightarrow \cup \succ)\ U$ implies $S \succcurlyeq S'\ (\Rightarrow \cdot \succcurlyeq)^n\ T\ (\Rightarrow \cdot \succcurlyeq)^k\ U$ for some $n + k \leqslant m + 1$ for $k \leqslant 1$. We analyse the last step:

Case. $S\ (\Rightarrow \cup \succ)^m\ T \Rightarrow U$ By induction hypothesis exists some $n \leqslant m$ and by reflexivity of $\succcurlyeq$ we have $T \Rightarrow \cdot \succcurlyeq U$, hence $k = 1$ and $n + k \leqslant m + 1$.

Case. $S\ (\Rightarrow \cup \succ)^m\ T \succ U$ If the $(m-1)^{th}$ step is $\succ$, by transitivity of $\succ$ we have $S\ (\Rightarrow \cup \succ)^m\ U$ and hence by induction hypothesis exists an $n \leqslant m$. Otherwise, we have $S\ (\Rightarrow \cup \succ)^{m-1}\ T' \Rightarrow T \succ U$, which can be combined to the single step $T' \Rightarrow \cdot \succcurlyeq U$. By applying the induction hypothesis we get $S \succcurlyeq S'\ (\Rightarrow \cdot \succcurlyeq)^{n-1}\ U$ for some $n - 1 \leqslant m - 1$ and together with $T' \Rightarrow \cdot \succcurlyeq U$ we have $n \leqslant m$.

□

The next Lemma 4.39 follows immediately from the above Lemma 4.36 and Lemma 4.38, but is also a equality obtained by standard reasoning in the Kleene algebra: for binary relations $\rhd_1$ and $\rhd_2$, where $\rhd_2$ is transitive and $\rhd_2^{=}$ is the reflexive closure of $\rhd_2$ we have $(\rhd_1 \cup \rhd_2)^m = \rhd_2^{=} \cdot (\rhd_1 \cdot \rhd_2^{=})^n$.

Lemma 4.39. *For $n \leqslant m$ and $m \leqslant (n{+}1){\times}(|S|{+}n{\times}C)$ with $C = \mathsf{max}\{|R| \mid L \Rightarrow R \in \mathcal{G}\}$ of $\mathcal{G}$ underlying $\Rightarrow$ we have:*

$$S \succcurlyeq \cdot\ (\Rightarrow \cdot \succcurlyeq)^n\ T = S\ (\Rightarrow \cup \succ)^m\ T \quad .$$

Hence there is a linear relationship between the graph rewrite relation combined with union and combined through concatenation—given an appropriate pre- or post-processing step following Lemma 4.18.

This chapter illustrated how sensitive term graph rewriting is to small changes. Whether we use concatenation or union, collapsing or collapsing to normal form, collapsing before or after the graph rewrite step—we always provoke slightly different effects. In particular some notorious examples arise. On the one hand we cannot *apply* a rewrite rule—either because we cannot collapse the term graph or we collapse to normal form and thereby collapse too much. On the other hand, we cannot *reach* some term graphs after applying a rewrite step. However most of the differences vanish, if we do not restrict to single steps.

With this we conclude our investigation of how to combine the graph rewriting relation with the collapsing relation. In the next two chapters we investigate the termination behaviour of graph rewriting. Therefore we start with Kruskal's Tree Theorem for term graphs.

5 Kruskal's Tree Theorem for Term Graphs

We know that termination of term rewriting implies termination of graph rewriting. But: the opposite direction does not hold. This is witnessed by, e.g. Toyama's counter example for modularity of termination in term rewriting [35]. For term graph rewriting this example does terminate [20].

Hence there are some terminating GRSs for which the corresponding TRS does not terminate. We are interested in this gap and in techniques to show termination of such GRSs. This is also the interest of [28] where Plump develops a technique to show termination of term graph rewrite systems. In this chapter we follow and extend upon his idea. We published the results in this and the next chapter in [23]. We start by the example that serves as motivation of many works on term graph rewriting, e.g. [26], [20], [33], or [37].

Example 5.1. Recall Toyama's TRS $\mathcal{R}$:

$$\mathsf{f}(\mathsf{a},\mathsf{b},x) \to \mathsf{f}(x,x,x) \quad , \quad \mathsf{g}(x_1,x_2) \to x_i, i \in \{1,2\} \quad .$$

This allows the non-terminating term rewrite sequence:

$$\mathsf{f}(\mathsf{a},\mathsf{b},\mathsf{g}(\mathsf{a},\mathsf{b})) \to_{\mathcal{R}} \mathsf{f}(\mathsf{g}(\mathsf{a},\mathsf{b}),\mathsf{g}(\mathsf{a},\mathsf{b}),\mathsf{g}(\mathsf{a},\mathsf{b})) \to^2_{\mathcal{R}} \mathsf{f}(\mathsf{a},\mathsf{b},\mathsf{g}(\mathsf{a},\mathsf{b})) \to_{\mathcal{R}} \cdots$$

The corresponding GRS $\mathcal{G}(\mathcal{R})$ has a unique representation. It is depicted next:

$$\begin{array}{c} \mathsf{f} \\ \swarrow\downarrow\searrow \\ \mathsf{a}\ \ \mathsf{b}\ \ x \end{array} \quad \Rightarrow \quad \begin{array}{c} \mathsf{f} \\ (\downarrow) \\ x \end{array} \quad , \quad \begin{array}{c} \mathsf{g} \\ \swarrow\ \ \searrow \\ x_1\quad x_2 \end{array} \quad \Rightarrow \quad x_i,\ i \in \{1,2\} \quad .$$

Note that in the first rule on the right-hand side the node corresponding to the variable x is shared. Now we try to simulate the above derivation starting from a graph representation of the above term:

$$\begin{array}{c} \mathsf{f} \\ \swarrow\downarrow\searrow \\ \mathsf{a}\ \ \mathsf{b}\ \ \mathsf{g} \\ \quad\ \ \swarrow\ \ \searrow \\ \quad\ \ \mathsf{a}\quad \mathsf{b} \end{array} \quad \Rightarrow_{\mathcal{G}(\mathcal{R})} \quad \begin{array}{c} \mathsf{f} \\ (\downarrow) \\ \mathsf{g} \\ \swarrow\ \ \searrow \\ \mathsf{a}\quad \mathsf{b} \end{array} \quad \Rightarrow_{\mathcal{G}(\mathcal{R})} \quad \begin{array}{c} \mathsf{f} \\ (\downarrow) \\ \mathsf{a} \end{array} \quad .$$

As opposed to the derivation with term rewriting the graph rewriting derivation reaches a normal form and is terminating. In the absence of uncollapsing it is not possible to simulate the term rewrite sequence.[1]

[1] The system is left linear and hence no explicit collapsing operation, e.g. $\succ$, is necessary.

Key here is the absence of uncollapsing. A node which is shared—either through a rewrite or a collapsing step—cannot be uncollapsed again by an explicit operation.[2] So in $\mathcal{G}(\mathcal{R})$ we can distinguish the function symbol f by the sharing of its arguments. In Example 5.1 in the first rule on the lhs the function symbol f has three distinct argument nodes, but on the rhs the three arguments of f are represented by the same node. This has been explored by Plump [28]. He defines an order on the Tops of term graphs. The Top of a term graph takes the structure of the arguments of a function symbol into account. We continue with Example 5.1 and show the Tops of the first rule.

Example 5.2. Let $\triangle$ be a fresh constant—similar to $\square$ in a term. The left rule in Example 5.1 gives rise to the following two different Tops for the function symbol f:

$$\begin{tikzcd}[column sep=tiny] & \mathsf{f} \arrow[dl] \arrow[d] \arrow[dr] & \\ \triangle & \triangle & \triangle \end{tikzcd} \quad , \quad \begin{tikzcd} \mathsf{f} \arrow[d, bend right] \arrow[d] \arrow[d, bend left] \\ \triangle \end{tikzcd} \quad .$$

We give formal definitions for *(i)* the Top of a term graph S starting from a node n, and *(ii)* the set of Tops based on a function symbol f. We start with *(i)* and compute the Top of a term graph S from a node n. Thereby, n remains unchanged, the labels for the successors are set to $\triangle$, and the successors of the successors of n are discarded.

Definition 5.3. Let S be a term graph over $\mathcal{F}$, $n \in S$, and $\triangle$ a fresh constant wrt. $\mathcal{F}$. Then $\mathsf{Top}_S(n) = (\{n\} \cup \mathsf{succ}_S(n), \mathsf{lab}_{\mathsf{Top}}, \mathsf{succ}_{\mathsf{Top}})$ is a term graph, where

- $\mathsf{lab}_{\mathsf{Top}}(n) = \mathsf{lab}_S(n)$ and $\mathsf{succ}_{\mathsf{Top}}(n) = \mathsf{succ}_S(n)$,
- for $\mathsf{succ}_S(n) = n_1, \ldots, n_k$ and $1 \leqslant i \leqslant k$, set $\mathsf{lab}_{\mathsf{Top}}(n_i) =\triangle$, and $\mathsf{succ}_{\mathsf{Top}}(n_i) = [\ ]$.

We abbreviate $\mathsf{Top}_S(\mathsf{rt}(S))$ with $\mathsf{Top}(S)$.

The previous Definition 5.3 shows how to compute a Top from a given term graph and a given node. The next Definition 5.4 defines *(ii)*, the set of Tops computed from a function symbol by exploiting the reflexive and transitive closure of $\succ$.

Definition 5.4. Let $f \in \mathcal{F}$, $\triangle$ a fresh constant, and $S = \mathsf{tree}^{\mathcal{G}}(f(\triangle, \ldots, \triangle))$. Then $\mathsf{Tops}(f) = \{T \mid S \succ^* T\}$. This definition extends to a signature $\mathcal{F}$ with $\mathsf{Tops}(\mathcal{F}) = \bigcup_{f \in \mathcal{F}} \mathsf{Tops}(f)$.

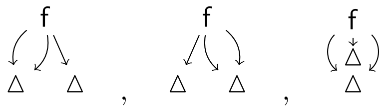
Neither Top nor Tops necessarily produce canonical term graphs. For Tops this depends on the implementation of collapsing ($\succ$), as $\mathsf{tree}^{\mathcal{G}}$ produces canonical term graphs. This is a rather technical detail concerning node numbers but to ensure for some term graph S that $\mathsf{Top}(S) \in \mathsf{Tops}(\mathcal{F})$, we have to deal with it. To do so we extend the definition of Tops to capture all isomorphic copies.

Definition 5.4 (continued). For $T \in \mathsf{Tops}(\mathcal{F})$ and $T \cong T'$, let $T' \in \mathsf{Tops}(\mathcal{F})$.

[2] If there is an edge to a node from the context to the matched lhs, this node will remain after a rewrite step. This could be seen as some form of uncollapsing, cf. Chapter 6.

By definition the elements of Tops are also term graphs, i.e. $\mathsf{Tops}(\mathcal{F}) \subseteq \mathcal{TG}(\mathcal{F} \cup \{\triangle\})$.

Example 5.5. If we add the three Tops:

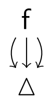
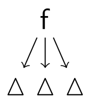


to the Tops from Example 5.2, we have $\mathsf{Tops}(\mathsf{f})$ with $\mathsf{ar}(\mathsf{f}) = 3$.

As a side remark: For a function symbol f with $\mathsf{ar}(f) = k$ the amount of Tops corresponds to the equivalence relation, or partitions, of k-element sets. It is given by $B_0 = 1$ and $B_k = \sum_{i=0}^{k-1} \binom{k}{i} B_{k-1}$.

Now we can define an order $\sqsubseteq$, a *precedence*, on $\mathsf{Tops}(\mathcal{F})$. This is similar to the precedence on the signature $\mathcal{F}$ in the term rewrite setting. We start by giving an order on the Tops of Toyama's GRS.

Example 5.6. The GRS in Example 5.1 can be proven terminating with the following precedence on $\mathsf{Tops}(\mathcal{F})$:

 $\sqsubseteq$  and $\mathsf{a} \sqsubseteq \mathsf{b}$.

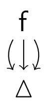
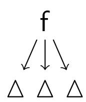
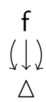
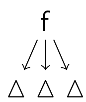
Definition 5.7. A *precedence* on a signature $\mathcal{F}$ is a transitive relation $\sqsubseteq$ on $\mathsf{Tops}(\mathcal{F})$ such that for $S, T \in \mathsf{Tops}(\mathcal{F})$ the following conditions hold:

(i) $S \cong T$ implies $S \sqsubseteq T$ and $T \sqsubseteq S$, and

(ii) $T \sqsubseteq S$ implies $|T| \leqslant |S|$.

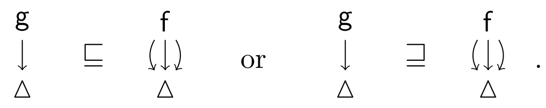
By Condition *(i)* $\sqsubseteq$ is reflexive, but also includes isomorphic copies of Top. Condition *(ii)* guarantees that the larger Top has at least as many distinct successor nodes as the smaller one. While capturing isomorphic copies to avoid problems with node numbers in Condition *(i)* is a technical detail, Condition *(ii)* is crucial. Thus we give an intuition in the following example.

Example 5.8. First we observe a difference to term rewriting: We can distinguish the same function symbol f by the sharing of its successor nodes. So an f which shares more successor nodes can be smaller in the precedence than an f which shares fewer, as on the left:

 $\sqsubseteq$  but  $\not\sqsupseteq$  .

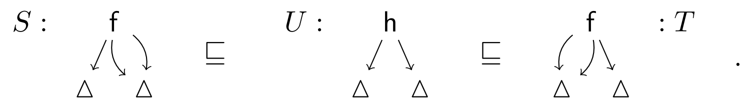
On the right we try the opposite, embedding an f which shares less. This violates Condition *(ii)*. Intuitively we try here to embed one successor node in three distinct successor nodes.

The second difference to term rewriting is that a function symbol f with larger arity can embed the function symbol g with smaller arity and *vice versa*.



The left embedding does not hold any surprises—the function symbol f with arity 3 embeds the function symbol g with arity 1. The right embedding seems unusual: We embed one distinct successor node on the left into one distinct successor node on the right. Granted this successor node represents three arguments, but from the structure we know these arguments to be, and to remain[3], equal. Consequently Condition *(ii)* is not violated.

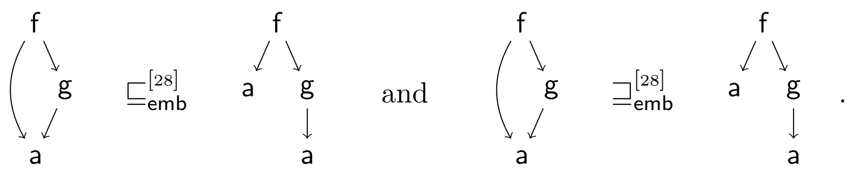
Finally note that for two tops with the *same* function symbol f one may embed the other although they are incomparable wrt. $\succ$. That is, for $S, T \in \mathsf{Tops}(\mathsf{f})$, with $S \not\succ T$ and $T \not\succ S$, still $T \sqsubseteq S$ may hold, as witnessed here:



There is no restriction that forbids $S \sqsubseteq U$ and none for $U \sqsubseteq T$. Hence by transitivity of $\sqsubseteq$ we have to have $S \sqsubseteq T$. Condition *(ii)* in Definition 5.7 only assures that there are sufficiently many distinct successor nodes to embed the smaller Top. No restrictions on the order of the successor nodes is presumed—as opposed to Plump's [28], where the order on Top has to be compatible with collapsing.

So far we considered the precedence of Tops. Now we want to extend this precedence to an order on term graphs—an embedding relation $\sqsubseteq_{\mathsf{emb}}$. Plump defines $\sqsubseteq_{\mathsf{emb}}$ by encoding the arguments below the root into strings [28]. We write $\sqsubseteq_{\mathsf{emb}}^{[28]}$ to indicate his notion of embedding. Through this the sharing information of nodes below direct arguments to the root is lost. To clarify this, consider the following example given in [28].

Example 5.9. The following term graphs are mutually embedded in each other:



This mutual embedding is counter-intuitive. For us this is the starting point for a new definition of embedding. This new definition should take sharing into account, i.e. $\preccurlyeq \subseteq \sqsubseteq_{\mathsf{emb}}$. Therefore we follow two main ideas: For one, the new definition is based on a morphism between the two graphs as morphisms inherently are about finding structures. Secondly the new definition treats the argument of a term graph as *one* graph. We start with motivating the second idea.

[3]Due to the absence of explicit uncollapsing.

5.1 The Argument of a Term Graph

Essentially the question is: What is the argument of a term graph?

Example 5.10. Consider the following three term graphs.

$$
\begin{tikzcd} \mathsf{f}:1 \arrow[d, bend right] \arrow[d, bend left] \\ \mathsf{g}:2 \arrow[d] \\ \mathsf{a}:3 \end{tikzcd}
\preccurlyeq
\begin{tikzcd} & \mathsf{f}:1 \arrow[dl] \arrow[dr] & \\ \mathsf{g}:2 \arrow[dr] & & \mathsf{g}:3 \arrow[dl] \\ & \mathsf{a}:4 & \end{tikzcd}
\preccurlyeq
\begin{tikzcd} & \mathsf{f}:1 \arrow[dl] \arrow[dr] & \\ \mathsf{g}:2 \arrow[d] & & \mathsf{g}:3 \arrow[d] \\ \mathsf{a}:4 & & \mathsf{a}:5 \end{tikzcd} .
$$

We want to compute their argument(s). Now we have two options. For one, we could consider the arguments as separate term graphs. Each of the three term graphs above then has two distinct arguments—two as dictated by the arity of f:

$$
\left\{ \begin{tikzcd} \mathsf{g}:2 \arrow[d] \\ \mathsf{a}:3 \end{tikzcd} , \begin{tikzcd} \mathsf{g}:2 \arrow[d] \\ \mathsf{a}:3 \end{tikzcd} \right\} ,
\left\{ \begin{tikzcd} \mathsf{g}:2 \arrow[d] \\ \mathsf{a}:4 \end{tikzcd} , \begin{tikzcd} \mathsf{g}:3 \arrow[d] \\ \mathsf{a}:4 \end{tikzcd} \right\} ,
\left\{ \begin{tikzcd} \mathsf{g}:2 \arrow[d] \\ \mathsf{a}:4 \end{tikzcd} , \begin{tikzcd} \mathsf{g}:3 \arrow[d] \\ \mathsf{a}:5 \end{tikzcd} \right\} .
$$

This is the equivalent to arguments in the term rewriting setting. Note that implicitly the sharing information is still kept through the node numbers. On the other hand we also could consider the arguments as one graph and get the following three argument graphs for the three term graphs:

$$
\begin{tikzcd} \mathsf{g}:2 \arrow[d] \\ \mathsf{a}:3 \end{tikzcd} ,
\begin{tikzcd} \mathsf{g}:2 \arrow[dr] & & \mathsf{g}:3 \arrow[dl] \\ & \mathsf{a}:4 & \end{tikzcd} ,
\begin{tikzcd} \mathsf{g}:2 \arrow[d] & \mathsf{g}:3 \arrow[d] \\ \mathsf{a}:4 & \mathsf{a}:5 \end{tikzcd} .
$$

By considering the argument of a term graph as *one* graph, we explicitly keep information about shared nodes. One may also note the similarity to a graph rewrite rule (cf. Definition 3.21), which is also *one* graph with two distinguished roots.

As our original goal is to keep information about sharing we prefer the second version: one argument graph. However we also loose information: What are "roots" of the argument graph?

Example 5.11. Consider the following term graph S_1:

$$
S_1 : \begin{tikzcd} & \mathsf{f}:1 \arrow[dl] \arrow[dd, bend left] \\ \mathsf{g}:2 \arrow[dr] & \\ & \mathsf{a}:3 \end{tikzcd}
\quad \text{with argument} \quad
\begin{tikzcd} \mathsf{g}:2 \arrow[d] \\ \mathsf{a}:3 \end{tikzcd}
\quad \begin{array}{l}\text{which is also} \\ \text{also argument of}\end{array} \quad
\begin{tikzcd} \mathsf{f}:1 \arrow[d, bend right] \arrow[d, bend left] \\ \mathsf{g}:2 \arrow[d] \\ \mathsf{a}:3 \end{tikzcd} : S_2 .
$$

The information about the roots of the argument graph has been lost. To keep this we extend the argument graph by remembering the roots of the argument: $[2, 3]$ for the argument of S_1 and [②, ②] for the argument of S_2. Strictly speaking ③ is *not* a root as not every node is reachable from ③ (compare Definition 3.3). Hence we refer to the roots of the argument graph as *inlets*.

We now formally define the notion of argument graph based on *inlet graphs*.

Definition 5.12. Let $G = (N, \mathsf{succ}, \mathsf{lab})$ be a term dag. An *inlet graph*, extends G with an ordered sequence of nodes, $\mathsf{inlets} = [n_1, \ldots, n_k]$, where $n_1, \ldots, n_k \in N$.

The definition of sub-graph to inlet graphs extends in a natural way. We simply consider all nodes reachable from inlets $[n_1, \ldots, n_k]$.

Definition 5.13. The sub-graph $G{\restriction}[n_1, \ldots, n_k]$ of an inlet graph $G = (N, \mathsf{succ}, \mathsf{lab}, \mathsf{inlets})$ is $G' = (N', \mathsf{succ}', \mathsf{lab}', [n_1, \ldots, n_k])$, where $N' = \{n \mid n_i \rightharpoonup^* n, 1 \leqslant i \leqslant k\}$, and the domains of $\mathsf{succ}_{G'}$ and $\mathsf{lab}_{G'}$ are restricted to $N_{G'}$.

We can now use inlet graphs together with the definition of sub-graph to compute the argument graph of a term graph: the root is deleted and the inlets of are the direct successors of the original root.

Definition 5.14. The *argument graph* of a term graph $T = (N, \mathsf{succ}, \mathsf{lab})$, denoted by $\mathsf{arg}(T)$, is an inlet graph $(N, \mathsf{succ}, \mathsf{lab}, [\mathsf{rt}(T)]){\restriction}\mathsf{inlets}$ where $\mathsf{inlets} = \mathsf{succ}(\mathsf{rt}(T))$.

In this definition we already hid the construction of an inlet graph from a term graph T: set $\mathsf{inlets} = \mathsf{rt}(T)$. We illustrate argument graphs by an example.

Example 5.15. Reconsider Example 5.11. The argument graphs for S_1 and S_2 have the same $N = \{②, ③\}$, the same $\mathsf{succ}(②) = ③$ and $\mathsf{succ}(③) = [\,]$, and the same $\mathsf{lab}(②) = \mathsf{g}$ and $\mathsf{lab}(③) = \mathsf{a}$. However, they are different in their inlets: for the left graph [②, ③] versus [②, ②] for the right graph.

The next example of argument graph illustrates the argument of an argument of an argument.

Example 5.16. The following graph

$$\begin{tikzcd}[column sep=small] & \mathsf{f} : \textcircled{1} \arrow[dl] \arrow[dr] \arrow[dd, bend left] & \\ \mathsf{g} : \textcircled{2} \arrow[dr] & & \mathsf{g} : \textcircled{4} \arrow[ddl, bend left] \\ & \mathsf{g} : \textcircled{3} \arrow[d] & \\ & \mathsf{a} : \textcircled{5} & \end{tikzcd}$$

has

$$\begin{aligned} \mathsf{inlets}(\mathsf{arg}(G)) &= [\textcircled{2}, \textcircled{3}, \textcircled{4}] \\ \mathsf{inlets}(\mathsf{arg}(\mathsf{arg}((G)))) &= [\textcircled{3}, \textcircled{5}, \textcircled{5}] \\ \mathsf{inlets}(\mathsf{arg}(\mathsf{arg}(\mathsf{arg}((G))))) &= [\textcircled{5}] \end{aligned} \quad .$$

By construction, if n is a root in an inlet graph then $n \in \mathsf{inlets}$. The reverse does not hold as witnessed by node ③ in Example 5.11

5.2 Embedding

We now define an embedding relation on inlet graphs. It has a similar structure as Definition 3.15, which defines a Δ-morphism between two term graphs.

We continually develop our definition of embedding throughout this section, but start with giving an intuition.

Example 5.17. The following three graphs are embedded from the left to the right, under the given precedence:

$$
\begin{tikzcd}[column sep=small]
 & \mathsf{f} : ① \arrow[dl] \arrow[dr] & & & \mathsf{g} : Ⓐ \arrow[d] & & \mathsf{f} : \text{Ⅰ} \arrow[d, bend right] \arrow[d, bend left] \\
\mathsf{a} : ② & & \mathsf{a} : ③ \arrow[r, phantom, "\sqsupseteq_{\mathsf{emb}}"] & {} & \mathsf{a} : Ⓑ \arrow[r, phantom, "\sqsupseteq_{\mathsf{emb}}"] & {} & \mathsf{a} : \text{Ⅱ}
\end{tikzcd}
\quad \text{with precedence} \quad
\begin{tikzcd}[column sep=small]
 & \mathsf{f} \arrow[dl] \arrow[dr] & & & \mathsf{g} \arrow[d] & & \mathsf{f} \arrow[d, bend right] \arrow[d, bend left] \\
\triangle & & \triangle \arrow[r, phantom, "\sqsupseteq"] & {} & \triangle \arrow[r, phantom, "\sqsupseteq"] & {} & \triangle
\end{tikzcd} .
$$

In a first attempt to define embedding we try to find a morphism from the embedded "smaller" graph to the embedding "larger" graph—which will not work.

Definition 5.18 (first attempt)**.** Let $\sqsubseteq$ be a precedence. We say that S is *embedded* in T, denoted as $S \sqsubseteq_{\mathsf{emb}} T$, if there exists a function $m \colon S \to T$ such that for all nodes $s \in S$, we have

(i) $\mathsf{Top}_S(s) \sqsubseteq \mathsf{Top}_T(m(s))$, and

(ii) if $s \rightharpoonup_S s'$ for some $s' \in S$, then $m(s) \rightharpoonup_T^+ m(s')$ holds.

Condition *(i)* demands the decrease in the order $\sqsubseteq$ of the Top for every node. Condition *(ii)* demands that every path in the smaller graph can be simulated by a, potentially larger, path in the larger graph. To illustrate the definition consider the following example.

Example 5.19. The embedding given below is valid after Definition 5.18:

$$
S: \;
\begin{tikzcd}[column sep=small]
 & \mathsf{f} : Ⓐ \arrow[dl] \arrow[dr] & \\
\mathsf{g} : Ⓑ \arrow[d] & & \mathsf{g} : Ⓓ \arrow[d] \\
\mathsf{a} : Ⓒ & & \mathsf{a} : Ⓔ
\end{tikzcd}
\quad \sqsubseteq_{\mathsf{emb}} \quad
\begin{tikzcd}[column sep=small]
 & \mathsf{f} : ① \arrow[dl] \arrow[dr] & \\
\mathsf{g} : ② \arrow[d] & & \mathsf{a} : ④ \\
\mathsf{a} : ③ & &
\end{tikzcd}
\; : T \quad .
$$

Here the morphism $m : S \to T$ satisfies both conditions: $m(Ⓐ) = ①$, $m(Ⓑ) = ② = m(Ⓓ)$, and $m(Ⓒ) = ③ = m(Ⓔ)$.

This embedding could be prohibited by demanding m to be injective. But demanding injectivitiy prohibits to capture sharing in the embedding relation.

Example 5.20. Embedding the smaller, i.e. more collapsed, S in T is not possible, as can be seen next:

$$
\begin{tikzcd}
S: & & \mathsf{f} : A \arrow[dl] \arrow[dr] & & & & \mathsf{f} : 1 \arrow[dl] \arrow[dr] & & : T \\
& \mathsf{g} : B \arrow[dr] & & \mathsf{g} : C \arrow[dl] & \not\sqsubseteq_{\mathsf{emb}} & \mathsf{g} : 2 \arrow[d] & & \mathsf{g} : 3 \arrow[d] & \\
& & \mathsf{a} : D \arrow[rrr, squiggly, bend right] \arrow[rrrrr, squiggly, bend right] & & & \mathsf{a} : 4 & & \mathsf{a} : 5 &
\end{tikzcd}
.
$$

We cannot map D to 4 *and* 5.

The above Examples 5.19 and 5.20 demonstrate that a mapping from the embedded to the embedding graph prohibits to take collapsing into account. But taking collapsing into account was our aim. Thus in a second attempt we map from the “larger” embedding graph to the “smaller” embedded graph.

Definition 5.21 (second attempt)**.** Let $\sqsubseteq$ be a precedence. We say that S *embeds* T, denoted as $S \sqsupseteq_{\mathsf{emb}} T$, if there exists a partial, surjective function $m\colon S \to T$ such that for all nodes s in the domain of m:

(i) $\mathsf{Top}_S(s) \sqsupseteq \mathsf{Top}_T(m(s))$, and

(ii) $m(s) \rightharpoonup_T m(s')$ implies $s \rightharpoonup_S^+ n'$ for some $n' \in \{n \mid m(n) = m(s')\}$.

Condition *(i)* is a straight-forward adaptation of Definition 5.18 ensuring that the embedded node’s Top is smaller in the precedence. Condition *(ii)* is a bit more involved, because m is not necessarily injective. In this case, the node s' in $m(s')$ is not uniquely determined, but element of the set of pre-images of s', i.e. $m^{-1}(s')$. Surjectivity ensures that every node in T is embedded.

Example 5.22. Recall Example 5.20 , where we want to justify $S \sqsubseteq_{\mathsf{emb}} T$. We map $m(2) = B$ and $m(4) = D$, and we have $B = m(2) \rightharpoonup m(4) = D$. But we also (have to) map $D = m(5)$, and have $m(2) \rightharpoonup m(5)$, but clearly $2 \not\rightharpoonup 5$. However, we already have a witness in $m^{-1}(D) = \{4, 5\}$, i.e. $n' = 4$ wrt. Definition 5.21 where $2 \rightharpoonup 4$.

Both definitions of embedding are very permissive and do not take the order of the arguments into account. Consider the next example, where the arguments are swapped, but the embedding holds in both directions.

Example 5.23. The two term graphs representing the terms $\mathsf{f}(\mathsf{a}, \mathsf{b})$ and $\mathsf{f}(\mathsf{b}, \mathsf{a})$ are mutually embedded:

$$
\begin{tikzcd}
& \mathsf{f} : 1 \arrow[dl] \arrow[dr] & & \sqsubseteq_{\mathsf{emb}} & & \mathsf{f} : A \arrow[dl] \arrow[dr] & \\
\mathsf{a} : 2 & & \mathsf{b} : 3 & \sqsupseteq_{\mathsf{emb}} & \mathsf{b} : B & & \mathsf{a} : C
\end{tikzcd}
.
$$

From left to right we have the morphism m with $m(1) = A$, $m(2) = B$, and $m(3) = C$. However, the inverse morphism m^{-1} fulfills the conditions too.

This leads us to our third attempt where we also want to take the order of the arguments into account. Informally speaking we want to preserve the relative order between the nodes: if a node n is "left of" a node n', $m(n)$ should be "left of" $m(n')$ in the embedded graph. In the following we describe requirements on this "left of"-relation, which we write as $\ll$.

It is not sufficient to define $\ll$ only on direct successors of some node. Put differently: a local perspective is not sufficient. We have to take the successors of the successors into account, as shown by the next example.

Example 5.24. We want to include the following embedding, with $m(\textcircled{4}) = \textcircled{B}$ and $m(\textcircled{3}) = \textcircled{C}$.

$$
\begin{tikzcd}
 & \mathsf{f} : \textcircled{1} \arrow[dl] \arrow[dr] & & & & \\
\mathsf{g} : \textcircled{2} \arrow[d] & & \mathsf{b} : \textcircled{4} & \sqsupseteq_{\mathsf{emb}} & & \mathsf{f} : \textcircled{A} \arrow[dl] \arrow[dr] & \\
\mathsf{a} : \textcircled{3} & & & & \mathsf{a} : \textcircled{B} & & \mathsf{b} : \textcircled{C}
\end{tikzcd}
.
$$

Intuitively we have $\textcircled{B} \ll \textcircled{C}$, hence we need to compare $\textcircled{3}$ and $\textcircled{4}$.

We start with a very liberal requirement on $\ll$, where a node n_1 is left of a node n_2, if they have ancestors which are left of each other.

Definition 5.25 (first attempt)**.** Let $\ll$ be a partial order on nodes in an inlet graph. Further $\ll$ satisfies the following condition: if $n_1 \ll n_2$ then we have $\mathsf{succ}(n) = [\ldots, n'_1, \ldots, n'_2, \ldots]$, where $n'_1 \rightharpoonup^* n_1$ and $n'_2 \rightharpoonup^* n_2$, for some n.

We investigate consequences of this definition and show two notorious cases, which directly relate to anti-symmetry and transitivity. For this it is sufficient to consider the special case of $n_1 = n'_1$ and $n_2 = n'_2$.

First we inspect the case for anti-symmetry with two distinct nodes n_1 and n_2, where both $n_1 \ll n_2$ *and* $n_2 \ll n_1$ satisfy Definition 5.25, but $n_1 \neq n_2$.

Corollary 5.26 (anti-symmetry)**.** *Consider two nodes n_1, n_2 with $n_1 \neq n_2$, and a node n with* $\mathsf{succ}(n) = [n_1, n_2, n_1]$*. If $n_1 \ll n_2$* and *$n_2 \ll n_1$ this contradicts anti-symmetry by the following counter-example:*

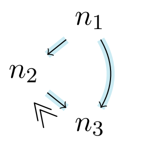
$$
\begin{tikzcd}
n \arrow[d] \arrow[dd, bend right] \arrow[dd, bend left] \\
n_2 \\
n_1
\end{tikzcd}
.
$$

Note that here $\mathsf{Pos}(n_1) = \{1, 3\}$ *and* $\mathsf{Pos}(n_2) = \{2\}$*, and* $1 <_{\mathsf{lex}} 2 <_{\mathsf{lex}} 3$*.*

Thus when fixing some order $\ll$ on nodes either $n_1 \ll n_2$ or $n_2 \ll n_1$, or n_1 and n_2 are incomparable.

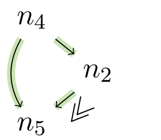
Now we consider the second case. The key observation here is that a node n_3 can be a successor *and* a "neighbour" of another node n_2, i.e. $n_2 \ll n_3$ satisfy Definition 5.25, *and* $n_2 \rightharpoonup n_3$.

Corollary 5.27. *By* $\mathsf{succ}(n_1) = [n_2, n_3]$*, we allow* $n_2 \ll n_3$ *and* $n_2 \rightharpoonup n_3$ *as witnessed by the following example:*



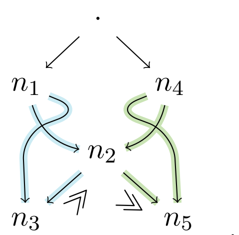
Symmetrically a node n_2 can be a ancestor *and* a "neighbour" of another node n_5, i.e. $n_5 \ll n_2$ satisfy Definition 5.25, *and* $n_2 \rightharpoonup n_5$.

Corollary 5.28. *By* $\mathsf{succ}(n_4) = [n_5, n_2]$*, we allow* $n_5 \ll n_2$*, and* $n_2 \rightharpoonup n_5$ *as witnessed by the following example:*



These two corollaries show that a node can be either an successor or an ancestor and still be left of a neighbouring node. That is, there is no relation between the successor and the $\ll$-relation. We combine these two observations and investigate transitivity. For the combination note that the node numbers are kept from the above examples and additionally the colors may aid.

Corollary 5.29 (transitivity)**.** *If* $n_2 \ll n_3$ *and* $n_5 \ll n_2$ *then by transitivity of* $\ll$ *we get* $n_5 \ll n_3$*. Thus by Definition 5.25,* $\mathsf{succ}(n_2) = [\ldots, n_5, \ldots, n_3, \ldots]$ *has to hold. The following counter-example contradicts this as* $\mathsf{succ}(n_2) = [n_3, n_5]$*:*



As before when fixing some order $\ll$ on nodes either $n_3 \ll n_5$ or $n_5 \ll n_2$ are prohibited. From this we conclude that we need to exclude situations, where a nodes is at the same time a neighbour and reachable from another node. That is, we only compare nodes which are parallel.

For a formal description of "left of", we employ positions (cf. Definition 3.11). For a inlet graph G with inlets_G, the base case is adapted slightly: $\mathsf{Pos}_G(n) := \{i\}$ if n is on ith position in inlets_G.

Definition 5.30. Let p and q be positions. Then p is left—or above—of q, if $p = p_1 \cdots p_k <_{\mathsf{lex}} q_1 \cdots q_l = q$, i.e. $p_i = q_i$ for $1 \leqslant i \leqslant j$ and $j = k < l$ or $p_j < q_j$.

We now have to extend this comparison from positions to nodes. This entails on the one hand an intra-node comparison which finds the smallest position within a node. Then an inter-node comparison comparing the smallest positions of two nodes. This solves the problem detected in Corollary 5.26—by fixing one as the primary. We solve the problem described in Corollary 5.29 by restricting the comparison to parallel nodes.

Definition 5.31. Let G be a inlet graph, $n, n' \in G$, and suppose n and n' are parallel. We define a partial order $\ll_G$ on the parallel nodes in G. Further suppose $p \in \mathsf{Pos}(n)$ is minimal wrt. $<_{\mathsf{lex}}$ and $q \in \mathsf{Pos}(n')$ is minimal wrt. $<_{\mathsf{lex}}$. Then $n \ll_G n'$ if $p <_{\mathsf{lex}} q$.

For proving transitivity of $\sqsubseteq_{\mathsf{emb}}$, we require the following lemma on $\ll$.

Lemma 5.32. *Let G be a inlet graph. For two distinct nodes n_1, n_2 in G, $\neg(n_1 \ll n_2)$ implies $(n_1 \rightharpoonup^+ n_2) \vee (n_2 \rightharpoonup^+ n_1)$ or $(n_2 \ll n_1)$.*

Proof. By definition if $n_1 \ll n_2$ then n_1, n_2 are mutually unreachable, and by totality of $\ll$ on parallel nodes. $\square$

We develop Definition 5.21 further to the final version of embedding.

Definition 5.33 (final)**.** Let $\sqsubseteq$ be a precedence. We say that S *embeds* T, denoted as $S \sqsupseteq_{\mathsf{emb}} T$, if there exists a partial, surjective function $m\colon S \to T$ such that for all nodes s in the domain of m:

(i) $\mathsf{Top}_T(m(s)) \sqsubseteq \mathsf{Top}_S(s)$, and

(ii) $m(s) \rightharpoonup_T m(s')$ implies $s \rightharpoonup_S^+ n'$ for some $n' \in \{n \mid m(n) = m(s')\}$, and

(iii) $m(s) \ll_T m(s')$ implies either

  (a) that none of the nodes in the pre-image of $m(s')$ is parallel to s, or

  (b) there exists $n' \in \{n \mid m(n) = m(s')\}$ such that $s \ll_S n'$.

We next illustrate the definition with a couple of examples. Recall our original motivating Example 5.23. With the final definition of embedding, the two term graphs are not mutually embedded per se—embedding now depends on $\sqsubseteq$.

Example 5.34. For the following two term graphs we have the following morphism: $m(1) = A$, $m(2) = B$, $m(3) = C$, and $m(4) = D$. Here we have $B \ll C$ but 2 and 3 are not parallel.

$$\begin{tikzcd}
 & \mathsf{f} : 1 \arrow[dl] \arrow[dd, bend left] & & & & \mathsf{f} : A \arrow[dl] \arrow[dr] & \\
\mathsf{g} : 2 \arrow[dr] & & \sqsupseteq_{\mathsf{emb}} & & \mathsf{g} : B \arrow[dr] & & \mathsf{g} : C \arrow[dl] \\
 & \mathsf{g} : 3 \arrow[d] & & & & \mathsf{a} : D & \\
 & \mathsf{a} : 4 & & & & &
\end{tikzcd}$$

Still even with $\ll$ the two graphs below are mutually embedded. Here we have neither $2 \ll 3$ nor $B \ll C$, so Condition *(iii)* holds trivially in both directions.

$$\begin{tikzcd}
\mathsf{f} : 1 \arrow[dr] \arrow[dd, bend right] & & & & \mathsf{f} : A \arrow[dl] \arrow[dd, bend left] \\
& \mathsf{g} : 2 \arrow[dl] & \begin{array}{c} \sqsupseteq_{\mathsf{emb}} \\ \sqsubseteq_{\mathsf{emb}} \end{array} & \mathsf{g} : B \arrow[dr] & \\
\mathsf{a} : 3 & & & & \mathsf{a} : C
\end{tikzcd}$$

One of the main challenges of proving transitivity is the non-injectivity of morphisms. For the proof we introduce the following notation. Given a morphism $m_{XY} : X \to Y$. Then $m_{XY}^{-1} : Y \to \mathcal{P}(X)$, and

$$m_{XY}^{-1}(y) = \{x \in X \mid m_{XY}(x) = y\} \quad .$$

By definition then

$$m_{XY}^{-1}(m_{XY}(x)) = \{x' \in X \mid m_{XY}(x') = m_{XY}(x)\} \quad .$$

Lemma 5.35. *The order $\sqsubseteq_{\mathsf{emb}}$ is transitive.*

Proof. Assume term graphs S, T, U. We show that $S \sqsupseteq_{\mathsf{emb}} T$ (assumption $\mathbb{A}_{\mathsf{S} \sqsupseteq_{\mathsf{emb}} \mathsf{T}}$) and $T \sqsupseteq_{\mathsf{emb}} U$ (assumption $\mathbb{A}_{\mathsf{T} \sqsupseteq_{\mathsf{emb}} \mathsf{U}}$) imply $S \sqsupseteq_{\mathsf{emb}} U$. Therefore we construct a morphism $\mathsf{m}_{\mathsf{SU}} : S \to U$ and show that m_{SU} fulfills the conditions in Definition 5.33.

By $\mathbb{A}_{\mathsf{T} \sqsupseteq_{\mathsf{emb}} \mathsf{U}}$ we have a surjective morphism $\mathsf{m}_{\mathsf{TU}} : T \to U$ and by $\mathbb{A}_{\mathsf{S} \sqsupseteq_{\mathsf{emb}} \mathsf{T}}$ we have a surjective morphism $\mathsf{m}_{\mathsf{ST}} : S \to T$. We set $\mathsf{m}_{\mathsf{SU}}(s) := \mathsf{m}_{\mathsf{TU}}(\mathsf{m}_{\mathsf{ST}}(s))$. By surjectivity of m_{ST} and m_{TU}, also m_{SU} is surjective.

We show next that m_{SU} fulfills Definition 5.33*(i)*:

$$\mathsf{Top}_S(s) \sqsupseteq \mathsf{Top}_U(\mathsf{m}_{\mathsf{SU}}(s))$$

By definition $\mathsf{Top}_U(\mathsf{m}_{\mathsf{SU}}(s)) = \mathsf{Top}_U(\mathsf{m}_{\mathsf{TU}}(\mathsf{m}_{\mathsf{ST}}(s)))$. By $\mathbb{A}_{\mathsf{S} \sqsupseteq_{\mathsf{emb}} \mathsf{T}}$, $\mathsf{Top}_S(s) \sqsupseteq \mathsf{Top}_T(\mathsf{m}_{\mathsf{ST}}(s))$ and by $\mathbb{A}_{\mathsf{T} \sqsupseteq_{\mathsf{emb}} \mathsf{U}}$, for all $\mathsf{m}_{\mathsf{TU}}(t) \in U$ we have $\mathsf{Top}_T(t) \sqsupseteq \mathsf{Top}_U(\mathsf{m}_{\mathsf{TU}}(t))$, in particular for $t = \mathsf{m}_{\mathsf{ST}}(s)$, hence $\mathsf{Top}_T(\mathsf{m}_{\mathsf{ST}}(s)) \sqsupseteq \mathsf{Top}_U(\mathsf{m}_{\mathsf{TU}}(\mathsf{m}_{\mathsf{ST}}(s)))$. By transitivity of $\sqsupseteq$ we conclude Condition *(i)*.

Next, we need to show that m_{SU} fulfills Definition 5.33*(ii)*:

$$\text{if } \mathsf{m}_{\mathsf{SU}}(s) \rightharpoonup_U \mathsf{m}_{\mathsf{SU}}(s') \text{ then } s \rightharpoonup_S^+ n \text{ where } n \in \mathsf{m}_{\mathsf{SU}}^{-1}(\mathsf{m}_{\mathsf{SU}}(s'))$$

By definition $\mathsf{m}_{\mathsf{TU}}(\mathsf{m}_{\mathsf{ST}}(s)) \rightharpoonup_U \mathsf{m}_{\mathsf{TU}}(\mathsf{m}_{\mathsf{SU}}(s'))$ and by $\mathbb{A}_{\mathsf{T} \sqsupseteq_{\mathsf{emb}} \mathsf{U}}$, Condition *(ii)*, we get $\mathsf{m}_{\mathsf{ST}}(s) \rightharpoonup_T^l n_2$, where $n_2 \in \mathsf{m}_{\mathsf{TU}}^{-1}(\mathsf{m}_{\mathsf{TU}}(\mathsf{m}_{\mathsf{ST}}(s')))$ for $l \geq 1$. We show $s \rightharpoonup_S^+ n$ where $n \in \mathsf{m}_{\mathsf{SU}}^{-1}(\mathsf{m}_{\mathsf{SU}}(s'))$ by induction on l.

Base Case. $l = 1$. By surjectivity we know there is a $n_3 \in S$ such that $\mathsf{m}_{\mathsf{ST}}(n_3) = n_2$. Then by $\mathbb{A}_{\mathsf{S} \sqsupseteq_{\mathsf{emb}} \mathsf{T}}$, Condition *(ii)*, we get $s \rightharpoonup_S^k n_4$ for $n_4 \in \mathsf{m}_{\mathsf{ST}}^{-1}(\mathsf{m}_{\mathsf{ST}}(n_3))$ and $k \geqslant 1$. Then by $\mathsf{m}_{\mathsf{ST}}(n_4) = \mathsf{m}_{\mathsf{ST}}(n_3) = n_2$ and $\mathsf{m}_{\mathsf{TU}}(n_2) = \mathsf{m}_{\mathsf{TU}}(\mathsf{m}_{\mathsf{ST}}(s'))$ we conclude $\mathsf{m}_{\mathsf{TU}}(\mathsf{m}_{\mathsf{ST}}(n_4)) = \mathsf{m}_{\mathsf{TU}}(\mathsf{m}_{\mathsf{ST}}(s'))$ and thus $n_4 \in \mathsf{m}_{\mathsf{SU}}^{-1}(\mathsf{m}_{\mathsf{SU}}(s'))$.

Step Case. $\mathsf{m}_{\mathsf{ST}}(s) \rightharpoonup_T^l n_3 \rightharpoonup_T n_2$. By induction hypothesis we get $s \rightharpoonup_T^+ n_4$ for $n_4 \in \mathsf{m}_{\mathsf{ST}}^{-1}(n_3)$. By surjectivity we have a node n_5 such that $\mathsf{m}_{\mathsf{ST}}(n_5) = n_2$. Combining these two facts we get $\mathsf{m}_{\mathsf{ST}}(n_3) = \mathsf{m}_{\mathsf{ST}}(n_4) \rightharpoonup_T \mathsf{m}_{\mathsf{ST}}(n_5)$. With the same reasoning as in the base case we get $n_4 \rightharpoonup_S^+ n_6$ where $n_6 \in \mathsf{m}_{\mathsf{ST}}^{-1}(\mathsf{m}_{\mathsf{ST}}(n_5))$. Hence $\mathsf{m}_{\mathsf{ST}}(m_6) = \mathsf{m}_{\mathsf{ST}}(n_5)$, with $\mathsf{m}_{\mathsf{ST}}(n_5) = n_2$ and $\mathsf{m}_{\mathsf{TU}}(n_2) = \mathsf{m}_{\mathsf{TU}}(\mathsf{m}_{\mathsf{ST}}(s'))$, we get $n_6 \in \mathsf{m}_{\mathsf{SU}}^{-1}(\mathsf{m}_{\mathsf{SU}}(s'))$.

Finally, we need to show m_{SU} fulfills Definition 5.33*(iii)*. Therefore, we state Condition *(a)* more formally: none of the nodes in the pre-image of $m(s')$, i.e. $m^{-1}(m(s'))$, is parallel to s, i.e. none of the nodes in $m^{-1}(m(s'))$ is mutually unreachable from/to s, i.e. $\forall n \in m^{-1}(m(s'))$ either $n \rightharpoonup_S^+ s$ or $s \rightharpoonup_S^+ n$. Hence we have to show:

$$\text{if } \mathsf{m}_{\mathsf{SU}}(s) \ll \mathsf{m}_{\mathsf{SU}}(s') \text{ then either} \tag{α}$$

$$\forall n \in \mathsf{m}_{\mathsf{SU}}^{-1}(\mathsf{m}_{\mathsf{SU}}(s')) \text{ either } n \rightharpoonup_S^+ s \text{ or } s \rightharpoonup_S^+ n, \text{ or} \tag{β}$$

$$\exists n \in \mathsf{m}_{\mathsf{SU}}^{-1}(\mathsf{m}_{\mathsf{SU}}(s')) \text{ with } s \ll n \quad . \tag{γ}$$

We have to show $\alpha \Rightarrow \beta \vee \gamma$. Therefore we show that $\alpha \Rightarrow \neg\beta \Rightarrow \gamma$.

For $\neg\beta$ we assume there exists a $n_2 \in \mathsf{m}_{\mathsf{SU}}^{-1}(\mathsf{m}_{\mathsf{SU}}(s'))$ so that $\neg(n_2 \rightharpoonup_S^+ s)$ and $\neg(s \rightharpoonup_S^+ n_2)$. Then by Lemma 5.32 we know $s \ll n_2$ or $n_2 \ll s$. The former shows γ, for the latter we derive a contradiction.

By α and definition we know $\mathsf{m}_{\mathsf{TU}}(\mathsf{m}_{\mathsf{ST}}((s)) \ll \mathsf{m}_{\mathsf{TU}}(\mathsf{m}_{\mathsf{ST}}((s'))$ and then by $\mathbb{A}_{\mathsf{T} \sqsupseteq_{\mathsf{emb}} \mathsf{U}}$ we can conclude either $\forall n_3 \in \mathsf{m}_{\mathsf{TU}}^{-1}(\mathsf{m}_{\mathsf{SU}}(s'))$ either $n_3 \rightharpoonup_S^+ \mathsf{m}_{\mathsf{ST}}(s)$ or $\mathsf{m}_{\mathsf{ST}}(s) \rightharpoonup_S^+ n_3$, or $\exists n_3 \in \mathsf{m}_{\mathsf{TU}}^{-1}(\mathsf{m}_{\mathsf{SU}}(s'))$ with $\mathsf{m}_{\mathsf{ST}}(s) \ll n_3$. By surjectivity we know that there is a n_4 such that $\mathsf{m}_{\mathsf{ST}}(n_4) = n_3$, and $\mathsf{m}_{\mathsf{TU}}(n_3) = \mathsf{m}_{\mathsf{SU}}(s') = n_2$.

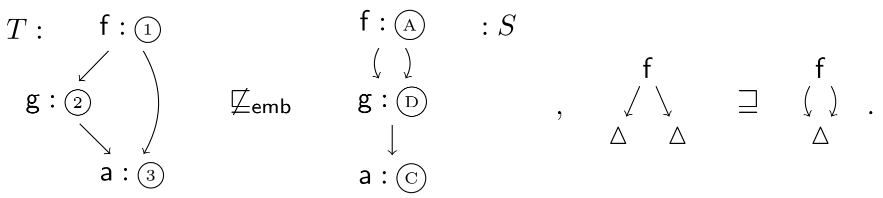
- For $\mathsf{m}_{\mathsf{ST}}(s) \rightharpoonup_S^+ \mathsf{m}_{\mathsf{ST}}(n_4)$ by Condition *(ii)* we have $s \rightharpoonup_S^+ n_5$ and $n_5 \in \mathsf{m}_{\mathsf{ST}}^{-1}(\mathsf{m}_{\mathsf{ST}}(n_4))$. Hence $\mathsf{m}_{\mathsf{ST}}(m_5) = \mathsf{m}_{\mathsf{ST}}(n_4) = n_3$, and $\mathsf{m}_{\mathsf{TU}}(\mathsf{m}_{\mathsf{ST}}(n_5)) = \mathsf{m}_{\mathsf{SU}}(s') = n_2$. $\lightning$ to $\neg\beta$.

- For $\mathsf{m}_{\mathsf{ST}}(n_4) \rightharpoonup_S^+ \mathsf{m}_{\mathsf{ST}}(s)$ by Condition *(ii)* we have $n_4 \rightharpoonup_S^+ n_6$ and $n_6 \in \mathsf{m}_{\mathsf{ST}}^{-1}(\mathsf{m}_{\mathsf{ST}}(s))$. If $n_6 = s$ we have $\lightning$ to $\neg\beta$. If $n_6 \neq s$ we have $\neg(n_3 \rightharpoonup_S^+ s)$ but by Lemma 5.32 we also have $\lightning$ to $\neg\beta$.

- For $\mathsf{m}_{\mathsf{ST}}(s) \ll n_3$ we use $\mathbb{A}_{\mathsf{S} \sqsupseteq_{\mathsf{emb}} \mathsf{T}}$ to derive a contradiction to α.

$\square$

We conclude this section with a comparison between the embedding relation for terms and our embedding relation for term graphs. Not unexpectedly the connection is very weak: $\mathsf{term}(T) \sqsubseteq_{\mathsf{emb}} \mathsf{term}(S)$ does not imply $T \sqsubseteq_{\mathsf{emb}} S$. As a counter example consider the embedding $\mathsf{term}(T) = \mathsf{f}(\mathsf{g}(\mathsf{a}), \mathsf{a}) \sqsubseteq_{\mathsf{emb}} \mathsf{f}(\mathsf{g}(\mathsf{a}), \mathsf{g}(\mathsf{a})) = \mathsf{term}(S)$. For S the successors may be shared:



Due to the order on Tops on the right we have $T \not\sqsubseteq_{\mathsf{emb}} S$. For a different representation of $\mathsf{term}(S)$, i.e. if node D were not shared, the embedding of term graphs would be possible. The reverse, $T \sqsubseteq_{\mathsf{emb}} S$ implies $\mathsf{term}(T) \sqsubseteq_{\mathsf{emb}} \mathsf{term}(S)$, does not hold either. This is easily seen as Tops with function symbols of larger arity can be smaller in the precedence Tops with function symbols of smaller arity.

5.3 Proof

Now we can move on to the main proof of this chapter: Kruskal's Tree Theorem [19] for term graphs. It closely follows the proof for the term setting in [22], which in turn follows the minimal bad sequence argument of Nash-Williams [24]: assuming the existence of a minimal "bad" infinite sequence, an even smaller "bad" infinite sequence is constructed—contradicting minimality.

The most important insight concerns the arguments of a term graph—or rather *the* argument. For a term structure we have several sub-terms as arguments. For a term graph structure it is beneficial to regard the arguments as only a single argument graph. This preserves sharing of nodes. Moreover a single argument simplifies the proof as extending the order to sequences, Higman's Lemma [15], can be omitted.

Theorem 5.36. *If $\sqsubseteq$ is a wqo on $\mathsf{Tops}(\mathcal{F})$, then $\sqsubseteq_{\mathsf{emb}}$ is a wqo on ground term graphs over $\mathcal{F}$.*

Proof. By definition $\sqsubseteq_{\mathsf{emb}}$ is wqo if for every infinite sequence exist indices i, j with $1 \leqslant i < j$ such that $T_i \sqsubseteq_{\mathsf{emb}} T_j$ for term graphs T_i, T_j. That is, every infinite sequence is good. We construct a minimal bad sequence of term graphs $\mathbf{T}$ in the following way. Assume we picked (canonical) term graphs $T_1, \ldots, T_{n-1}$. We pick the (canonical) term graph T_n, which is minimal with respect to its size $|T_n|$, such that there are bad sequences that start with $T_1, \ldots, T_n$.

Let G_i be the argument graph of the i^{th} term graph T_i. We collect in G the arguments of all term graphs in $\mathbf{T}$, i.e. $G = \bigcup_{i \geqslant 1} G_i$.

Now we first prove that $\sqsubseteq_{\mathsf{emb}}$ is a wqo on G. For a contradiction, we assume G admits a bad sequence $\mathbf{H}$. We pick some $G_k \in G$ with $k \geqslant 1$. In G' we collect all argument graphs up to G_k, i.e. $G' = \bigcup_{i \geqslant 1}^{k} G_i$. The set G' is finite, hence there exists an index $l > 1$, such that for all H_i with $i \geqslant l$ we have that $H_i \in G$ but $H_i \notin G'$. We write $\mathbf{H}_{\geqslant l}$ for the sequence $\mathbf{H}$ starting at index l. Now consider the sequence $T_1, \ldots, T_{k-1}, G_k, \mathbf{H}_{\geqslant l}$. By minimality of $\mathbf{T}$ this is a good sequence. So we try to find $H_i \sqsubseteq_{\mathsf{emb}} H_j$. We distinguish on i, j:

Case. $\underbrace{T_1, \ldots, T_{k-1}}_{i,j}, G_k, \mathbf{H}_{\geqslant l}$. For $1 \leqslant i < j \leqslant k-1$, we have $H_i = T_i \sqsubseteq_{\mathsf{emb}} T_j = H_j$, which contradicts the badness of $\mathbf{T}$.

Case. $\underbrace{T_1, \ldots, T_{k-1}}_{i}, \underbrace{G_k}_{j}, \mathbf{H}_{\geqslant l}$. For $1 \leqslant i \leqslant k-1$ and $j = k$, we have $H_i = T_i \sqsubseteq_{\mathsf{emb}} G_k = H_j$ and $G_k \sqsubseteq_{\mathsf{emb}} T_k$, but then, by transitivity, $T_i \sqsubseteq_{\mathsf{emb}} T_j$, which contradicts the badness of $\mathbf{T}$.

Case. $\underbrace{T_1, \ldots, T_{k-1}}_{i}, G_k, \underbrace{\mathbf{H}_{\geqslant l}}_{j}$. For $1 \leqslant i \leqslant k-1$ and $j \geqslant l$, we have $H_j \notin G'$ by construction, but $H_j = G_m \sqsubseteq_{\mathsf{emb}} T_m, m > k$ and $H_i = T_i \sqsubseteq_{\mathsf{emb}} G_m = H_j$ hence by transitivity $T_i \sqsubseteq_{\mathsf{emb}} T_m$, which contradicts the badness of $\mathbf{T}$.

Case. $T_1, \ldots, T_{k-1}, \underbrace{G_k, \mathbf{H}_{\geqslant l}}_{i,j}$. Hence for some $1 \leqslant i < j$, where $i, j \notin \{2, \ldots, l-1\}$, we have $H_i \sqsubseteq_{\mathsf{emb}} H_j$, which contradicts the badness of $\mathbf{H}$.

We conclude $\mathbf{H}$ is a good sequence and $\sqsubseteq_{\mathsf{emb}}$ is wqo on G.

By assumption $\sqsubseteq$ is a wqo on $\mathsf{Tops}(\mathcal{F})$. Let $\mathbf{f}$ be the sequence of Tops of $\mathbf{T}$. By Lemma 2.6 we know that $\mathbf{f}$ contains a chain $\mathbf{f}_\phi$, i.e. $f_{\phi_i} \sqsubseteq f_{\phi_{i+1}}$ for all $i \geqslant 1$. We proved $\sqsubseteq_{\mathsf{emb}}$ to be a wqo on G. Hence we have $G_{\phi_i} \sqsubseteq_{\mathsf{emb}} G_{\phi_j}$ for some $1 \leqslant i < j$.

It remains to be shown, that $f_{\phi_i} \sqsubseteq f_{\phi_j}$ and $G_{\phi_i} \sqsubseteq_{\mathsf{emb}} G_{\phi_j}$ implies $T_{\phi_i} \sqsubseteq_{\mathsf{emb}} T_{\phi_j}$. The plan is the following: We have a morphism from G_{ϕ_i} to G_{ϕ_j}, and two Tops f_{ϕ_i} and f_{ϕ_j}. From that we construct a morphism from T_{ϕ_i} to T_{ϕ_j} First we construct T_{ϕ_i}, and analogously T_{ϕ_j}, from $f_{\phi_i} = (n_i, \mathsf{lab}_{f_i}, \mathsf{succ}_{f_i})$ and $G_{\phi_i} = (N_{G_i}, \mathsf{lab}_{G_i}, \mathsf{succ}_{G_i}, \mathsf{inlets}_{G_i})$. We have $n_i \notin G_{\phi_i}$, i.e. $N_{G_i} \cap \{n_i\} = \varnothing$. Then $T_{\phi_i} = (N_{T_i}, \mathsf{lab}_{T_i}, \mathsf{succ}_{T_i})$ where

- $N_{T_i} := N_{G_i} \cup \{n_i\}$,
- $\mathsf{lab}_{T_i} := \mathsf{lab}_{G_i} \cup \{\mathsf{lab}_{T_i}(n_i) = \mathsf{lab}_{f_i}(n_i)\}$, and
- $\mathsf{succ}_{T_i} := \mathsf{succ}_{G_i} \cup \{\mathsf{succ}_{T_i}(n_i) = \mathsf{inlets}_{G_i}\}$.

From $G_{\phi_i} \sqsubseteq_{\mathsf{emb}} G_{\phi_j}$, we obtain a morphism $m_G : G_{\phi_j} \to G_{\phi_i}$. We construct the morphism $m : T_{\phi_j} \to T_{\phi_i}$, where $m(n_j) = n_i$ and $m(n) = m_G(n)$ for the remaining $n \in G_{\phi_j}$. It remains to be shown that m fulfills Definition 5.33. Surjectivity of m follows directly from the surjectivity of m_G. Condition *(i)* holds for all nodes in m_G, and by $f_{\phi_i} \sqsubseteq f_{\phi_j}$ also for $\mathsf{rt}(T_{\phi_j}) = n_j$. For Condition *(ii)* we have to show: If $m(n_j) \rightharpoonup_{T_{\phi_i}} n'_i = m(n'_j)$ then $n_j \rightharpoonup^+ n'$ for some $n' \in m^{-1}(m(n'_j))$. We show the stronger $n_j \rightharpoonup^+ n'_j$ and trivially $n'_j \in m^{-1}(m(n'_j))$. By definition $n'_i \in \mathsf{inlets}_{G_i}$ and hence also $n'_i \in G_i$. By surjectivity of m_G exist $m_G(n'_j) = n'_i$. It remains to show that $n_j \rightharpoonup^+ n'_j$. By definition $n_j \rightharpoonup u_j$, where $u_j \in \mathsf{inlets}_{G_j}$. By definition of argument graph, all nodes in G_j are reachable from nodes in inlets_{G_j}, and in particular $n_j \rightharpoonup u_j \rightharpoonup^* n'_j$. For Condition *(iii)* note that $\ll$ in is not affected by constructing T_{ϕ_i} and T_{ϕ_j} as $\mathsf{Pos}(n_i) = \mathsf{Pos}(n_j) = \{\epsilon\}$.

Hence, $T_{\phi_i} \sqsubseteq_{\mathsf{emb}} T_{\phi_j}$, which contradicts the badness of $\mathbf{T}$. □

This concludes the proof and the chapter, where transferred the definitions in [28] to our formalism of term graphs. Inspired by [28] we defined an embedding relation. Then we re-proved Kruskal's Tree Theorem, but as opposed to [28], which uses an encoding to terms, we prove it directly for term graphs. Thereby it was beneficial to view the argument of a term graph again as *one* graph with inlets. In the next chapter we will use Kruskal's Tree Theorem for term graphs to prove a simplification order well-founded.

6 Termination of Term Graph Rewriting

In the previous chapter we developed an embedding relation for term graphs. Based on this embedding relation we now give a definition for simplification orders for term graphs in Section 6.1. We then prove simplification orders to be well-founded using the main result of the previous chapter. Next we define a simplification order: a lexicographic path order on term graphs. There are several challenges attached. For one we have to restrict the set of the term graphs to term graphs with only parallel nodes. On the other hand, as opposed to term rewriting, it is not sufficient to find an order on the rewrite rules. We highlight this challenge of automation in Section 6.2, where we give two simple scenarios of non-termination.

6.1 Lexicographic Path Order

As a first step we transfer the definition of simplification orders from the term rewrite setting, cf. Section 2.2, to the term graph rewriting setting. We adopt the following definition from [28, Definition 12].

Definition 6.1. Let $\sqsubseteq_{\mathsf{emb}}$ be the embedding relation induced by the underlying well-quasi ordered precedence $\sqsubseteq$. A transitive relation $\prec$ is a *simplification order*, if

(i) $\sqsubset_{\mathsf{emb}} \subseteq \prec$, and

(ii) for all S and T, if $S \sqsubseteq_{\mathsf{emb}} T$ and $T \sqsubseteq_{\mathsf{emb}} S$ then $S \not\prec T$.

Condition *(i)* directly compares to the term rewrite setting. Condition *(ii)* is required because $\sqsubseteq_{\mathsf{emb}}$ is not anti-symmetric in general—even if the underlying precedence is anti-symmetric. That is, for term graphs, $S \sqsubseteq_{\mathsf{emb}} T$ and $T \sqsubseteq_{\mathsf{emb}} S$ does not imply $S \cong T$. A direct consequence of Condition *(ii)* is that simplification orders are irreflexive. Due to Kruskal's Tree Theorem for term graphs we obtain the following result, where the proof is originally from [28, Theorem 13].

Theorem 6.2. *Every simplification order $\prec$ is well-founded.*

Proof. For $\prec$ we have by definition an embedding relation $\sqsubset_{\mathsf{emb}}$, such that $\sqsubset_{\mathsf{emb}} \subseteq \prec$. Due to Kruskal's Tree Theorem for term graphs 5.36, $\sqsubseteq_{\mathsf{emb}}$ is a wqo. We assume an infinite sequence $S_1 \succ S_2 \succ \ldots$ As $\sqsubseteq_{\mathsf{emb}}$ is a wqo, we have $S_i \sqsubseteq_{\mathsf{emb}} S_j$ for $i < j$. On the other hand we have $S_i \succ S_{i+1} \succ \ldots \succ S_j$, and by transitivity $S_i \succ S_j$. By Definition 6.1 Condition *(ii)* and $S_i \succ S_j$, not $S_i \sqsubseteq_{\mathsf{emb}} S_j$ and $S_j \sqsubseteq_{\mathsf{emb}} S_i$. Hence $S_i \sqsubset_{\mathsf{emb}} S_j$, and by assumption we have $S_i \prec S_j$. We derived a contradiction. Thus $\prec$ is well-founded. $\square$

In the next step we adapt the *lexicographic path order* (*LPO*) from term rewriting to term graph rewriting. It is natural to define LPO on inlet graphs in the setting of this thesis. The term graph case is trivial as a term graph S is an inlet graph with $\mathsf{inlets}_S = [\mathsf{rt}(S)]$.

Definition 6.3. Let $\sqsubseteq$ be a well-quasi ordered precedence. We write $\sqsubset_{\mathsf{lex}}$ for the lexicographic extension of $\sqsubset$. Let S and T be inlet graphs with $\mathsf{inlets}_S = [s_1, \ldots, s_k]$ and $\mathsf{inlets}_T = [t_1, \ldots, t_l]$, where s_i, s_j and t_i, t_j are parallel for $s_i \neq s_j$ and $t_i \neq t_j$. Then $T <_{\mathsf{lpo}} S$ if one of the following holds

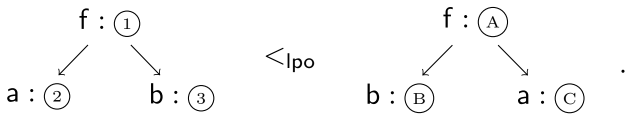
(i) $T \leqslant_{\mathsf{lpo}} S\lceil[s_{i_1}, \ldots, s_{i_{k'}}]$ for some $1 \leqslant i_1 < \ldots < i_{k'} \leqslant k$, or

(ii) $[\mathsf{Top}(t_1), \ldots, \mathsf{Top}(t_l)] \sqsubset_{\mathsf{lex}} [\mathsf{Top}(s_1), \ldots, \mathsf{Top}(s_k)]$ and $\mathsf{arg}(T) <_{\mathsf{lpo}} S$, or

(iii) $[\mathsf{Top}(t_1), \ldots, \mathsf{Top}(t_l)] = [\mathsf{Top}(s_1), \ldots, \mathsf{Top}(s_k)]$ and $\mathsf{arg}(T) <_{\mathsf{lpo}} \mathsf{arg}(S)$.

The next examples demonstrate our LPO. We start with Example 5.23, which motivated the importance of ordering successor nodes.

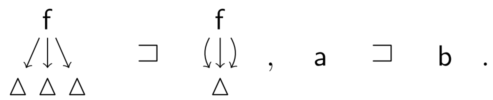
Example 6.4. Given the precedence $\mathsf{a} \sqsubset \mathsf{b}$ we can compare the two term graphs:



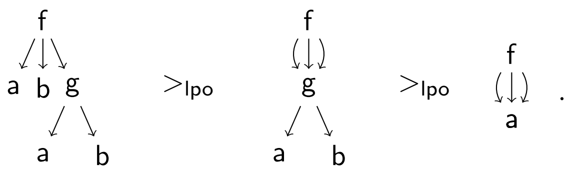
To compare the term graphs with $<_{\mathsf{lpo}}$ we first use *(iii)* and compare the argument graphs. Then we compare their respective inlets lexicographically, i.e. $[\mathsf{Top}(②), \mathsf{Top}(③)] \sqsubset_{\mathsf{lex}} [\mathsf{Top}(Ⓑ), \mathsf{Top}(Ⓒ)]$ using *(ii)*.

Next recall Example 5.1—Toyama's example. It is non-terminating in the term rewrite setting, and served as motivation example in Chapter 5.

Example 6.5. Given the following precedence:



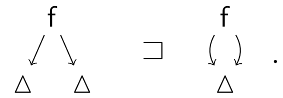
We can compare the graphs in the following rewrite sequence with $>_{\mathsf{lpo}}$ as follows:



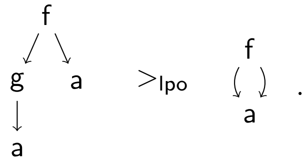
For the first step we use *(ii)* to begin. To compare the argument we then can project the corresponding sub-graph with *(i)*. For the second step we use case *(iii)* followed by *(i)* again.

As a final example consider the following two term graphs.

Example 6.6. Given the following precedence:



We can compare the two term graphs in the following by first using *(ii)* and then *(i)*.



To prove that $<_{\mathsf{lpo}}$ is a simplification order, we have to prove that $<_{\mathsf{lpo}}$ contains $\sqsubset_{\mathsf{emb}}$. Thereby it important to note that $<_{\mathsf{lpo}}$ requires that nodes are parallel within inlets. That means, we can inductively step through an inlet graph with inlets forming a level in the inlet graph.

Theorem 6.7. *The order $<_{\mathsf{lpo}}$ is a simplification order.*

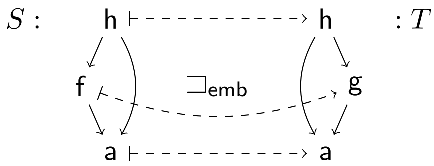
Proof. To show that $<_{\mathsf{lpo}}$ is a simplification order, we need to show that it satisfies both conditions of show Definition 6.1. We start by showing Condition *(i)*: $\sqsubset_{\mathsf{emb}} \subseteq <_{\mathsf{lpo}}$. For inlet graphs S and T with $\mathsf{inlets}_S = [s_1, \ldots, s_k]$ and $\mathsf{inlets}_T = [t_1, \ldots, t_l]$, we have to show that $S \sqsupset_{\mathsf{emb}} T$ implies $S >_{\mathsf{lpo}} T$. We continue by induction on $|S| + |T|$. By $S \sqsupset_{\mathsf{emb}} T$ we know there is a morphism $m : S \to T$ satisfying the conditions in Definition 5.33 denoted by $m_{\textit{(i)-(iii)}}$. By surjectivity of m, and $m_{\textit{(ii)}}$, we have $s_i \rightharpoonup^{+} s'_j$ such that $m(s'_j) = t_j$ for $1 \leq j \leq l$. By Definition 6.3*(i)* it suffices to show $S{\restriction}[s'_1, \ldots, s'_l] \geqslant_{\mathsf{lpo}} T{\restriction}[t_1, \ldots, t_l]$. Now by $m_{\textit{(i)}}$ and $m_{\textit{(iii)}}$ we know $[\mathsf{Top}(t_1), \ldots, \mathsf{Top}(t_l)] \sqsubseteq_{\mathsf{lex}} [\mathsf{Top}(s'_1), \ldots, \mathsf{Top}(s'_l)]$. Hence, by *(ii)*, *(iii)* it suffices to show $\mathsf{arg}(S{\restriction}[s'_1, \ldots, s'_l]) >_{\mathsf{lpo}} \mathsf{arg}(T{\restriction}[t_1, \ldots, t_l])$. By definition we have to show $S{\restriction}\mathsf{succ}(s'_1) \cdots \mathsf{succ}(s'_l) >_{\mathsf{lpo}} T{\restriction}\mathsf{succ}(t_1) \cdots \mathsf{succ}(t_l)$. As all successor nodes in S, T are parallel, we conclude the proof by induction hypothesis.

For Condition *(ii)* it suffices to observe that for term graphs with parallel nodes $S \sqsubseteq_{\mathsf{emb}} T$ and $T \sqsubseteq_{\mathsf{emb}} S$ implies $S \cong T$, and as $<_{\mathsf{lpo}}$ is irreflexive $S \not<_{\mathsf{lpo}} T$ and vice versa. ☐

Alternatively we potentially could have used the results in [14] to prove the well-foundedness of $<_{\mathsf{lpo}}$. There the presented techniques are applied to graphs. However, the notion of graphs is much more liberal than our definition, and thus is not immediately transferable.

It is important to note that our definition of $<_{\mathsf{lpo}}$ does only work on a special shape of inlet graphs: inlet graphs which only have parallel nodes in the successors, i.e. for all $n \in G$ and for all $n_i, n_j \in \mathsf{succ}(n)$, n_i and n_j are parallel. This restriction is motivated by the following example.

Example 6.8. Given the precedence $\mathsf{f} \sqsupset \mathsf{g}$ and $\mathsf{a} \sqsupset \mathsf{f}$. For S and T below we have $S \sqsupset_{\mathsf{emb}} T$, but $S \not>_{\mathsf{lpo}} T$.

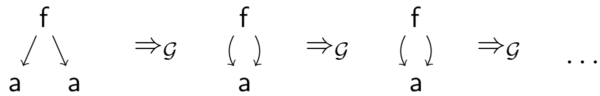


In the recursive case we have to compare the inlets $[\mathsf{f}, \mathsf{a}]$ with the inlets $[\mathsf{a}, \mathsf{g}]$—but $\mathsf{a} \sqsupset \mathsf{f}$. On a side note this may be possible with a multi-set comparison.

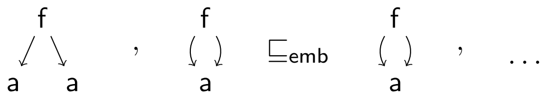
Finally consider the embedding relation again. As opposed to the term rewrite setting in the graph rewrite setting it is not sufficient to find an order on the rules to conclude termination of every instance.

Example 6.9. Consider a term graph rewrite system $\mathcal{G}$ with the following rule on the left, which we can compare with $\sqsupset_{\mathsf{emb}}$ on the right:

But we still may get an infinite rewrite sequence:



It is important to note that this infinite rewrite sequence is not bad wrt. $\sqsubseteq_{\mathsf{emb}}$ as we can see next:



This problem is not caused by our definition of embedding and also occurs in [28]. Rather the reason is that from an order on the rules, we cannot conclude an order on all the rewrite steps. Still we can follow Plump [28, Theorem 25] and find an order on every rewrite step—and not solely the rewrite rules.

Here we restrict our attention to $\Rightarrow_{\mathcal{G}}$ and do not incorporate any explicit form of collapsing. This is easily justified as $\succ$, $\succcurlyeq$, $\succ^{!}$, and $\succ^{!+}$ are all contained in embedding and hence in $\geqslant_{\mathsf{lpo}}$.

Theorem 6.10. *Given a simplification order $>_{\mathsf{lpo}}$. Then $\Rightarrow_{\mathcal{G}}$ is terminating if $S \Rightarrow T$ implies $S >_{\mathsf{lpo}} T$ and all ground term graphs S and T.*

Proof. Assume an infinite sequence $S_1 \Rightarrow_{\mathcal{G}} S_2 \Rightarrow_{\mathcal{G}} \dots$ By assumption we get $S_1 >_{\mathsf{lpo}} S_2 >_{\mathsf{lpo}} S_3 >_{\mathsf{lpo}} S_4 \dots$ But $>_{\mathsf{lpo}}$ is a simplification order, and we have $\sqsupset_{\mathsf{emb}} \subseteq >_{\mathsf{lpo}}$. Hence we can select $S_{i_1} >_{\mathsf{lpo}} S_{i_2} >_{\mathsf{lpo}} S_{i_3} >_{\mathsf{lpo}} S_{i_4} \dots$ As $>_{\mathsf{lpo}}$ is well-founded, we derive a contradiction. $\square$

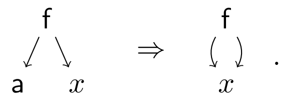
Unfortunately we cannot conclude an order on every rewrite step from an order on the rewrite rules. Here the problem lies in the definition of context and substitution for graph rewriting—two topics we will briefly touch in the next section.

6.2 Non-Termination

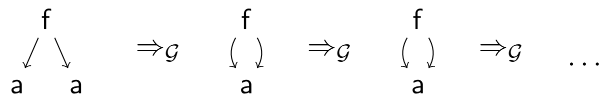
In this section we present some rather straight-forward results on non-termination. Here we loosely relate two scenarios of non-termination to closure under substitution and closure under context. Both concepts, context and substitution, are not so straight-forward for term graph rewriting.

We start with substitution and the notion of instance. In the term rewrite setting, if a right-hand side of a rule is an instance of a left-hand side we trivially have non-termination. This transfers to the graph rewrite setting—but additionally collapsing has to be taken into account.

Example 6.11. Consider the following graph rewrite system $\mathcal{G}$, which is very similar to Example 6.9:



We can easily find a non-terminating rewrite sequence, because the rewrite rule expresses collapsing for the following term graph:



Intuitively we can find an infinite rewrite sequence if we find some instance of a term graph rewrite rule, such that the instance of the left-hand side collapses to the instance of the right-hand side. This explains why we have to compare all ground instances in Theorem 6.10. We define a mapping from variables in a term graph to ground term graphs.

Definition 6.12. Let $\sigma : \mathcal{V}\mathsf{ar}(S) \to \mathcal{TG}(\mathcal{F}, \mathcal{V})$. For $\{n_1, \dots, n_k\} \in \mathcal{V}\mathsf{ar}(S) \cap \mathsf{dom}(\sigma)$ we define $S\sigma$ as S_{k+1} inductively:

$$
\begin{aligned}
S_1 &:= S \\
S_{i+1} &:= (S_i \oplus \sigma(n_i))[\mathsf{rt}(\sigma(n_i)) \leftarrow n_i]
\end{aligned}
$$

Here we assume $N_{S_i} \cap N_{\sigma(n_i)} = \varnothing$ and delete n_i from S_{i+1}.

We then arrive at the following straight-forward lemma, which indicates non termination for a term graph rewrite system.

Lemma 6.13. *If for $L \Rightarrow R \in \mathcal{G}$ and $\sigma : \mathcal{V}\mathsf{ar}(L) \to \mathcal{TG}(\mathcal{F}, \mathcal{V})$ we have $L\sigma \succcurlyeq R\sigma$, then there exists an infinite rewrite sequence.*

Proof. We have $m_1 : L \to_{\mathcal{V}} L\sigma$ and $m_2 : L\sigma \to R\sigma$. By transitivity, and $\mathcal{V}\mathsf{ar}(R) \subseteq \mathcal{V}\mathsf{ar}(L)$, we have a morphism $m : L \to_{\mathcal{V}} R\sigma$, hence L matches $R\sigma$ giving rise to the infinite rewrite sequence $L\sigma \Rightarrow R\sigma \Rightarrow R\sigma \ldots$ ☐

With respect to the size of S and T, we observe that $S \succ T$ guarantees $|S| > |T|$, as $m : S \to T$ is injective. But even if $|L| > |R|$, we cannot be conclude that for $S \Rightarrow_{\mathcal{G}} T$ we have $|S| > |T|$. For every step we have only a constant growth of $|T|$, as we can see in Lemma 3.33, but no determined decrease. We conclude that the notion of instance is not as clear cut for term graphs as for terms. If S collapses to T, $S \succcurlyeq T$, is T then an instance of S? And vice versa—is T and instance of S?

Similarly the notion of context is not clear for term graph rewriting. That is, a node can at the same time be part of the context and of the matched left-hand side. Put differently, the node is reachable from the redex node *and* a node, which is not necessarily above the redex node.

Example 6.14. Consider the following GRS $\mathcal{G}$, where all rules $L \Rightarrow R$ are left-linear, $\mathcal{V}\mathsf{ar}(L) = \mathcal{V}\mathsf{ar}(R)$, and $|L| > |R|$.

$$
\begin{tikzcd}[row sep=small]
\mathsf{f} \arrow[d] \\
\mathsf{g} \arrow[d] \\
x
\end{tikzcd}
\Rightarrow
\begin{tikzcd}[row sep=small]
\mathsf{f} \arrow[d] \\
x
\end{tikzcd}
\quad , \quad
\begin{tikzcd}[row sep=small, column sep=small]
& \mathsf{h} \arrow[dl] \arrow[dr] & \\
\mathsf{f} \arrow[d] & & \mathsf{g} \arrow[d] \\
\mathsf{0} & & x
\end{tikzcd}
\Rightarrow
\begin{tikzcd}[row sep=small, column sep=small]
& \mathsf{h} \arrow[dl] \arrow[ddl, bend left] \\
\mathsf{f} \arrow[d] & \\
\mathsf{g} \arrow[d] & \\
x &
\end{tikzcd}
.
$$

This allows the following infinite rewrite sequence:

$$
\begin{tikzcd}[row sep=small, column sep=small]
& \mathsf{h} \arrow[dl] \arrow[ddl, bend left] \\
\boxed{\mathsf{f}} \arrow[d] & \\
\mathsf{g} \arrow[d] & \\
\mathsf{0} &
\end{tikzcd}
\Rightarrow_{\mathcal{G}}
\begin{tikzcd}[row sep=small, column sep=small]
& \boxed{\mathsf{h}} \arrow[dl] \arrow[dr] & \\
\mathsf{f} \arrow[dr] & & \mathsf{g} \arrow[dl] \\
& \mathsf{0} &
\end{tikzcd}
\Rightarrow_{\mathcal{G}}
\begin{tikzcd}[row sep=small, column sep=small]
& \mathsf{h} \arrow[dl] \arrow[ddl, bend left] \\
\mathsf{f} \arrow[d] & \\
\mathsf{g} \arrow[d] & \\
\mathsf{0} &
\end{tikzcd}
\Rightarrow_{\mathcal{G}} \quad \ldots
$$

The problem here is that some kind of uncollapsing takes place. The shared node g stays. The reasons is, that g is at the same time part of the left-hand side and the context. The only node, which is guaranteed to not be part of the context is the root of the left-hand side. We see that the context may be affected by the rewrite step—raising the question whether it really is only a context?

With this two open questions we conclude the chapter on termination and non-termination of term graph rewriting. We will revisit termination in the next chapter on related work—together with work on different representation of term graphs, confluence, modularity, and memoisation.

7 Related Work

In this chapter we present related work whereby we understand related in a broad sense. We present general results from the literature on graph rewriting where the main focus lies on acyclic term graphs. However there are many different notions of term graphs and we give an overview over these notions in Section 7.1. Then we look at results on termination in Section 7.2, and at results for confluence in Section 7.3. We continue with results on modularity in Section 7.4. In the final Section 7.5 we show the difference between sharing nodes and sharing computation and present two formalisms which explicitly incorporate memoisation in term graph rewriting.

7.1 Representations of Term Graphs

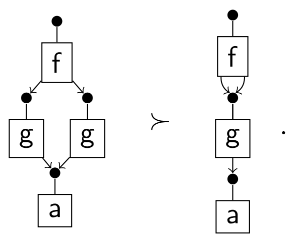
Term graphs come in many different flavours. This section presents some of these flavours, i.e. alternative definitions of term graphs. One of the most distinct differences is whether a term graph is acyclic or not. We first describe acyclic term graphs and then we look at definitions, which allow for cyclic term graphs.

The acyclic term graphs of Plump in [28, 29, 30] and Rao [31, 32, 33] are conceptually very similar to our setting. Term graphs are defined on the basis of hyper-graphs.

Definition 7.1. A *hyper-graph* is of the form $G = (N_G, E_G, \mathsf{lab}_G, \mathsf{att}_G)$, where N_G is a finite set of nodes, E_G is a finite set of hyper-edges, $\mathsf{lab}_G : E_G \to \mathcal{F}$ and $\mathsf{att}_G : E_G \to N_G^*$ assigns a string of nodes to a hyper-edge. For each edge $e \in E_G$, the length of $\mathsf{att}_G(e)$ is $1 + \mathsf{ar}(\mathsf{lab}_G((e))$.

Given an edge e with $\mathsf{att}(e) = n \cdot n_1 \cdots n_k$, then node n is the result node and $n_1 \cdots n_k$ are the argument nodes. A hyper-graph S is a term graph if there is a node $\mathsf{rt}(S)$ from which all nodes are reachable, if S is acyclic, and each node is the result node of a unique edge.

Example 7.2. Two term graphs based on hyper-graphs are shown next. The right term graph is a collapsed version of the left term graph, as indicated by collapsing ($\succ$) lifted to hyper-graphs:



Term graph rewriting on hyper-graphs is defined very similarly to our setting presented in Chapter 3: we find a morphism and redirect the edges appropriately followed by collecting only reachable nodes. With hyper-graphs usually also an explicit collapse relation is added to the rewrite relation by union.

Another acyclic formalism is non-copying term rewriting. Non-copying term rewriting is introduced by Kurihara and Ohuchi in [20]. It is a term-based formalism for graph rewriting. For an infinite set of marks M the signature $\mathcal{F}$ is extended to $\mathcal{F}^* = \{f^\mu \mid f \in \mathcal{F}, \mu \in M\}$ and $\mathcal{V}$ to $\mathcal{V}^* = \{x^\mu \mid f \in \mathcal{V}, \mu \in M\}$. Terms are then built over $\mathcal{T}(\mathcal{F}^*, \mathcal{V}^*)$ and called *marked terms*. A sub-set of marked terms are *well-marked* terms. A term is well-marked if all sub-terms have the same mark if and only if the sub-terms are identical. Well-marked terms directly correspond to term graphs.

Example 7.3. The two terms a^1 and a^1 are shared in the well-marked term $\mathsf{f}^0(\mathsf{a}^1, \mathsf{a}^1)$.

The rewriting relation is defined based on the term rewriting relation. There is an additional condition which enforces that every shared sub-term, i.e. sub-terms marked the same way, are replaced simultaneously. This is achieved by marking the rules. For the lhs the mark is determined by the matching, for the rhs no restriction is imposed. This allows to re-use marks from the term and thereby collapse implicitly.

Both, hyper-graphs and well-marked terms, are close to our notion of term graphs: they are acyclic and correspond to first-order terms. We chose our formalism as we were most familiar with it. Moreover the underlying data structure of graphs, as opposed to hyper-graphs or marked terms, is intuitive to start with. In some settings hyper-graphs and well-marked terms are more intuitive, e.g. a Top from Definition 5.3 is a hyper-edge in the hyper-graph setting.

Some authors, e.g. Ariola et al [1], Barendregt et al [7], and Barendsen [8] distinguish between *horizontal* sharing, i.e. graphs sharing common sub-graphs, and *vertical* sharing, i.e. graphs with cycles. So far we have only taken horizontal sharing into account. Now we want to present some formalisms which incorporate vertical sharing:

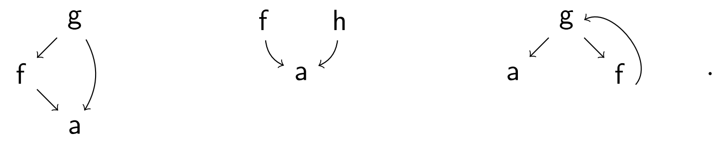
One of the most influential works is by Barendregt et al [7]. Their formalism does incorporate cycles, and also does not require a unique root from which all nodes are reachable. Their following definition gives a linear notion for graphs. Shared sub-graphs are identified by identifiers x or y.

Definition 7.4. Let $f \in \mathcal{F}$ and each identifier must exactly occur once in the context of $\mathsf{identifier} : f(\mathsf{node}, \ldots, \mathsf{node})$:

$$\begin{aligned} \mathsf{graph} &:= \mathsf{node} \mid \mathsf{node} + \mathsf{graph} \\ \mathsf{node} &:= f(\mathsf{node}, \ldots, \mathsf{node}) \mid \mathsf{identifier} \mid \mathsf{identifier} : f(\mathsf{node}, \ldots, \mathsf{node}) \quad . \end{aligned}$$

To illustrate the definition with some examples:

Example 7.5. The graphs $\mathsf{g}(\mathsf{f}(x : \mathsf{a}), x)$, $\mathsf{f}(x : \mathsf{a}) + \mathsf{h}(x)$ and $\mathsf{f}(x : \mathsf{g}(\mathsf{a}, \mathsf{f}(x)))$, are depicted next:

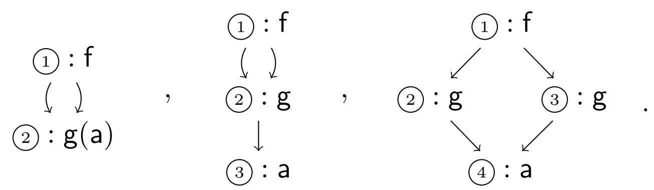


The rewriting relation is defined similarly to our setting: a morphism is found, the edges are redirected, and unreachable nodes are discarded. Most notable is the restriction to left-linear rules.

Ariola et al [1] and Barendsen [8] rely on a formalism that is based on terms and equations. Underlying their graph specification is a set of node specifications.

Definition 7.6. A *graph specification* S is a pair, $S = (\alpha, \{\alpha_1 = t_1, \ldots, \alpha_n = t_n\})$, where the α_i are pairwise disjoint *node variables* $\mathcal{N}$ and $t_i \in \mathcal{T}(\mathcal{F}, \mathcal{V} \cup \mathcal{N})$ for $1 \leqslant i \leqslant n$. The node α is a distinguished node in $\{\alpha_1, \ldots, \alpha_n\}$ and the root of the graph. The equations $\alpha_i = t_i$ are called *node specifications.*

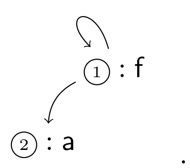
Example 7.7. We first draw three term graphs, where $\textcircled{1}$, $\textcircled{2}$, $\textcircled{3}$, and $\textcircled{4}$ are node variables, and then give their graph specification:



The graph specification for the left graph is $(\textcircled{1}, \{\textcircled{1} = \mathsf{f}(\textcircled{2}, \textcircled{2}), \textcircled{2} = \mathsf{g}(\mathsf{a})\})$. The graph specification for the graph in the middle is $(\textcircled{1}, \{\textcircled{1} = \mathsf{f}(\textcircled{2}, \textcircled{2}), \textcircled{2} = \mathsf{g}(\textcircled{3}), \textcircled{3} = \mathsf{a}\})$. Finally the graph specification for the right graph is $(\textcircled{1}, \{\textcircled{1} = \mathsf{f}(\textcircled{2}, \textcircled{3}), \textcircled{2} = \mathsf{g}(\textcircled{4}), \textcircled{3} = \mathsf{g}(\textcircled{4}), \textcircled{4} = \mathsf{a}\})$.

In the second example we see how we can represent a cyclic term graph.

Example 7.8. A graph may also be cyclic, which can be seen next, where again $\textcircled{1}$ and $\textcircled{2}$ are node variables:



The corresponding graph specification is $(\textcircled{1}, \{\textcircled{1} = \mathsf{f}(\textcircled{1}, \textcircled{2}), \textcircled{2} = \mathsf{a}\})$.

Each graph specification can be transformed into a canonical graph specification. Therefore one transforms each node specification to the form $x_0 = f(x_1, \ldots x_k)$ where

$x_0 \dots x_k \in \mathcal{V} \cup \mathcal{N}$. For example, the graph specification $(⓪, \{⓪ = f(g(x), ①), ① = ⓪\})$ is transformed into $(⓪, \{⓪ = f(②, ⓪), ② = g(x)\})$.

We presented these different representations to give an impression on the wealth of notions of term graphs. We now move on to another topic, which already received some attention in this thesis: termination of term graph rewriting.

7.2 Termination

After presenting our lexicographic path order for term graphs in Chapter 6, we now want to look at further results for termination and techniques to show termination.

We start by re-stating results by Plump [29, 30], which are based on acyclic hypergraphs. The first result has been mentioned before: every graph rewrite step can be simulated by k term rewrite steps—but not vice versa. Now we also incorporate collapsing.

Lemma 7.9. *Let $\mathcal{G}$ be a GRS. If $S \Rightarrow_{\mathcal{G}} \cup \succ T$, then $\mathsf{term}(S) \to^{k}_{\mathcal{R}(\mathcal{G})} \mathsf{term}(T)$, where $k \geqslant 0$.*

Proof. By assumption we have either $S \Rightarrow T$ or $S \succ T$. The first case follows by Lemma 4.1. The second case follows by Lemma 3.38. ☐

The result relies on a graph rewrite relation $\Rightarrow_{\mathcal{G}} \cup \succ$. With Chapter 4 we can transfer this result to other combinations. A direct consequence of Lemma 7.9 of is preservation of non-termination.

Lemma 7.10. *Let $\mathcal{R}$ be a TRS and $\mathcal{G}(\mathcal{R})$ the corresponding GRS. If $\to_{\mathcal{R}}$ is terminating, then $\Rightarrow_{\mathcal{G}(\mathcal{R})} \cup \succ$ is terminating.*

Proof. The proof is based on [30] and follows directly from Lemma 4.1 and the well-foundedness of $\succ$. By contra-position, we assume that $\Rightarrow_{\mathcal{G}} \cup \succ$ is not terminating for a graph rewrite system $\mathcal{G}$. Hence there exists an infinite sequence $S_1 \Rightarrow_{\mathcal{G}} \cup \succ S_2 \Rightarrow_{\mathcal{G}} \cup \succ \dots$ By Lemma 7.9 we have the infinite term rewrite sequence $\mathsf{term}(S_1) \to^{*}_{\mathcal{R}} \mathsf{term}(S_2) \to^{*}_{\mathcal{R}} \dots$ for the implied term rewrite system $\mathcal{R}(\mathcal{G})$ and $\to_{\mathcal{R}}$ is not terminating. ☐

This implies that techniques to show termination of term rewriting are applicable to term graph rewriting. The reverse does not hold. We gave a counter-example before: Example 5.1, which we also have shown terminating with our developed termination order in Chapter 6. We also want to present an ad-hoc proof of termination given in [30]. We start by defining a weight function $\tau : \mathcal{TG} \to \mathbb{N}$. For a term graph S, $\tau(S) = m + n + p$, where m is the number of nodes with label f, where the first two argument nodes are distinct, n is the number of nodes with label a and p is $|S|$. Then, $S \Rightarrow \cup \succ T$ implies $\tau(S) > \tau(T)$, hence there is no infinite sequence of $\Rightarrow \cup \succ$-steps.

Some results on weak normalisation, also in the setting of acyclic term graph rewriting with hyper-graphs, are by Rao [31]. He finds that a weakly normalising term rewrite system induces a weakly normalising graph rewrite system for right-linear rules, and for weakly innermost normalising rewrite steps.

Finally we present some techniques to show termination of term graph rewriting. For acyclic term graphs, Plump developed a recursive path order in [28]—which was the main inspiration for our results in Chapter 5 and 6.

Then most results for termination work on a more general notion of graphs. These graphs usually have cycles, have no distinguished root node, place more emphasis on edges through edge labels—in short: they do not resemble terms very much. They are thus not presented in Section 7.1.

One interesting line of work is by Bonfante and Guillaume in the context of natural language processing [10, 9]. Here the rewrite rules are graphs which represent grammatical transformations. Most notably these graphs are not size increasing, i.e. no new nodes are added. The authors show termination of the rules by assigning weights to graphs based on the nodes and edges in the graph. They then show a decrease of this weight for each step. Here the rewrite relation prohibits application of the rule in case a node is part of the context and part of the rewrite rule by the notion of context edges—a problem we too presented in Chapter 6. They implemented their termination technique in the tool Grew.[1]

Another termination technique for graph transformation systems is developed by Bruggink et al [12], [11]. Also there the idea is to assign weights to a special form of graphs called type graphs, and insisting on a strict decrease with transformation steps. The later [11] extends the earlier work and adds tropical and arctic type graphs.

Based on these works Zantema et al transfer the above techniques to a term graph setting in [37]—for left-linear and non-collapsing rules. The authors implemented the techniques in Grez.[2]

With this we conclude our investigation of termination and consider some confluence results for term graph rewriting.

7.3 Confluence

For confluence the relationship between term and term graph rewriting is reversed: If a graph rewrite system is confluent then the corresponding term rewrite system is confluent. Intuitively this makes sense: less rewrite steps are possible in the graph setting, which is good for termination, but bad for confluence. We start by showing confluence results by Plump [27]:

Lemma 7.11. *Let $\mathcal{R}$ be a TRS and $\mathcal{G}(\mathcal{R})$ the corresponding GRS. If $\Rightarrow_{\mathcal{G}(\mathcal{R})} \cup \succ$ is confluent, then $\rightarrow_{\mathcal{R}}$ is confluent.*

Proof. Let s, t, u be terms, where $s \rightarrow^* u$ and $u \rightarrow^* t$, and term graphs S and T, such that $s = \mathsf{term}(S)$ and $t = \mathsf{term}(T)$. By Lemma 4.2 and confluence of $\Rightarrow_{\mathcal{G}(\mathcal{R})} \cup \succ$ a term graph W exists such that $S\ (\Rightarrow_{\mathcal{G}(\mathcal{R})} \cup \succ)^*\ W$ and $T\ (\Rightarrow_{\mathcal{G}(\mathcal{R})} \cup \succ)^*\ W$. By Lemma 7.9, then $s \rightarrow^*_{\mathcal{R}} \mathsf{term}(W)$ and $t \rightarrow^*_{\mathcal{R}} \mathsf{term}(W)$. Hence, $\rightarrow_{\mathcal{R}}$ is confluent. □

[1] cf. `grew.loria.fr`
[2] cf. `www.ti.inf.uni-due.de/research/tools/grez/`

The lemma relies on $\Rightarrow_{\mathcal{G}(\mathcal{R})} \cup \succ$, but we can use our investigation in Chapter 4 to transfer it to other combinations. A counter example for the reverse direction is given next:

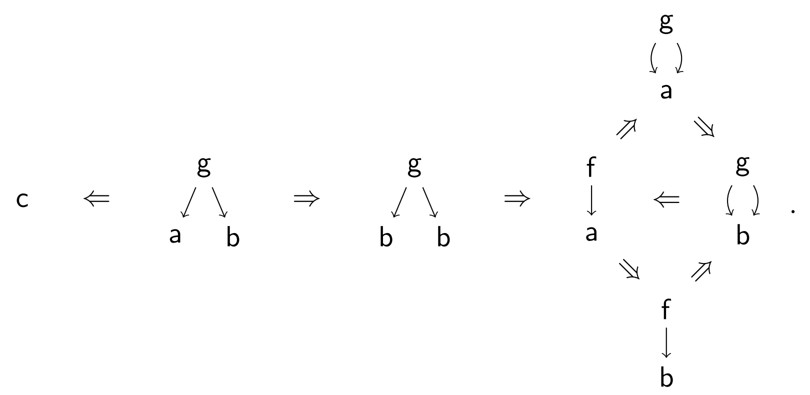
Example 7.12. The following TRS $\mathcal{R}$ is confluent:

$$\mathsf{f}(x) \to \mathsf{g}(x, x) \quad , \quad \mathsf{a} \to \mathsf{b} \quad , \quad \mathsf{g}(\mathsf{a}, \mathsf{b}) \to \mathsf{c} \quad , \quad \mathsf{g}(\mathsf{b}, \mathsf{b}) \to \mathsf{f}(\mathsf{a}) \quad .$$

Consider the following sequence:

$$\mathsf{c} \leftarrow \mathsf{g}(\mathsf{a}, \mathsf{b}) \to \mathsf{g}(\mathsf{b}, \mathsf{b}) \to \mathsf{f}(\mathsf{a}) \to \mathsf{g}(\mathsf{a}, \mathsf{a}) \quad .$$

The corresponding graph rewrite system is not even confluent for $\Rightarrow$ as can be seen next. Note, that the rule $\mathsf{f}(x) \to \mathsf{g}(x, x)$ prevents the graph rewrite step $\mathsf{g}(\mathsf{a}, \mathsf{b}) \to \mathsf{c}$ (as x is shared in the graph rewrite rule and no unsharing is present).



Additionally imposing (weak) normalisation remedies this situation.

Corollary 7.13. *Let $\mathcal{R}$ be a TRS and $\mathcal{G}(\mathcal{R})$ the corresponding GRS. If $\Rightarrow_{\mathcal{G}(\mathcal{R})} \cup \succ$ is (weakly) normalising, then $\Rightarrow_{\mathcal{G}(\mathcal{R})} \cup \succ$ if and only if $\to_{\mathcal{R}}$ is confluent.*

Important for confluence are overlaps. Given S_1 and S_2 then two term graphs *overlap* if there is a $m_1 : S_1 \to_{\mathcal{V}} T$ and $m_2 : S_2 \to_{\mathcal{V}} T$. Two rules overlap if $L_1 \restriction n$ overlaps with L_2 for some $n \in L_1$.

Two results on confluence that rely on overlaps, or rather the absence of overlaps, are given in [1] and [8]. Ariola et al [1] show that their graph specifications are confluent if there are no overlaps. If the rules are left-linear as well then confluence even holds in the presence of uncollapsing. Also Barendsen [8] shows confluence of left-linear and non-overlapping systems.

After investigating the termination and confluence behaviour in the previous two sections, we now investigate how those two properties behave with respect to modularity.

7.4 Modularity

Surprisingly many results are known for modularity of term graph rewriting. We present them here—with reference to their underlying notion of term graph rewriting. All presented result are based on acyclic formalisms.

We start by introducing an important notion for modularity: *disjoint* rewrite systems. Two rewrite systems $\mathcal{R}_1$ and $\mathcal{R}_2$ are disjoint, if their respective signatures $\mathcal{F}_1$ and $\mathcal{F}_2$ are disjoint, i.e., $\mathcal{F}_1 \cap \mathcal{F}_2 = \varnothing$.

Interestingly enough modularity of termination and confluence again behave quite differently for term and term graph rewriting. We start by modularity of termination which is studied in [26], [27], [32], [33], [25]. Although there are small differences in the underlying formalisms all find the following theorem.

Theorem 7.14. *For two disjoint GRS $\mathcal{G}_1$ and $\mathcal{G}_2$, $\Rightarrow_{\mathcal{G}_1} \cup \Rightarrow_{\mathcal{G}_2}$ is terminating if and only if $\Rightarrow_{\mathcal{G}_1}$ and $\Rightarrow_{\mathcal{G}_1}$ are terminating.*

Here Plump [26], and Rao [32, 33] use hyper-graphs as their underlying formalism. Rao [32, 33] incorporates collapsing by a union of the graph rewrite relation. He proves that modularity of termination needs neither confluence nor termination [32], and modularity of weak and strong normalisation, as well as semi-completeness and completeness, with different extensions by constructing different hierarchical combinations on function symbols [33].

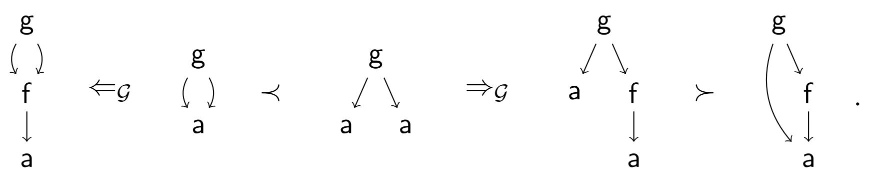
Theorem 7.14 is also shown by Kurihara et al. [20] based on their non-copying term rewriting approach to term graph rewriting—even if the rewrite systems share constructor symbols. They do not incorporate an explicit collapsing relation. Based on the same formalism Ohlebusch [25] provides a simple proof of the modularity of Theorem 7.14. His main insight on simplifying the proof is similar to our insight of argument of term graphs: instead of using multi-sets of sub-terms sets are sufficient.

In contrast to term rewriting, the union of disjoint systems need not preserve confluence. Even worse for term graph rewriting wit collapsing confluence is not preserved under signature extension. This is shown by the following example from [27].

Example 7.15. Consider the following left-linear GRS $\mathcal{G}$:

$$\mathsf{a} \Rightarrow \mathsf{f}(\mathsf{a}) \quad .$$

It is easy to see that $\mathcal{G}$ is confluent. But if we add a binary function symbol g confluence is lost:



In the presence of termination confluence of $\Rightarrow \cup \succ$ is preserved by the union of two disjoint systems as Plump shows in [27]:

Theorem 7.16. *Let $\mathcal{G}_1 \cup \mathcal{G}_2$ be the union of two disjoint GRSs. If $\Rightarrow_{\mathcal{G}_1} \cup \succ$ and $\Rightarrow_{\mathcal{G}_2} \cup \succ$ are confluent and terminating, then $\Rightarrow_{\mathcal{G}_1 \cup \mathcal{G}_2} \cup \succ$ is confluent and terminating.*

With this we move on towards the last section of this chapter: memoisation.

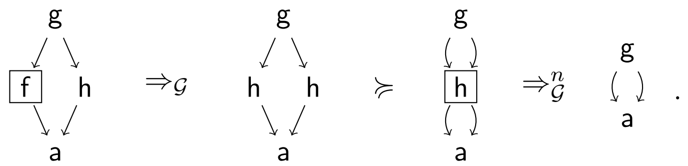
7.5 Shared Nodes and Memoisation

By sharing nodes we may share computation. Still there is a difference between sharing computation, i.e. memoisation and shared nodes. The next example serves to highlight this difference.

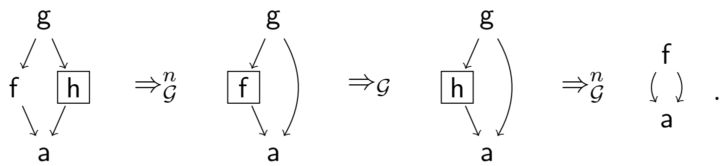
Example 7.17. Assume the linear GRS $\mathcal{G}$, where abusing notation we assume that the second rule takes n steps.

$$\begin{array}{c} \mathsf{f} \\ \downarrow \\ x \end{array} \Rightarrow \begin{array}{c} \mathsf{h} \\ \downarrow \\ x \end{array} \quad , \quad \begin{array}{c} \mathsf{h} \\ \downarrow \\ x \end{array} \Rightarrow^{n} x \quad .$$

The amount of steps we need depends on which redex node we choose. If we rewrite the f-node first, we need one collapsing step and $n+1$ rewrite steps:



If we rewrite the h-node first, the derivation yields $2 \cdot n + 1$ rewrite steps:



So while collapsing may save computation steps as in the first scenario, it is not memoisation. In the second scenario with memoisation we could have saved the second rewrite from the h-node by looking up the result.

Two approaches try to include memoisation to term graph rewriting—namely [16] and [3]. In [16] Hoffmann combines memoisation and sharing in a formalism of graph rewriting based on acyclic hyper-graphs. Therefore he incorporates dedicated rules for memoisation: tabulation and look-up. He then keeps results stored within the graph. The work by Avanzini et al. [3] is partly inspired by [16]. Also they introduce a formalism based on graphs to enable sharing and avoid blow-up in size, together with memoisation, i.e. tabulation to avoid re-computation.

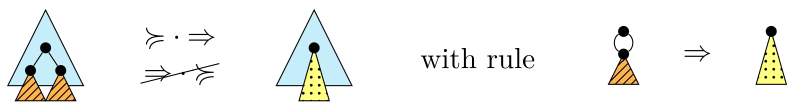
With this we conclude our review of the literature of term graph rewriting. The literature on term graph rewriting is very diverse—which also imposes challenges. Often the differences of the underlying formalisms are small, but it is not easy to see what the effects of these differences are—something we have already observed in Chapter 4.

8 Conclusion

We structure this conclusion along the three major blocks in this thesis: the influence of collapsing on the graph rewrite relation in Chapter 4, the influence on termination of term graph rewriting in Chapters 5 and 6, and the literature on term graph rewriting in Chapter 7. For each block we briefly re-capture the main results, highlight the most important insights, and show directions for future work.

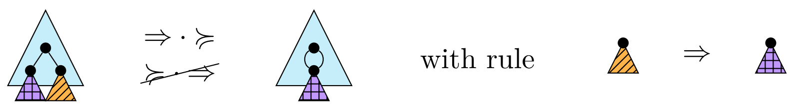
8.1 On Collapsing

To investigate the influence of collapsing on the graph rewrite relation the leading questions were: How to reasonably combine the graph rewrite relation with the collapsing relation: through concatenation or union? And which collapsing relation to choose: $\succ$, $\succcurlyeq$, $\succ^!$, or $\succ^{!+}$? We studied the different combinations with respect to inclusion and normal forms. Most importantly we provide notorious examples, which highlight the subtle differences. When we combine $\Rightarrow$ and $\succcurlyeq$ through concatenation, the obvious question is whether to perform rewriting or collapsing first. If we collapse first, then we can rewrite in the following:



$(\star_1)$.

If on the other hand we had chosen to rewrite before collapsing, we could not have applied the rule. As expected there is a dual scenario. In the following we cannot reach a term graph if we collapse first. That is, here we need to collapse after the rewrite step:



$(\star_2)$.

Both scenarios, $(\star_1)$ and $(\star_2)$, show the limitations of the graph rewrite relation ($\Rightarrow$) without collapsing. Interestingly enough, when we consider normal forms some differences disappear, and we find a different picture:

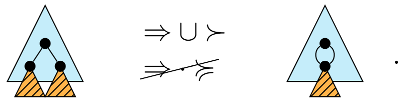
$$\mathsf{NF}(\succcurlyeq \cdot \Rightarrow) \subsetneq \mathsf{NF}(\Rightarrow \cdot \succcurlyeq) = \mathsf{NF}(\Rightarrow)$$

Informally speaking $\Rightarrow \cdot \succcurlyeq$ and $\Rightarrow$ allow for un-intuitive normal forms like the left term graph in $(\star_1)$ which is in normal form. We here refer to un-intuitive from a term rewriting perspective. To avoid this we need to incorporate collapsing. But as to the question of whether to collapse before or after the graph rewrite step, we know for n steps and $\triangleright = \succcurlyeq$, or $\triangleright = \succ^!$:

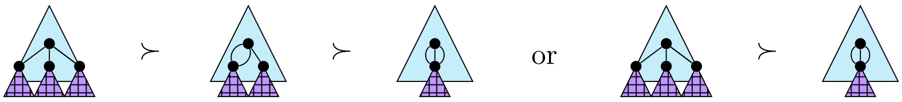
$$\triangleright \cdot (\Rightarrow \cdot \triangleright)^n = (\triangleright \cdot \Rightarrow)^n \cdot \triangleright$$

That is, if we have more than one step we only need an additional pre-processing step to apply a rule, or alternatively a post-processing step to reach a term graph. Thus the difference between $\Rightarrow \cdot \succcurlyeq$ and $\Rightarrow \cup \succ$ seems more interesting.

With $\Rightarrow \cdot \succcurlyeq$ we cannot perform a stand-alone collapsing step, but with $\Rightarrow \cup \succ$ we can:



On the other hand we need two steps in $\Rightarrow \cup \succ$ to simulate one step of $\Rightarrow \cdot \succcurlyeq$, e.g. in $(\star_1)$. Moreover, we could apply $\succ$ several times, or just once, as we can see next:



$(\star_3)$.

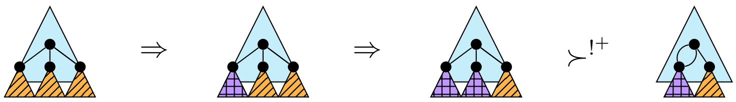
Still the amount of (strict) collapsing steps is bounded in the size of the term graph—and we can relate the number of steps between $\Rightarrow \cup \succ$ and $\Rightarrow \cdot \succcurlyeq$. For constants c_1 and c_2 and n steps we have:

$$\succcurlyeq \cdot (\Rightarrow \cdot \succcurlyeq)^n = (\Rightarrow \cup \succ)^{n \times c_1 + c_2 \times n}$$

There is a subtle difference between $\Rightarrow \cdot \succ^!$ and $\Rightarrow \cup \succ^{!+}$ though. This difference results in the following:

$$\succ^! \cdot (\Rightarrow \cdot \succ^!)^n \subsetneq (\Rightarrow \cup \succ^{!+})^{n \times c_1 + c_2 \times n}$$

We show the difference by an example. Take again the rule in $(\star_2)$. Because we can choose either $\Rightarrow$ or $\succ^{!+}$ we can also choose to apply $\Rightarrow$ subsequently more than once. This we cannot do for $\Rightarrow \cdot \succ^!$ and thus we cannot simulate the following rewrite sequence:



$(\star_4)$.

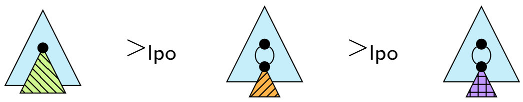
The main take away of Chapter 4 is: it's complicated. We see that small changes in how we combine the graph rewrite relation with collapsing can have significant consequences. As a rule of thumb we argue that $\Rightarrow \cup \succ$ provides the most freedom, but $\Rightarrow \cdot \succcurlyeq$ and $\succcurlyeq \cdot \Rightarrow$ provide control. Hereby the latter, $\succcurlyeq \cdot \Rightarrow$, is closer to term rewriting, because we do not run into the problem described in $(\star_1)$ where we cannot apply a rule based on the "wrong" degree of sharing. On the other hand, if every term graph is kept maximally shared by $\succ^!$, we have the advantage that the result of $\succ$ is predictable—that is, we increase control over $\succ$. We would recommend to choose $\succ^! \cdot \Rightarrow$ or $\Rightarrow \cdot \succ^!$ over $\Rightarrow \cup \succ^{!+}$, though. The effects of $\Rightarrow \cup \succ^{!+}$ are more difficult to predict as shown in $(\star_4)$.

Some of these differences disappear if we restrict the graph rewrite rules, for example if we restrict to left-linear rules. Thus we argue that for future work one should address what is the application of term graph rewriting. Much effort went towards term graph rewriting as implementation of term rewriting. For future work it seems beneficial to find application scenarios for term graph rewriting and derive requirements and restrictions from theses scenarios. This could help to choose the "best" combination. When we are interested in the number of steps, the choice does not vary significantly, as we have seen.

8.2 On Termination

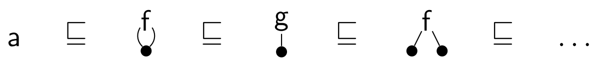
The second major part of this thesis was motivated by the gap between termination of term rewriting and termination of term graph rewriting. Our aim was to find out more about this gap and design a termination technique directly for term graph rewriting.

Based on the ideas in [28] we designed a lexicographic path order, $>_{\mathsf{lpo}}$, on term graphs. With $>_{\mathsf{lpo}}$ we can orient the rewrite sequence from the introduction.

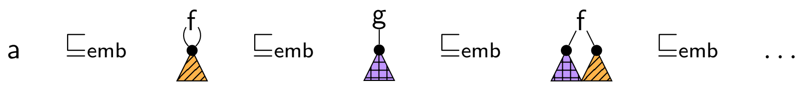


If we can find an order on all potential rewrite steps with $>_{\mathsf{lpo}}$, we can conclude termination. However we have a restriction on $>_{\mathsf{lpo}}$: it is defined only for term graphs whose nodes are mutually unreachable.

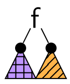
To show that $<_{\mathsf{lpo}}$ implies termination we employed and showed Kruskal's Tree theorem for term graphs. Informally it states that a wqo on Tops:



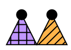
implies a wqo order on term graphs:



The main insight from this proof is that it is beneficial to treat the argument of a term graph as *one* graph i.e. an inlet graph:



with the argument graph



This preserves the structure of the argument and does not implicitly split up the argument into multiple argument graphs. It also slightly simplifies the proof as Higman's Lemma [15] can be omitted.

The main future challenge lies in implementing a termination technique and then developing an automated termination prover for term graph rewriting. To achieve automation we need a clear notion of context for term graphs. Therefore we need some restriction on how a term graph or a rewrite step can look like. This again brings us back to the conclusion we reached before: for a concrete application case some natural restrictions may hold.

8.3 On Literature

The third major block in this thesis is the related work and literature on term graphs. We showed several different formalisms: acyclic and cyclic term graphs, term graphs based on terms with markers or equations of terms, and term graphs based on hyper-graphs. We then presented results on termination, confluence, modularity, and memoisation—for these various formalisms.

One of the main insights from the related work is: there seems to be no general agreement on what a term graph actually is, and no standard way to represent graphs, rewrite systems, or collapsing. The main connection between the different formalisms is their close relationship to term rewriting. But this relationship to term rewriting is very different for every formalism. The relationship between the formalisms themselves hardly seems to be studied at all. So when approaching a paper on term graphs, one always needs to carefully study the underlying formalism first and check plenty of questions: Do term graphs have cycles? How is the rewrite relation defined? Are they restricted to left-linear rules? These small differences may have severe effects, thus they cannot be glossed over. Most results seem to transfer easily, which can be problematic too. To be sure one has to carefully check and re-prove a result—to find out that the result did indeed transfer easily, and not much was gained by the effort.

Consequently we find many results on the relationship between term rewriting and term graph rewriting, but not so many techniques to show a property like termination or confluence directly, let alone automatically, for a given term graph rewrite system.

Finally we observe that the literature on term graph rewriting seems rather scattered. The efforts go long back to the eighties. However as opposed to term rewriting, which is well established, basic notions of term graph rewriting are still fluid. We can witness this too as tools to analyse graph rewriting are only just emerging, but for a more general graph setting.

We conclude the conclusion with describing the impact of this thesis. As mentioned before, the results of Chapter 5 and 6 are published in [23]. Thus we presented the results on the 8th of April 2016 at the 9th International Workshop on Computing with Terms and Graphs (TERMGRAPH 2016), Eindhoven, Netherlands. Moreover I presented Chapter 5 as a poster at the ACM student research competition in course of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2016), St Petersburg, Florida on the 21st and 22nd of January 2016. The poster won the 3rd place in the graduate category. In course of the Oregon Programming Languages Summer School (OPLSS) 2016 on 23rd of June 2016 I gave a student research talk of the Chapter 5 and 6.

Finally I also gave a talk on this work in the workshop Logic, Complexity and Automation (LC&A) 2016, which was part of the Computational Logic in the Alps (CLA 2016) on 6th of September 2016.

Bibliography

[1] Zena M. Ariola and Jan Willem Klop. Equational Term Graph Rewriting. *Fundamenta Informaticae*, 26(3/4):207–240, 1996.

[2] Martin Avanzini. *Verifying Polytime Computability Automatically.* PhD thesis, University of Innsbruck, 2013.

[3] Martin Avanzini and Ugo Dal Lago. On Sharing, Memoization, and Polynomial Time. In *32nd International Symposium on Theoretical Aspects of Computer Science (STACS 2015)*, volume 30, pages 62–75. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015.

[4] Martin Avanzini and Georg Moser. Closing the Gap Between Runtime Complexity and Polytime Computability. In *Proceedings of the 21st International Conference on Rewriting Techniques and Applications*, volume 6 of *Leibniz International Proceedings in Informatics*, pages 33–48. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2010.

[5] Martin Avanzini and Georg Moser. Complexity Analysis by Graph Rewriting. In *Proceedings of the 10th International Symposium on Functional and Logic Programming*, volume 6009 of *Lecture Notes in Computer Science*, pages 257–271. Springer, 2010.

[6] Franz Baader and Tobias Nipkow. *Term Rewriting and All That.* Cambridge University Press, New York, USA, 1998.

[7] Hendrik P. Barendregt, Marko C.J.D. van Eekelen, John R.W. Glauert, Richard J. Kennaway, Marinus J. Plasmeijer, and Ronana M. Sleep. Term Graph Rewriting. In *Parallel Architectures and Languages Europe*, volume 259 of *Lecture Notes in Computer Science*, pages 141–158. Springer Berlin Heidelberg, 1987.

[8] Erik Barendsen. *Term Rewriting Systems*, chapter Term Graph Rewriting, pages 712–743. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2003.

[9] Guillaume Bonfante and Bruno Guillaume. Non-simplifying Graph Rewriting Termination. In *Proceedings of the 7th International Workshop on Computing with Terms and Graphs*, volume 110 of *Electronic Proceedings in Theoretical Computer Science*, pages 4–16. Open Publishing Association, 2013.

[10] Guillaume Bonfante and Bruno Guillaume. Non-size increasing Graph Rewriting for Natural Language Processing. *Mathematical Structures in Computer Science*, 2013. To Appear.

[11] H. J. Sander Bruggink, Barbara König, Dennis Nolte, and Hans Zantema. Proving Termination of Graph Transformation Systems Using Weighted Type Graphs over Semirings. In *Graph Transformation*, volume 9151 of *Lecture Notes in Computer Science*, pages 52–68. Springer International Publishing, 2015.

[12] H. J. Sander Bruggink, Barbara König, and Hans Zantema. Termination analysis for graph transformation systems. In *Proceedings of the 8th IFIP TC 1/WG 2.2 International Conference on Theoretical Computer Science*, volume 8705 of *LNCS*, pages 179–194. Springer, 2014.

[13] Nachum Dershowitz. Termination of Rewriting. *Journal of Symbolic Computation*, 3(1):69 – 115, 1987.

[14] Jean Goubault-Larrecq. Well-Founded Recursive Relations. In *Proceedings of the 15th International Workshop on Computer Science Logic*, pages 484–497, 2001.

[15] Graham Higman. Ordering by Divisibility in Abstract Algebras. *Procedings of the London Mathematical Society*, 3(2):326–336, 1952.

[16] Berthold Hoffmann. Term Rewriting with Sharing and Memoization. In *Proceedings of the 3rd International Conference on Algebraic and Logic Programming*, pages 128–142. Springer, 1992.

[17] Berthold Hoffmann and Detlef Plump. Implementing Term Rewriting by Jungle Evaluation. *Theoretical Informatics and Applications*, 25:445–472, 1991.

[18] Richard J. Kennaway, Jan Willem Klop, Michael R. Sleep, and Fer-Jan de Vries. On the Adequacy of Graph Rewriting for Simulating Term Rewriting. *ACM Transactions on Programming Languages and Systems*, 16(3):493–523, 1994.

[19] Joseph B. Kruskal. Well-Quasi-Ordering, The Tree Theorem, and Vazsonyi's Conjecture. *Transactions of the American Mathematical Society*, 95(2):210–225, 1960.

[20] Masahito Kurihara and Azuma Ohuchi. Modularity in Noncopying Term Rewriting. *Theoretical Computer Science*, 152(1):139–169, 1995.

[21] A. Middeldorp and H. Zantema. Simple Termination of Rewrite Systems. *Theoretical Computer Science*, 175:127–158, 1997.

[22] Aart Middeldorp. Term Rewriting. Lecture Notes, 2009. University of Innsbruck.

[23] Georg Moser and Maria A Schett. Kruskal's Tree Theorem for Acyclic Term Graphs. In *Proceedings of the 9th International Workshop on Computing with Terms and Graphs*, volume 225 of *Electronic Proceedings in Theoretical Computer Science*, pages 25–34. Open Publishing Association, 2016.

[24] Crispin St. J. A. Nash-Williams. On Well-Quasi-Ordering Finite Trees. *Mathematical Proc. Cambridge Philosophical Society*, 59:833–835, 1963.

[25] Enno Ohlebusch. Modularity of Termination for Disjoint Term Graph Rewrite Systems: A Simple Proof. *Bulletin of the European Association for Theoretical Computer Science*, 66, 1998.

[26] Detlef Plump. Implementing Term Rewriting by Graph Reduction: Termination of Combined Systems. In *Proceedings of the International Workshop on Conditional Term Rewriting Systems*, pages 307–317, 1990.

[27] Detlef Plump. Collapsed Tree Rewriting: Completeness, Confluence, and Modularity. In *Proceedings of the 3rd International Workshop on Conditional Term Rewriting Systems*, pages 97–112. Springer, 1993.

[28] Detlef Plump. Simplification Orders for Term Graph Rewriting. In *Proceedings of the 22nd International Symposium on Mathematical Foundations of Computer Science*, pages 458–467. Springer Berlin Heidelberg, 1997.

[29] Detlef Plump. *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 2, chapter Term Graph Rewriting, pages 3–61. World Scientific, 1999.

[30] Detlef Plump. Essentials of Term Graph Rewriting. *Electronic Notes in Theoretical Computer Science*, 51:277–289, 2002.

[31] M. R. K. Krishna Rao. Graph Reducibility of Term Rewriting Systems. In *Proceedings of the 20th International Symposium on Mathematical Foundations of Computer Science*, pages 371–381, 1995.

[32] M.R.K. Krishna Rao. Modularity of Termination in Term Graph Rewriting. In *Proceedings of the 7th International Conference on Rewriting Techniques and Applications*, volume 1103 of *Lecture Notes in Computer Science*, pages 230–244. Springer Berlin Heidelberg, 1996.

[33] M.R.K. Krishna Rao. Modular aspects of term graph rewriting. *Theoretical Computer Science*, 208(1-2):59 – 86, 1998.

[34] TeReSe, editor. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.

[35] Yoshihito Toyama. On the Church-Rosser property for the direct sum of term rewriting systems. *Journal of the ACM*, 34(1):128–143, 1987.

[36] Roel de Vrijer Vincent van Oostrom. *Term Rewriting Systems*, chapter Strategies, pages 475–547. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2003.

[37] Hans Zantema, Dennis Nolte, and Barbara König. Termination of Term Graph Rewriting. In *Proceedings of the 15th International Workshop on Termination*, pages 14:1–14:5, 2016.

Index