Equivalence of Reductions in Higher-Order Rewriting

Equivalentie van reducties in hogere-orde herschrijven
(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor aan de Universiteit Utrecht op gezag van de rector magnificus, prof. dr. J. C. Stoof, ingevolge het besluit van het college voor promoties in het openbaar te verdedigen op donderdag 15 mei 2008 des middags te 12.45 uur

door

Harrie Jan Sander Bruggink

geboren op 1 juni 1978
te Washington, Verenigde Staten van Amerika

Promotor: Prof. Dr. A. Visser
Co-promotor: Dr. V. van Oostrom

Contents

Acknowledgements

When I graduated from Cognitive Artificial Intelligence I was actively looking for a Ph.D. position in computer linguistics, but I ended up doing research on higher-order term rewriting, a completely different area of research. I'm sure I would have worked on a more linguistics oriented subject with great pleasure, but in the end I'm glad I chose the more mathematical one. The subject of rewriting is, in my opinion, a very interesting one, which uncovers deep insights in the modern, more and more computerized world.

Writing a Ph.D. thesis, even in an area one likes, is not something one can do entirely on his own. So here I would like to spend a page or two thanking those people who, in various ways, helped me to find my way in the world of theoretical computer science, to overcome the obstacles that every Ph.D. student seems to have to cope with from time to time, and, in general, just to have a nice life during this period.

First of all, I would like to express my gratitude to my promotor Albert Visser for the opportunity for doing this research. Albert's views and teachings on the subjects of logic and philosophy have helped me a lot in understanding the foundations of mathematics, and as such, the foundations and methodology of theoretical computer science.

Second, I am indebted to Vincent van Oostrom, my copromotor and daily supervisor. He was always there when I had questions, needed pointers into the existing literature, or faced difficult obstacles. I remember the time that I had almost given up hope of finishing this thesis at all; but Vincent's encouraging comments and clear schedule put me on track again. Also, his – sometimes controversial – world views ("everything is rewriting") were entertaining, and yet, insightful.

Third, I thank the members of my reading committee, Jan Bergstra, Marc Bezem, Delia Kesner, Jan Willem Klop and Roel de Vrijer.

Also, thanks to Jelke van Hoorn and Rick van Ewijk, for agreeing to be my "paranimfen", and my 'local ears and eyes' in Utrecht.

Writing a thesis is not possible without a pleasant working environment. In particular I would like to thank my respective roommates at Utrecht University: Joost Joosten (who couldn't beat me at penguin throwing), Dimitri Hendriks (who had a passion for chess) and Joop Leo (who magically turns a 10 euro note into a 20 euro one). Also I want to thank them and the other members

of the Theoretical Philosophy group for their interesting conversations during our daily lunches and weekly seminar.

One cannot conduct science 24 hours a day. For the necessary 'extra-curricular' diversion I would like to thank the "Marktoberdorf posse", Jeroen, Martijn, Hendrik Wietse and Arthur; my former roommates at the Tuindorp-West Complex, in particular Andreas, Anouk, Engel, Ion, Jeanine, Johan, Simone, Suzanne, Yvonne and Wouter; the "Sunday evening games crew", Joachim, Sander v.d. M. and Tomas; the members of my darts team, Dieuwertje, Eeske, Erik, Jelke, Maarten and Robbert; and all my other friends (insofar not previously mentioned), in particular Anne, Bouke, Fleur, Karianne, Paul, Roos, Sijmen and Wietse. And for all the people I forgot to mention: rest assured, it may have slipped my mind to mention you, but that doesn't mean I've forgotten you!

A special mention goes to my colleagues in Duisburg, Barbara König, Tobias Heindel and Vitali Kozioura. We had a great time together while I was waiting for my thesis to get accepted, and I hope we can continue our collaboration in the future.

Last but not least, I wish to thank my parents, Bia and Jos, my brother Sjoerd and his girlfriend Jiska for their support and love during the writing of this thesis.

One

Introduction

1.1 Rewriting as a model of computation

Theoretical computer science is to computer science what logic is to mathematics and theoretical philosophy to the natural sciences: it provides foundations and models of computations and programming languages. In this dissertation we investigate the realm of rewriting, in particular its subfield of rewriting theory. In general, rewriting is a computational paradigm in which computations are modelled by transitions (called steps or productions) between objects. The objects represent program states and the steps actions that change the state of the program. These steps are described by rewrite rules. A rewrite rule consists of a left-hand side and a right-hand side and applying a rewrite rule to an object consists of removing the left-hand side of the rule from that object (called redex, after *red*ucible *ex*pression) and replacing it with the right-hand side. Many types of mathematical objects and rewrite rules can be used, including first-order terms, strings and graphs, giving rise to various rewriting paradigms. The different concrete formats of rewriting share the following common properties:

- Rewriting is *discrete*. If we apply a rewrite rule to an object, we immediately obtain a new object. There are no objects 'in between'.
- Rewriting is *local*. Rewrite rules are applied to part of the object, and leave the rest of the object 'as-is'.
- A related property is that rewriting is *asynchronous*. Steps which do not depend on each other can be applied in any order, or at the same time. A consequence of this is that rewriting is very suitable for modelling parallel and concurrent computations, that is, computations in which independent parts of the same computation are carried out synchronously.

- In principle, rewriting is *non-deterministic*. There is no a priori preference on the possible steps, although much research is done on the subject of strategies, where such a preference on the steps is added. Also, most actual (rewriting based) programming languages employ some predefined, evaluation strategy, such as *strict* evaluation or *lazy* evaluation.

The form of rewriting that we investigate here is higher-order rewriting, in particular the class of Higher-order Rewrite Systems, as introduced by Nipkow [36, 37, 31]. Higher-order rewriting is a symbiosis of two classical rewriting paradigms: the λ-calculus, which features higher-order variables and variable binding, and first-order term rewriting, which features algebraic pattern matching. Higher-order Rewrite Systems (HRSs) are a powerful tool to study the meta-theory of declarative programming languages, such as λProlog and Haskell, on the one hand, and theorem provers and proof assistants, such as Isabelle, on the other. Additionally, many rewriting paradigms, such as first-order term rewrite systems and (extensions of) λ-calculi can be encoded as instances of HRSs (see for example Sect. 4.6.2 for an HRS which encodes the simply typed λ-calculus), so that results obtained for HRSs easily carry over to other interesting domains.

1.2 Equivalence of reductions

As noted above, rewriting is an asynchronous, non-deterministic computational paradigm: steps which do not depend on each other by locality, can be performed in any order or even at the same time. We want to equate computations which are the same except for the order in which such independent steps are performed. As a metaphor, we consider lists of natural numbers. Two lists are considered equivalent if they contain the same numbers, in any order. Likewise, two reductions are equivalent, if they contain the same steps. Formalizing the notion of equivalence of lists is, of course, trivial, but when dealing with reductions this is not the case, because steps may be duplicated, erased, nested, etc. We give examples of duplication and erasure in first-order term rewriting systems (TRSs), a form of rewriting which I assume the reader is familiar with:

- Consider the following TRS:

$$\begin{aligned} \mathsf{f}(x) &\to \mathsf{g}(x, x) \\ \mathsf{a} &\to \mathsf{b} \end{aligned}$$

  The term $\mathsf{f}(\mathsf{a})$ contains two independent redexes: the f-redex and the a-redex. Consider the following two reductions:

$$\begin{aligned} &\mathsf{f}(\mathsf{a}) \to \mathsf{f}(\mathsf{b}) \to \mathsf{g}(\mathsf{b}, \mathsf{b}) \\ &\mathsf{f}(\mathsf{a}) \to \mathsf{g}(\mathsf{a}, \mathsf{a}) \to \mathsf{g}(\mathsf{b}, \mathsf{a}) \to \mathsf{g}(\mathsf{b}, \mathsf{b}) \end{aligned}$$

  The first reduction contract the a-redex first and then the f-redex, while the second contracts the f-redex first and the a-redexes afterwards.

Intuitively, both reductions are equivalent. However, they do not even have the same number of steps, because the redex a of the source term is duplicated in the second reduction.

- Consider the following TRS:

$$\begin{aligned} \mathsf{f}(x) &\to \mathsf{c} \\ \mathsf{a} &\to \mathsf{b} \end{aligned}$$

and the following two reductions:

$$\begin{aligned} &\mathsf{f}(\mathsf{a}) \to \mathsf{f}(\mathsf{b}) \to \mathsf{c} \\ &\mathsf{f}(\mathsf{a}) \to \mathsf{c} \end{aligned}$$

Again, both reductions are intuitively equivalent, but, because the step from a to b was erased in the second reduction, both reductions do not have the same number of steps.

Above is demonstrated how duplication and erasure make defining a notion of equivalence of reductions already non-trivial in the first-order case. In higher-order rewriting, we will need to deal with the additional problem of nesting. Examples of this will be given in later chapters.

1.3 Motivations

What are the reasons for studying equivalence of reductions? First, studying equivalence of reductions is interesting from a conceptual point of view. The answers to (related) questions like "What makes one step in one reduction 'the same' as another step in another reduction?" and "What does it mean for two reductions to be equivalent?" gives us insight in the nature of computation. However, answering these questions is not only interesting by itself. It is also required if one wants to investigate, for example, strategies or optimal reductions.

Strategies and the needed strategy. A strategy for a rewrite system is defined in [53, Chap. 9] as a sub-rewrite system which shares the same normal forms. In practice, a strategy can be seen as a (potentially non-deterministic) algorithm which selects the step which must be taken from a given object. Equating reductions and steps helps in defining and investigating strategies for rewriting systems. For example, the property of *fairness*, which states that every redex must be contracted at some point, requires a notion of persistence of redexes, and thus a notion of equivalence of reductions.

This is also required to define the so-called *needed strategy*. A step is *needed in a reduction* if it is present in all equivalent reductions. (This means that the descendents of the step are never erased; for the notion of standardness we define in Chapter 5, the steps which are needed in a reduction are exactly the ones which are present in the equivalent standard reduction.)

In orthogonal term rewrite systems, all reductions to normal form are equivalent. We define that a step is *needed* if it is needed in some reduction to normal form, that is, if, in order to reach a normal form, the step must be performed at some point. The needed strategy can now be defined as being the strategy which alway performs a needed step. It is well-known that the needed strategy is normalizing for orthogonal, first-order TRSs, that is, if a normal form can be reached, it will be reached by the strategy. Unfortunately, whether a redex is needed or not is undecidable in general.

Optimality. The question which redex must be contracted in order to reach a normal form in the least number of steps, is in general undecidable. Lévy [29] showed that, in case of the λ-calculus, the question is decidable if, instead of steps, we take 'family steps' to be the atomic transitions. Family steps are steps, in which redexes, which are created 'in the same way', may be contracted at the same time. For example, in the first example of the previous page, a family step may reduce the two copies of the duplicated a-redex in one go. The notion of equivalence of reductions is essential to formally define the notion of redex family.

It is not immediately clear that a family step is an appropriate unit of computation, but the graph implementation of Lamping [27] made it seem plausible to assume that it is. In his graph representation, terms are represented by graphs. It appeared that things could be set up in such a way, that redexes which are part of the same redex family, could be represented by the same part of a graph, and, since graph rewrite steps operate locally, be contracted at the same time.

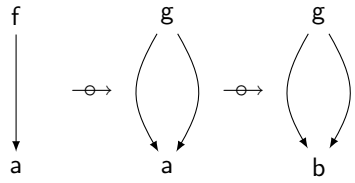
Consider for example the rewrite system consisting of the rules

$$\mathsf{f}(x) \to \mathsf{h}(x, x)$$
$$\mathsf{a} \to \mathsf{b}$$

and the reduction

$$\mathsf{f}(\mathsf{a}) \to \mathsf{h}(\mathsf{a}, \mathsf{a}) \to \mathsf{h}(\mathsf{b}, \mathsf{a}) \to \mathsf{h}(\mathsf{b}, \mathsf{b}).$$

The a-redex is duplicated, and therefore two proper steps are needed to reduce to a normal form. When the terms are represented as graphs, the same normal form can be reached as follows:[1]



[1]Note that the figure is only an illustration; actual implementations of the idea require control nodes to carry out duplication and, in the presence of bound variables, to keep track of those bound variables.

Although it was subsequently proved that family steps are not an appropriate unit of computation [2], because the number of control steps needed to isolate the 'family' redexes is only bounded by a superexponential function, in practice promising results have been obtained (see below) and the field of optimality is still alive.

Optimal (in the sense of Lévy) graph-based implementations have been devised for the λ-calculus [27, 3] and first-order TRSs [48, 49, 50]. Interestingly, in the first case the difficult part is to keep track of nested bound variables, while in the second case the difficult part is making sure that matching works without 'unsharing' too much of the graph. An optimal implementation of higher-order rewriting would need to cope with both of those difficulties.

1.4 Three equivalences of reduction

In this dissertation we formalize the notion of equivalence of reductions in higher-order rewriting in three different ways:

- permutation equivalence,
- standardization equivalence, and
- projection equivalence.

All of these notions were previously introduced for other rewriting paradigms, but they were all called "permutation equivalence". Van Oostrom & De Vrijer [42, 43] were the first to distinguish between the various ways in which to formalize permutation equivalence, and to explicitly prove them equivalent. We mostly adopt their terminology.

Permutation equivalence. Two reductions are *permutation equivalent* if the one can be transformed into the other by permuting adjacent, independent steps. In the list metaphor, lists are equivalent if one can be transformed into the other by swapping adjacent numbers.

Permutation equivalence is formalized, by giving a set of equations between reductions. Two reduction are permutation equivalent if the one can be converted into the other by these equations.

Standardization equivalence. In a standard reduction the redexes are contracted from outside to inside and left to right. Standardization consists of finding a permutation equivalent standard reduction for a given input reduction. This can be compared to sorting lists of natural numbers.

We formalize standardization equivalence by giving two standardization procedures, called weak and strong standardization by Klop [25], and selection and inversion standardization by Van Oostrom & De Vrijer [42, 43], respectively. The first can be seen as a specific strategy for the second.

Because, in the case of lists, each equivalence class of lists contains a unique sorted list, two lists can be considered equivalent if sorting the one

yields the same list as sorting the other. We show that each permutation equivalence class of reductions also contains a unique standard reduction. Hence, standardization can be used to decide permutation equivalence: two reductions are equivalent if and only if they have the same standard reduction. This form of equivalence of reductions will be called *standardization equivalence*.

Projection equivalence. Two reductions are *projection equivalent* if projection of either reduction over the other yields the empty reduction. In the list metaphor: lists A and B are equivalent if removing all numbers in B from A yields the empty list and vice versa.

We formalize projection equivalence by defining a projection operation / on higher-order reductions. For reductions $\mathcal{R}, \mathcal{S}$, the formula $\mathcal{R} \,/\, \mathcal{S}$ denotes the residual of $\mathcal{R}$ after $\mathcal{S}$ has been performed. This projection operator satisfies the laws of abstract residual theory.

The main result of this thesis (Theorem 7.2.1) will be that, for reductions on which all of the notions of equivalence are defined, the three notions of equivalence are the same.

1.5 Outline of this dissertation

Below, I give a short overview of the topics which will be covered in the various chapters of this dissertation:

Chapter 1: Introduction. You are currently reading the introduction. We give motivation for and an overview of this dissertation. Additionally, in the next section some mathematical preliminaries are introduced.

Chapter 2: Rewriting and proof terms. We introduce the various notions of rewriting that are used in this dissertation: Abstract Rewrite Systems, the λ-Calculus and Higher-order Rewrite Systems. But the chapter also discusses new results, in particular proof terms for Higher-order Rewriting Systems. These proof terms are a term representation of (multi)steps and reductions and are very convenient to manipulate steps and reductions.

Chapter 3: Permutation equivalence. In this (short) chapter we define the notion of permutation equivalence for both finite and infinite reductions, and we prove some handy properties of this notion.

Chapter 4: Finite Family Developments. In this chapter we show that Higher-order Rewriting Systems enjoy the property of Finite Family Developments. This property ensures that every reduction in which the 'creation depth' of every function symbol is bounded is finite and vice versa. The proof is quite technical but the result is essential in Chapter 5.

Chapter 5: Standardization. We prove that for every reduction there is a permutation equivalent reduction in which the redexes are contracted

visually from left to right. We prove this by giving two independent procedures which transform an arbitrary reduction into an equivalent standard one. The result of this chapter is then used to define an alternative notion of equivalence of reductions, standardization equivalence: two reductions are equivalent if they have the same standard reduction. An easy corollary of the earlier results of the chapter is finally that standardization equivalence is equivalent to permutation equivalence.

Chapter 6: Residuals. We define the notions of projection and residuals, which together answer the question: what remains of a reduction after another reduction has been performed. These notions are then used two define a third notion of equivalence of reductions, which is then proved to be equivalent to permutation equivalence and standardization equivalence.

Chapter 2 and 3 are required to understand the proceeding chapters and Chapter 5 uses the result of Chapter 4 but can be understood separately. Chapter 6 can be read independently of chapters 4 and 5, except for Sect. 6.6, which establishes a correspondence between the results of Chapters 5 and 6.

1.6 Mathematical preliminaries

In this section we fix some terminology and notations for common mathematical structures, such as relations, functions and sequences. It is not meant as a general introduction to the foundations of mathematics; a basic understanding of set theory is assumed.

Sets. We employ the usual notion of a set, for example as axiomatized in a formal set theory like ZFC. We use the notation $A \subseteq B$ to denote that A is a subset of B, and $A \subset B$ to denote that A is a proper subset of B. The inverses of $\subseteq$ and $\subset$ are denoted $\supseteq$ and $\supset$, respectively. We denote n-ary tuples by $\langle a_1, \ldots, a_n \rangle$. The Cartesian product of two sets A and B is denoted:

$$A \times B = \{\langle a, b \rangle \mid a \in A \,\&\, b \in B\}.$$

Relations. A *relation* R is a set of tuples together with its domain and codomain. Formally: a relation is a triple $R = \langle R_0, A, B \rangle$ such that $R_0 \subseteq A \times B$. We write $\mathit{Dom}(R) = A$ and $\mathit{Cod}(R) = B$, and we will say in this case that R is a relation *from* A *to* B. The expression $a\ R\ b$ denotes the fact that $a \in \mathit{Dom}(R)$, $b \in \mathit{Cod}(R)$ and $\langle a, b \rangle \in R_0$.

Note that we explicitly allow the possibilities that $\mathit{Dom}(R) \supset \{a \mid a\ R\ b \text{ for some } b\}$ and $\mathit{Cod}(R) \supset \{b \mid a\ R\ b \text{ for some } a\}$. We call a relation *total* if $\mathit{Dom}(R) = \{a \mid a\ R\ b \text{ for some } b\}$ and *surjective* if $\mathit{Cod}(R) = \{b \mid a\ R\ b \text{ for some } a\}$.

Let R be a relation from A to B. The inverse of R, denoted R^{-1}, is the relation such that $\mathit{Dom}(R^{-1}) = \mathit{Cod}(R)$, $\mathit{Cod}(R^{-1}) = \mathit{Dom}(R)$ and $a\ R^{-1}\ b$ if and only if $b\ R\ a$.

Let S be an arbitrary set. The following notations are used w.r.t. a relation R from S to S:

- $R^{=}$ is the reflexive closure of R;
- R^{+} is the transitive closure of R;
- R^{*} is the reflexive, transitive closure of R, i.e. $R^{*} = R^{=} \cup R^{+}$;
- $R^{!}$ denotes normalization with respect to R, that is $a\ R^{!}\ b$ if $a\ R^{*}\ b$ and there is no c such that $b\ R\ c$.

Let $R = \langle R_0, A, B\rangle$ be a relation from A to B, and $S = \langle S_0, B, C\rangle$ a relation from B to C. The composition of R and S is a relation from A to C and is defined as: $R\,;S = \langle R_0\,;S_0, A, C\rangle$, where

$$R_0\,;S_0 = \{\langle a, c\rangle \mid \exists b \in B : \langle a, b\rangle \in R_0\ \&\ \langle b, c\rangle \in S_0\}.$$

Let R be a relation from S to S, and $A \subseteq S$ a set. An *R-maximal* element of A is an $x \in A$ such that $\forall y \in A.x\ R\ y \rightarrow y\ R\ x$. An *$R$-minimal* element of A is a R^{-1}-maximal element of A.

Functions. A *function* F is a relation such that for each $a \in \mathit{Dom}(F)$ there is exactly one $b \in \mathit{Cod}(F)$ such that $a\ F\ b$. The expression $F(a)$ denotes the unique $b \in \mathit{Cod}(F)$ such that $a\ F\ b$; if $a \notin \mathit{Dom}(F)$, then $F(a)$ is said to be *undefined*. The symbol id_A denotes the *identity function* on the domain A, defined as $\mathsf{id}_A(a) = a$, for all $a \in A$. If the domain is clear from the context, it will be omitted.

Let F be a function from A to B. The function $F{\restriction}C$ is the function F *restricted* to C, that is, if $F' = F{\restriction}C$, then $\mathit{Dom}(F') = A \cap C$, $\mathit{Cod}(F') = B$ and $F'(a) = F(a)$ for all $a \in A \cap C$.

Ordinal numbers. We employ the Von Neumann definition of ordinal numbers (or ordinals for short): a set A is an ordinal if and only if A is totally ordered with respect to the subset relation and every element of A is also a subset of A. We are only interested in ordinals up to ω, the first infinite ordinal. The natural numbers, denoted by $\mathbb{N}$, are the finite ordinals. The "ordinals smaller than or equal to ω" are then $\mathbb{N} \cup \{\omega\}$. Greek lowercase letters α, β range over ordinal numbers, while i, j, k, n, m range over natural numbers (unless otherwise indicated).

We explicitly state that according to the definition, every ordinal is equal to the set of its predecessors. That is: $4 = \{0, 1, 2, 3\}$ and $\omega = \mathbb{N}$.

Sequences. A sequence $\mathbf{a}$ over a set A is a function from an ordinal $\alpha \leq \omega$ to A; here, α is called the *length* of the sequence, denoted $|\mathbf{a}| = \alpha$. Sequences of length ω are called *infinite*, other sequences are *finite*. The sequence of length 0, the empty sequence, is denoted by ε.

If $\mathbf{b} = \mathbf{a}{\restriction}\alpha$, then $\mathbf{b}$ is called the α-prefix of $\mathbf{a}$, also written $\mathbf{a}[\alpha]$; a sequence $\mathbf{b}$ is a prefix of $\mathbf{a}$, written $\mathbf{b} \sqsubseteq \mathbf{a}$, if it is the α-prefix of $\mathbf{a}$ for some α. *Composition* of sequences $\mathbf{a}$ and $\mathbf{b}$, denoted by $\mathbf{a}\,;\mathbf{b}$, is defined as follows:

$$(\mathbf{a}\,;\mathbf{b})(i) = \begin{cases} \mathbf{a}(i) & \text{if } i < |\mathbf{a}| \\ \mathbf{b}(i - |\mathbf{a}|) & \text{otherwise} \end{cases}$$

Note that the subtraction in the second clause is defined, because $|\mathbf{a}| < \omega$ (if not, the first clause would always apply). A related observation is the fact that, according to this definition, if $\mathbf{a}$ is an infinite sequence, then $\mathbf{a}\,;\mathbf{b} = \mathbf{a}$.

We introduce the following shorthand for finite sequences: a sequence $a_1, \ldots, a_n$ of length n will be written as $\overline{a_n}$, or as $\overline{a}$ if n is not important.

If $\mathcal{A}$ is a set of sequences over A, and $\mathbf{a}$ is a sequence over A, then we write:

- $\mathbf{a}\,;\mathcal{A}$ for $\{\mathbf{a}\,;\mathbf{b} \mid \mathbf{b} \in \mathcal{A}\}$; and
- $\mathcal{A}\,;\mathbf{a}$ for $\{\mathbf{b}\,;\mathbf{a} \mid \mathbf{b} \in \mathcal{A}\}$.

Lexicographic ordering. Let A be a set and $\sqsubset$ a strict ordering on A. The lexicographic ordering $\sqsubset_{\text{lex}}$ on sequences on A is defined as follows:

$$\mathbf{a} \sqsubset_{\text{lex}} \mathbf{b} \quad \text{if} \quad \exists k \Big(\big(\mathbf{a}(k) \sqsubset \mathbf{b}(k)\big) \;\&\; \forall i < k \, \big(\mathbf{a}(i) = \mathbf{b}(i)\big) \Big).$$

It is well-known that $\sqsubset_{\text{lex}}$ is well-founded on sequences of bounded length if and only if $\sqsubset$ is well-founded.

Notational conventions. Let $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ be sequences. We will write

$$a_1, \ldots, b_k, \ldots, a_n \quad \text{for} \quad a_1, \ldots, a_{k-1}, b_k, a_{k+1}, \ldots, a_n,$$

that is, the same sequence where the element a_k is replaced by b_k.

More notational conventions will be introduced in the respective chapters.

Two

Rewriting and proof terms

2.1 Introduction

In this chapter we introduce the various notions of rewriting that are used in this thesis. The most notable of those is the class of Higher-order Rewrite Systems (HRSs). HRSs are built 'on top of' the simply typed λ-calculus, which is therefore also briefly introduced. Also, we define Abstract Rewrite Systems (ARSs), firstly because they allow us to introduce some rewriting notions without restricting ourselves to a particular form of rewriting, and secondly, because they allow us to develop some theory from a more abstract viewpoint. Additionally, in Chapter 4 we define a λ-calculus with explicit substitutions, the λx-calculus, but since it is only used locally, we do not give it any attention here.

Additionally, we define *proof terms*, which are explicit witnesses to steps, for both the λ-calculus and HRSs. In a sense, proof terms are to steps what typed λ-expressions are to proofs: they are term representations of non term-like structures. Just like proof unfolding is much easier to describe as β-reduction on λ-terms than as an operation on proofs, our proof terms will prove to be very convenient to express meta-operations on and meta-equivalences of steps and reductions, such as permutation equivalence (Chapter 3), standardization (Chapter 5) and a residual operator (Chapter 6).

The idea of proof terms for rewriting paradigms is not new. They were also used for the λ-calculus by Hilken [17] and for first-order term rewrite systems (TRSs) by Van Oostrom & De Vrijer [42, 43]. In the second case, proof terms were introduced for the same reasons as here: in order to define operations on proof terms more easily.

2.2 Abstract rewriting

In this section, we explore the most abstract form of rewriting, abstract rewrite systems (ARSs). In ARSs, no structure of the objects which are rewritten is

assumed. This allows us to distinguish properties of rewriting that depend on the structure of the object (for example terms) from properties which hold for arbitrary forms of rewriting. Additionally, it avoids repeating definitions of similar notions for all sorts of rewriting. In later sections, specific instances of ARSs are given, in particular Higher-order Rewrite Systems and the simply typed λ-calculus with β-reduction.

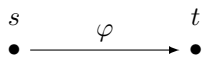
2.2.1 Abstract Rewrite Systems

Abstract rewrite systems (ARSs) consist of a set of steps, giving rise to the one-step rewrite relation.

Definition 2.2.1 (cf. [53, Def. 8.2.2])**.** An *Abstract Rewrite System* (ARS) is a structure $\mathfrak{A} = \langle A, \Phi, \mathrm{src}, \mathrm{tgt} \rangle$, where:

- A is an arbitrary set of *objects*, called the *domain* of $\mathfrak{A}$;
- Φ is a set of *steps*; and
- src and tgt are functions from Φ to A, returning the source and target of each step, respectively.

An ARS, as defined above, describes what the objects are that are being rewritten and has a set of (explicit witnesses to) steps, each of which is associated with a source and a target. If φ is a step with $s = \mathrm{src}(\varphi)$ and $t = \mathrm{tgt}(\varphi)$, we write $\varphi : s \to_{\mathfrak{A}} t$. The subscript $\mathfrak{A}$ may be omitted if clear from the context. Diagrammatically, we write:



Note, that we allow that two distinct steps have the same source and target. This is a desired feature: it allows us to cope with *syntactic accidents* (see for example [53, page 37]). However, often we are not interested in which specific step between two terms is performed, but only in which objects can be reached in one step from a particular object. For this purpose we introduce the notion of *rewrite relation*: $s \to_{\mathfrak{A}} t$ if there exists a φ such that $\varphi : s \to_{\mathfrak{A}} t$.

2.2.2 Reductions

In general, given some object $a \in A$, we will not only be interested in objects that are reachable from a in exactly one step, but we will be interested in objects that are reachable from a in any number of steps.

Definition 2.2.2. Let $\mathfrak{A} = \langle A, \Phi, \mathrm{src}, \mathrm{tgt} \rangle$. An $\mathfrak{A}$-reduction is a tuple $\mathcal{R} = \langle R, s \rangle$, where:

- $R = \varphi_0, \varphi_1, \ldots$ is a (possibly infinite) sequence of $\mathfrak{A}$-steps such that $\mathrm{tgt}(\varphi_i) = \mathrm{src}(\varphi_{i+1})$, for all consecutive steps φ_i, φ_{i+1} of R.

- $s \in A$, the *source* of the reduction, is a designated object such that, if $|R| > 0$, $s = \mathrm{src}(\varphi_0)$.

Keeping an explicit reference to the source of the reduction is necessary in order to distinguish between empty reductions from different objects: if a and b are distinct objects, then the empty reductions from a to a and from b to b should be distinct reductions. However, in practice we will usually not mention the source: in most cases, it is just the source of the first step. If the reduction under consideration is empty, then the source will be stated explicitly or clear from the context.

The functions src and tgt are extended to reductions in the following way. The function $\mathrm{src}(\mathcal{R})$ just returns the source component of the reduction $\mathcal{R}$. In the definition of $\mathrm{tgt}(\mathcal{R})$ the following three cases are distinguished:

- If $\mathcal{R}$ is empty, then $\mathrm{tgt}(\mathcal{R}) := \mathrm{src}(\mathcal{R})$.
- If $\mathcal{R}$ is the finite reduction $\varphi_0, \ldots, \varphi_{n-1}$ of length n, then $\mathrm{tgt}(\mathcal{R}) := \mathrm{tgt}(\varphi_{n-1})$.
- If $\mathcal{R}$ is an infinite reduction, then $\mathrm{tgt}(\mathcal{R})$ is undefined.

Reductions and steps with the same source are called coinitial. Reductions and steps with the same target (if defined) are called cofinal.

Reductions $\mathcal{R} = \langle R, r\rangle$ and $\mathcal{S} = \langle S, s\rangle$ are composable if $\mathcal{R}$ is a finite reduction and $\mathrm{tgt}(\mathcal{R}) = \mathrm{src}(\mathcal{S})$. The composition of the two reductions is then defined as: $\mathcal{R};\mathcal{S} := \langle R;S, r\rangle$. It is easy to see that, like composition of sequences, composition of reductions is associative, that is, $(\mathcal{R}\,;\mathcal{S})\,;\mathcal{T} = \mathcal{R}\,;(\mathcal{S}\,;\mathcal{T})$.

We write $\mathcal{R} : a \twoheadrightarrow_{\mathfrak{A}} \cdots$ if $\mathcal{R}$ is a reduction with $\mathrm{src}(\mathcal{R}) = a$. $\mathcal{R} : a \twoheadrightarrow_{\mathfrak{A}} b$ means that $\mathcal{R}$ is a (finite) reduction with $\mathrm{src}(\mathcal{R}) = a$ and $\mathrm{tgt}(\mathcal{R}) = b$. We write $a \twoheadrightarrow_{\mathfrak{A}} b$ if there exists a $\mathcal{R}$ such that $\mathcal{R} : a \twoheadrightarrow_{\mathfrak{A}} b$. In all cases, the subscript is omitted if clear from the context.

There is a small semantic difference between $\twoheadrightarrow$ and $\rightarrow^*$: $a \twoheadrightarrow b$ means that there is a reduction between a and b, while $\rightarrow^*$ is the reflexive, transitive closure of the one-step rewrite relation. Of course, the two notions are equivalent:

Lemma 2.2.3. *$a \twoheadrightarrow_{\mathfrak{A}} b$ if and only if $a \rightarrow^*_{\mathfrak{A}} b$.*

Proof. ($\Rightarrow$): By induction on the length of the reduction $\mathcal{R}$. If $\mathcal{R}$ is the empty reduction, then $a = b$ and we have that $a \rightarrow^* b$ by reflexivity. Otherwise, $\mathcal{R} = \varphi\,;\mathcal{R}'$, where $\mathrm{tgt}(\varphi) = \mathrm{src}(\mathcal{R}')$. Let $c = \mathrm{tgt}(\varphi)$. We know $a \rightarrow c$, and thus $a \rightarrow^* c$. By the induction hypothesis, $c \rightarrow^* b$. So, by transitivity, $a \rightarrow^* b$.

($\Leftarrow$): By induction on the derivation of $a \rightarrow^* b$. In the base case either $a \rightarrow b$ or $a = b$, and in both cases $a \twoheadrightarrow b$ follows immediately. If there is a c such that $a \rightarrow^* c$ and $c \rightarrow^* b$, then the induction hypothesis yields two composable reductions. □

We use the notation $\langle\!\langle a\rangle\!\rangle_{\mathfrak{A}}$, or just $\langle\!\langle a\rangle\!\rangle$ if $\mathfrak{A}$ is clear from the context, to denote the set of objects which are reachable from a in the rewrite system $\mathfrak{A}$, that is:

$$\langle\!\langle a\rangle\!\rangle_{\mathfrak{A}} := \{b \mid a \twoheadrightarrow_{\mathfrak{A}} b\}.$$

An ARS is a directed graph, where A are its nodes, Φ its edges and reductions are its paths. We can therefore use the following graph-theoretic result, which is well-established in the literature:

Lemma 2.2.4 (König's Lemma)**.** *A finitely branching, connected graph is infinite if and only if it has an infinite simple path.*

For some abstract results, it is convenient to consider finite reductions as steps. In the following definitions we set up an ARS which has as its steps the finite reductions of another ARS.

Definition 2.2.5 (cf. [53, Def. 8.2.8])**.** An Abstract Rewrite System with Composition (ARSC)[1] is a structure $\mathfrak{A} = \langle A, \Phi, \mathrm{src}, \mathrm{tgt}, 1, ; \rangle$ where:

- $\mathfrak{B} = \langle A, \Phi, \mathrm{src}, \mathrm{tgt} \rangle$ is an ARS,
- 1 is a (total) function from A to Φ such that $1_a : a \to a$ and
- $;$ is a (total) function from pairs $\langle \varphi, \psi \rangle \in \Phi^2$ with $\mathrm{tgt}(\varphi) = \mathrm{src}(\psi)$ to Φ,

such that the following identities hold:

$$1_a \,;\, \varphi = \varphi$$
$$\varphi \,;\, 1_b = \varphi$$
$$(\varphi \,;\, \psi) \,;\, \chi = \varphi \,;\, (\psi \,;\, \chi)$$

where $a = \mathrm{src}(\varphi)$ and $b = \mathrm{tgt}(\varphi)$.

We will usually omit the subscript of the 1. A step in the range of 1 will be called an *empty step*.

To each ARS, we associate a canonical ARSC in the following way:

Definition 2.2.6. Let $\mathfrak{A} = \langle A, \Phi, \mathrm{src}, \mathrm{tgt} \rangle$ be an ARS. Its *reflexive, transitive closure* $\mathfrak{A}^* = \langle A^*, \Phi^*, \mathrm{src}^*, \mathrm{tgt}^*, 1, ; \rangle$ is defined by:

- $A^* = A$;
- Φ^* is the set of *finite* $\mathfrak{A}$-reductions;
- src^* and tgt^* return the source and target of the reductions;
- 1 is the function which maps each object a to the empty reduction from a to a;
- $;$ is the concatenation function for reductions.

Lemma 2.2.7. *For each* ARS $\mathfrak{R}$*,* $\mathfrak{R}^*$ *is an* ARSC*.*

Proof. The requirements of Def. 2.2.5 are trivially satisfied. □

Lemma 2.2.8. $a \to_{\mathfrak{A}^*} b$ *if and only if* $a \twoheadrightarrow_{\mathfrak{A}} b$*.*

[1] What we call *Abstract Rewrite System with Composition* is called a *category* in [53].

Proof. Let $\mathfrak{A} = \langle A, \Phi, \mathrm{src}, \mathrm{tgt}\rangle$ be an ARS and $\mathfrak{A}^* = \langle A^*, \Phi^*, \mathrm{src}^*, \mathrm{tgt}^*, 1, ;\rangle$. The equivalence between $a \to_{\mathfrak{A}^*} b$ and $a \twoheadrightarrow_{\mathfrak{A}} b$ is trivial, because Φ^* is the set of finite reductions. The only remark we have to make is that $a \twoheadrightarrow_{\mathfrak{A}} b$ if and only if there is a *finite* reduction from a to b, because infinite reductions don't have a target. □

We have now introduced three ways to go from steps to reductions. All three are equivalent:

Corollary 2.2.9. $a \to_{\mathfrak{A}^*} b \Leftrightarrow a \twoheadrightarrow_{\mathfrak{A}} b \Leftrightarrow a \to^*_{\mathfrak{A}} b$.

Proof. Directly from Lemma 2.2.3 and Lemma 2.2.8. □

2.2.3 Properties of rewrite systems

The following properties of rewrite systems are frequently distinguished:

Definition 2.2.10. Let $\mathfrak{A} = \langle A, \Phi, \mathrm{src}, \mathrm{tgt}\rangle$ be an (abstract) rewrite system, and $a \in A$ an object.

(i) a is *irreducible*, if $\neg\exists b : a \to_{\mathfrak{A}} b$; in this case, we will say that a is a *normal form*;

(ii) a has the *diamond property* if $a \to_{\mathfrak{A}} b \;\&\; a \to_{\mathfrak{A}} c \Rightarrow \exists d : b \to_{\mathfrak{A}} d \;\&\; c \to_{\mathfrak{A}} d$;

(iii) a is *locally confluent* if $a \to_{\mathfrak{A}} b \;\&\; a \to_{\mathfrak{A}} c \Rightarrow \exists d.b \twoheadrightarrow_{\mathfrak{A}} d \;\&\; c \twoheadrightarrow_{\mathfrak{A}} d$;

(iv) a is *confluent* if $a \twoheadrightarrow_{\mathfrak{A}} b \;\&\; a \twoheadrightarrow_{\mathfrak{A}} c \Rightarrow \exists d.b \twoheadrightarrow_{\mathfrak{A}} d \;\&\; c \twoheadrightarrow_{\mathfrak{A}} d$;

(v) a is *normalizing* (also called weakly normalizing) if there is a normal form b such that $a \twoheadrightarrow_{\mathfrak{A}} b$;

(vi) a is *terminating* (also called strongly normalizing) if there are no infinite $\mathfrak{A}$-reductions from a.

$\mathfrak{A}$ is/has X, where X is one of diamond property, locally confluent, confluent, normalizing or terminating, if for all $a \in A$, a is/has X.

The following correspondences are well-known in the literature (see for example [53, Ch. 1]):

Lemma 2.2.11. *Let* $\mathfrak{A}$ *be an* ARS.

(i) (Newman's Lemma) *If* $\mathfrak{A}$ *is locally confluent and terminating, then* $\mathfrak{A}$ *is confluent.*

(ii) $\mathfrak{A}$ *is confluent if and only if* $\mathfrak{A}^*$ *has the diamond property.*

For a confluent and terminating rewrite system $\mathfrak{A}$, we will write $a{\downarrow_{\mathfrak{A}}}$ for the unique normal form b such that $a \twoheadrightarrow_{\mathfrak{A}} b$.

2.3 The simply typed λ-calculus

The style of Higher-order Rewriting that we employ uses β-reduction (and restricted η-expansion) in the simply typed λ-calculus as meta-theory (called substitution calculus in [39]). In this section we briefly introduce this calculus, and define proof terms and multisteps for it.

2.3.1 λ-Terms and β-rewriting

The set Type of simple types (in the following just called *types*), is generated from a set of base types by the type constructor $\to$, which associates to the right. The set of types is usually left implicit. Each type α can be written as

$$\tau = \tau_1 \to \cdots \to \tau_n \to \tau_0$$

where, for $1 \leq i \leq n$, τ_i is a type and τ_0 is a base type. The *arity* of this type, written $\mathrm{ar}(\tau)$, is n.

A *typed set* is a set S with associated type function $\mathtt{t}_S : S \to \mathrm{Type}$. For a typed set S, we write $a : \tau \in S$, (or just $a : \tau$ if S is clear from the context), for $a \in S$ and $\mathtt{t}_S(a) = \tau$. We fix in advance a countably infinite typed set Var of *variables*. In the following, x, y, z $(, u, w)$ range over these variables.

A *signature* is a finite or countably infinite typed set, disjoint from Var, the elements of which are called *function symbols*. The arity of a function symbol is equal to the arity of its type. In the following, f, g, h range over arbitrary function symbols, and a, b, c over function symbols of arity 0 (also called *constants*. On the other hand, sans-serif letters and words (for example $\mathsf{a}, \mathsf{b}, \mathsf{f}, \mathsf{g}, \mathsf{let}, \mathsf{map}$) are used to denote *specific* function symbols and constants (these are mainly used in examples).

Let a signature Σ be given. We define the typed set $\Lambda(\Sigma)$ of simply typed λ-terms over Σ to be the smallest set such that:

- if $x : \tau \in \mathrm{Var}$, then $x : \tau \in \Lambda(\Sigma)$;
- if $f : \tau \in \Sigma$, then $f : \tau \in \Lambda(\Sigma)$;
- if $M : \tau_1 \to \tau_2, N : \tau_2 \in \Lambda(\Sigma)$, then $(MN) : \tau_2 \in \Lambda(\Sigma)$;
- if $x : \tau_1 \in \mathrm{Var}$ and $M : \beta \in \Lambda(\Sigma)$, then $\lambda x.M : \tau_1 \to \tau_2 \in \Lambda(\Sigma)$.

With $\mathcal{S}ym(M)$ we denote the set of function symbols which occur in the λ-term M. A variable in a term is bound by the first matching λ above it. The free variables are variables which are not bound. The set of free variables of a term M, denoted $\mathrm{FV}(M)$, is inductively defined as follows:

$$\begin{aligned} \mathrm{FV}(x) &:= x \\ \mathrm{FV}(f) &:= \varnothing \\ \mathrm{FV}(M_1 M_2) &:= \mathrm{FV}(M_1) \cup \mathrm{FV}(M_2) \\ \mathrm{FV}(\lambda x.M_0) &:= \mathrm{FV}(M_0) - \{x\} \end{aligned}$$

A substitution is a mapping from variables to λ-terms of the same type. Applying a substitution σ to a term M, denoted M^σ (or $M\sigma$ if this is more clear), is defined as follows:

$$\begin{aligned} x^\sigma &:= \sigma(x) && \text{if } x \in Dom(\sigma) \\ x^\sigma &:= x && \text{otherwise} \\ f^\sigma &:= f \\ (MN)^\sigma &:= M^\sigma N^\sigma \\ (\lambda x.M)^\sigma &:= \lambda y.M[x \mapsto y]^\sigma && \text{where } y \text{ is the first fresh variable} \end{aligned}$$

A context C is a term over an extended signature which includes a special symbol $\Box$, called a *hole*, which can be of any type. A context which contains n $\Box$'s is called an n-ary context. If C is an n-ary context, then the expression $C[M_1, \ldots, M_n]$ is defined when M_i has the same type as the ith $\Box$ from the left, for each $1 \leq i \leq n$, and denotes C with the ith $\Box$ from the left literally replaced by M_i, for each i.

Although substitutions and contexts serve a similar purpose, there is one crucial difference: a λ-expression in a context may bind a free variable of a term which is replaced for a hole. On the other hand, when a substitution is applied, it is made sure, by renaming bound variables, that a variable in the term can never bind a free variable in a term that is substituted for a variable. To illustrate the difference, consider the term $M = \lambda x.y$ on the one hand, and the context $C = \lambda x.\Box$ on the other. Then:

$$M[y \mapsto x] = \lambda z.x \quad \text{but} \quad C[x] = \lambda x.x.$$

In the left case, the free variable x of the substitution remains free, while in the right case the variable becomes bound by the leading abstraction.

The three most important equivalence relations between λ-terms are α-, β- and η-equivalence, which are generated by the following equations:

$$\begin{array}{lrcll} \alpha: & \lambda x.M & =_\alpha & \lambda y.M[x \mapsto y] & \text{if } y \notin \mathrm{FV}(M) \\ \beta: & (\lambda x.M)N & =_\beta & M[x \mapsto N] & \\ \eta: & \lambda x.Mx & =_\eta & M & \text{if } x \notin \mathrm{FV}(M) \end{array}$$

As usual, α-equivalent λ-terms are identified. We employ the Variable Convention [4], which states that, in a given context all bound variables are named differently from other bound variables and the free variables.

The β-equation can be oriented from left to right, resulting in the following rewrite relation:

$$C[(\lambda x.M)N] \to_\beta C[M[x \mapsto N]].$$

It is well-known that β-reduction is terminating on simply typed terms [52]; see also Sect. 4.6.2.

2.3.2 Proof terms for β-multisteps

Next, we define β-multisteps, that is, steps in which zero, one or more than one β-redexes can be contracted at the same time. We define multisteps by

$$\frac{x \in \mathrm{Var}}{x : x \multimap_\beta x}\ \mathsf{var} \qquad \frac{f \in \Sigma}{f : f \multimap_\beta f}\ \mathsf{fun}$$

$$\frac{\gamma : M \multimap_\beta N}{\lambda x.\gamma : \lambda x.M \multimap_\beta \lambda x.N}\ \mathsf{abs} \qquad \frac{\gamma : M \multimap_\beta P \quad \delta : N \multimap_\beta Q}{\gamma\delta : MN \multimap_\beta PQ}\ \mathsf{app}$$

$$\frac{\gamma : M \multimap_\beta P \quad \delta : N \multimap_\beta Q}{\beta_x(\gamma, \delta) : (\lambda x.M)N \multimap_\beta P[x \mapsto Q]}\ \mathsf{beta}$$

Table 2.1: Proof terms for β-steps.

means of an inference system, and also define *proof terms*, which are explicit witnesses of steps. Our proof terms are equal to Hilken's [17], except that we use named variables rather than De Bruijn indices. We postpone a more detailed discussion of the advantages and potential problems of proof terms to Sect. 2.4.3, where multisteps and proof terms for higher-order rewriting are defined.

A β-proof term is a λ-term (see pag. 16) with an extra constructor which represents β-steps. Let a signature Σ be given. The typed set $Pt(\Sigma)$ of proof terms over Σ is defined to be the smallest set such that:

- if $x : \tau \in \mathrm{Var}$, then $x : \tau \in Pt(\Sigma)$;
- if $f : \tau \in \Sigma$, then $f : \tau \in Pt(\Sigma)$;
- if $\gamma : \tau_1 \to \tau_2, \delta : \tau_1 \in Pt(\Sigma)$, then $(\gamma\delta) : \tau_2 \in Pt(\Sigma)$;
- if $x : \tau_1 \in \mathrm{Var}$ and $\gamma : \tau_2 \in Pt(\Sigma)$, then $\lambda x.\gamma : \tau_1 \to \tau_2 \in Pt(\Sigma)$;
- if $x : \tau_1 \in \mathrm{Var}$ and $\gamma : \tau_2, \delta : \tau_1 \in Pt(\Sigma)$, then $\beta_x(\gamma, \delta) : \tau_2 \in Pt(\Sigma)$.

β-Multisteps are defined by means of the inference system in Table 2.1, together with the witnessing β-proof terms. The inference system has judgements of the form $\gamma : M \multimap_\beta N$, which mean that γ is a *proof term* which *witnesses* that λ-term N can be reached by performing one β-multistep starting from M. Consider the following example:

Example 2.3.1. The following inference proves that $\beta_x(\mathsf{f}(x), \mathsf{a})$ witnesses the step $(\lambda x.\mathsf{f}(x))\mathsf{a} \multimap_\beta \mathsf{f}(\mathsf{a})$:

$$\dfrac{\dfrac{\dfrac{}{\mathsf{f} : \mathsf{f} \multimap_\beta \mathsf{f}}\ \mathsf{fun} \qquad \dfrac{}{x : x \multimap_\beta x}\ \mathsf{var}}{\mathsf{f}(x) : \mathsf{f}(x) \multimap_\beta \mathsf{f}(x)}\ \mathsf{app} \qquad \dfrac{}{\mathsf{a} : \mathsf{a} \multimap_\beta \mathsf{a}}\ \mathsf{fun}}{\beta_x(\mathsf{f}(x), \mathsf{a}) : (\lambda x.\mathsf{f}(x))\mathsf{a} \multimap_\beta \mathsf{f}(\mathsf{a})}\ \mathsf{beta}$$

Note that the inference system allows proof terms to contain zero, one or more than one β_x-symbols. Such proof terms correspond to multisteps that contract

zero, one or more than one β-redexes, respectively.[2] We use the inference system to define the β-multistep rewrite relation in the following way:

$$M \multimap_\beta N \text{ if there is a } \gamma \text{ such that } \gamma : M \multimap_\beta N.$$

Example 2.3.2. The following inference proves that $\beta_x(x, \beta_y(y, \mathsf{a}))$ witnesses the multistep $(\lambda x.x)((\lambda y.y)\mathsf{a}) \multimap \mathsf{a}$:

$$\dfrac{\dfrac{}{x : x \multimap_\beta x}\,\mathsf{var} \qquad \dfrac{\dfrac{}{y : y \multimap_\beta y}\,\mathsf{var} \qquad \dfrac{}{\mathsf{a} : \mathsf{a} \multimap_\beta \mathsf{a}}\,\mathsf{fun}}{\beta_y(y, \mathsf{a}) : (\lambda y.y)\mathsf{a} \multimap \mathsf{a}}\,\mathsf{beta}}{\beta_x(x, \beta_y(y, \mathsf{a})) : (\lambda x.x)((\lambda y.y)\mathsf{a}) \multimap_\beta \mathsf{a}}\,\mathsf{beta}$$

In the following proposition it is established that $\multimap_\beta^*$ and $\twoheadrightarrow_\beta$ are actually the same rewrite relation between λ-terms:

Proposition 2.3.3. $\to_\beta \subseteq \multimap_\beta \subseteq \to_\beta^*$.

Proof. $\to_\beta \subseteq \multimap_\beta$ is shown by induction on the context of the $\to_\beta$-step. $\multimap_\beta \subseteq \to_\beta^*$ is show by induction on the inference with $\gamma : M \multimap_\beta N$ as a conclusion. □

We now construct an ARS from the simply typed λ-calculus:

Definition 2.3.4. Let Σ be a signature. The ARS $\overline{\lambda_\beta}(\Sigma)$ is defined as $\overline{\lambda_\beta}(\Sigma) = \langle A, \Phi, \mathrm{src}, \mathrm{tgt}\rangle$ where:

- A is the set of typed λ-terms over Σ;
- Φ is the set of proof terms over Σ;
- $\mathrm{src}(\gamma) = M$ and $\mathrm{tgt}(\gamma) = N$ if and only if $\gamma : M \multimap_\beta N$.

As said above, β-Proof terms can be seen as λ-terms with an extra term constructor $\beta_x(\cdot,\cdot)$. Substitutions can thus be defined for β-proof terms analogously to λ-terms. A (proof term) substitution σ is a mapping from variables to proof terms of the same type. We write $\mathrm{src}(\sigma)$ and $\mathrm{tgt}(\sigma)$ for the substitutions such that

$$(\mathrm{src}(\sigma))(x) := \mathrm{src}(\sigma(x)) \quad \text{and} \quad (\mathrm{tgt}(\sigma))(x) := \mathrm{tgt}(\sigma(x)).$$

Applying a substitution σ to a β-proof term γ is defined as follows:

$$\begin{aligned}
x^\sigma &:= \sigma(x) && \text{if } x \in Dom(\sigma)\\
x^\sigma &:= x && \text{otherwise}\\
f^\sigma &:= f &&\\
(MN)^\sigma &:= M^\sigma N^\sigma &&\\
(\lambda x.M)^\sigma &:= \lambda y.M[x \mapsto y]^\sigma && \text{where } y \text{ is the first fresh variable}\\
\beta_x(\gamma,\delta)^\sigma &:= \beta_y(\gamma[x \mapsto y]^\sigma, \delta^\sigma) && \text{where } y \text{ is the first fresh variable}
\end{aligned}$$

[2] Multisteps are also sometimes called *parallel steps*. We follow the terminology of [43], where parallel steps are distinguished from multisteps, in that the latter may contract two nested redexes in one go, while the former may not.

Using this definition of substitution, we can define α-, β- and η-equivalence analogous to α-, β- and η-equivalence of λ-terms. We can now show that terms and proof terms play nicely together with respect to these equivalences.

Lemma 2.3.5. *Let $\gamma\colon M \multimap\!\!\to_\beta N$ be a proof term and σ a proof term substitution. Then $\gamma^\sigma\colon M^{\mathrm{src}(\sigma)} \multimap\!\!\to_\beta N^{\mathrm{tgt}(\sigma)}$.*

Proof. By induction on the structure of γ, using the observation that the same variables occur in the proof term and its source, so that "the first fresh variable" is the same variable in both cases. □

Proposition 2.3.6. *Let $\gamma\colon M \multimap\!\!\to_\beta M$ and $\gamma'\colon M' \multimap\!\!\to_\beta N'$ be proof terms.*

(i) If $\gamma =_\alpha \gamma'$, then: $M =_\alpha M'$ and $N =_\alpha N'$.
(ii) If $\gamma =_\beta \gamma'$, then: $M =_\beta M'$ and $N =_\beta N'$.
(iii) If $\gamma =_\eta \gamma'$, then: $M =_\eta M'$ and $N =_\eta N'$.

Proof. All three items follow easily from Lemma 2.3.5. □

2.3.3 Positions and tracing

λ-Terms may be considered as trees with function symbols, variables, applications and abstractions as nodes. Every node of the tree is uniquely determined by the path from the root of the tree to the node in question. Positions are representations of such paths, and therefore representations of specific places within the term.

Definition 2.3.7.

(i) A λ-position p is a finite sequence over $\{1, 2\}$.
λ-Position composition is denoted by juxtaposition, that is $pq = p\,;q$.
For a set of positions P and position q, we use the following notation:

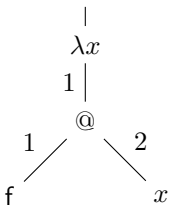
$$q\,;P := \{qp \mid p \in P\}$$

(ii) Let M be a λ-term. The set of λ-positions of M, denoted $\mathcal{P}os(M)$ and the symbol at λ-position p, denoted $M(p)$:

- If $M = \Box$, then:
  $\mathcal{P}os(M) := \varnothing$.
- If $M = x$, then:
  $\mathcal{P}os(M) := \{\varepsilon\}$
  $M(\varepsilon) := x$
- If $M = f$, then:
  $\mathcal{P}os(M) := \{\varepsilon\}$
  $M(\varepsilon) := f$

- If $M = M_1 M_2$, then:
  $\mathcal{P}os(M) := \{\varepsilon\} \cup (1\,;\mathcal{P}os(M_1)) \cup (2\,;\mathcal{P}os(M_2))$
  $M(\varepsilon) := @$ and $M(ip) := M_i(p)$, for $i \in \{1,2\}$.
- If $M = \lambda x.M_0$, then:
  $\mathcal{P}os(M) := \{\varepsilon\} \cup (1\,;\mathcal{P}os(M_0))$
  $M(\varepsilon) := \lambda x$, $M(1p) := M_0(p)$.

Example 2.3.8. Consider the term $M = \lambda x.\mathsf{f}x$. This term is represented by the following tree:



Positions of occurences of symbols can be found by composing the numbers found on the path from the root to the occurrence. Thus $\mathcal{P}os(M) = \{\epsilon, 1, 11, 12\}$ and:

- $M(\varepsilon) = \lambda x$;
- $M(1) = @$;
- $M(11) = \mathsf{f}$; and
- $M(12) = x$.

We define two orderings on positions. We write $p \sqsubseteq q$ if p is a prefix of q, that is there is a position p' such that $p\,;p' = q$. Furthermore, the ordering $\leq_{\text{lex}}$ is the lexicographic extension of $<$. We extend this second ordering to an ordering of sets of positions as follows:

$$P \leq_{\text{lex}} Q \text{ if } p \leq_{\text{lex}} q \text{ for all } p \in P, q \in Q.$$

In practice, the above orderings mean the following: if $p \sqsubseteq q$, then position p is closer to the root than q; if $p \leq_{\text{lex}} q$, then position p is closer to the root than or to the left of q.

Next we define a trace relation, which keeps track of how symbols move when a step is done. Consider for instance the β-step $(\lambda x.x)\mathsf{a} \to_\beta \mathsf{a}$. The position of the a in the source of this step is 2, while the position of the a in the target is ε. Still, in a sense, the a in the target is the same symbol as the a in the source. The trace relation formalizes this idea by stating exactly which positions in the source and target of a step (or reduction) are 'the same'.

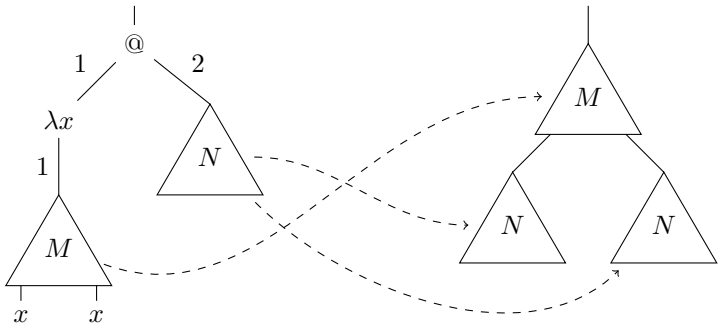


Figure 2.1: Visual representation of the λ-trace relation.

Definition 2.3.9. The β-trace relation over a β-step $\gamma : M \multimap_\beta N$, denoted by $[\![\gamma]\!]_\beta$, is defined as follows:

$$\begin{aligned}
[\![x]\!]_\beta &:= \mathsf{id}_{\{\varepsilon\}} \\
[\![f]\!]_\beta &:= \mathsf{id}_{\{\varepsilon\}} \\
[\![\lambda x.\gamma]\!]_\beta &:= \lambda.[\![\gamma]\!]_\beta \\
[\![\gamma\delta]\!]_\beta &:= @([\![\gamma]\!]_\beta, [\![\delta]\!]_\beta) \\
[\![\beta_x(\gamma,\delta)]\!]_\beta &:= \beta_X([\![\gamma]\!]_\beta, [\![\delta]\!]_\beta)
\end{aligned}$$

where, for relations of positions R, R_1, R_2:

- $\lambda.R$ is notation for the relation which relates ε to ε and $1p$ to $1q$ iff $p\ R\ q$;
- $@(R_1, R_2)$ is notation for the relation which relates ε to ε and ip to iq iff $p\ R_i\ q$, for $i \in \{1, 2\}$; and
- $\beta_P(R_1, R_2)$ is notation for the relation which relates $11q$ to q' iff $q\ R_1\ q'$ and $q \notin P$ and $2q$ to pq', for any $p \in P$, iff $q\ R_2\ q'$.
- in the last equation, $X = \{p \mid M(p) = x\}$, where $M = \mathrm{tgt}(\gamma)$;

The β-trace relations of a β-reduction $\mathcal{M}$, denoted $[\![\mathcal{M}]\!]_\beta$, is the composition of the β-trace relations of its respective steps.

Note that the well-definedness of X in the fourth item of the definition above depends on the variable convention (see pag. 17), which ensures that variables which occur within the scope of another variable are named differently. In Fig. 2.1 a visual representation of the λ-trace relation is given.

Example 2.3.10. Let the β-proofterm $\gamma = \beta_x(\mathsf{f}x, \mathsf{a}) : (\lambda x.\mathsf{f}x)\mathsf{a} \multimap_\beta \mathsf{fa}$. The trace relation $[\![\gamma]\!]$ is as follows:

$$11 \mapsto \epsilon \qquad 111 \mapsto 1 \qquad 2 \mapsto 2$$

The trace relation has the property that there is at most one position in the source of the β-step that traces to a given position in the target of the step; for linear λ-terms – that is λ-terms that have the property that in each subterm of the form $\lambda x.M$, x occurs exactly once in M – the converse is also true. This is formally stated in the next lemma:

Lemma 2.3.11.

(i) If $p \, [\![\mathcal{M}\rangle\!\rangle \, q$ *and* $p' \, [\![\mathcal{M}\rangle\!\rangle \, q$*, then* $p = p'$*.*

(ii) If $\mathrm{src}(\mathcal{M})$ *is linear,* $p \, [\![\mathcal{M}\rangle\!\rangle \, q$ *and* $p \, [\![\mathcal{M}\rangle\!\rangle \, q'$*, then* $q = q'$*.*

Proof. Both items are proved by induction on the length of $\mathcal{M}$, using for (ii) the fact that, for some β-step γ, $\mathrm{tgt}(\gamma)$ is linear if $\mathrm{src}(\gamma)$ is. □

2.4 Higher-order Rewrite Systems

This thesis is about Higher-order Rewrite Systems (HRS), as introduced by Nipkow [36, 37, 31]. In fact, we consider HRSs as HORSs [39] with the simply typed λ-calculus as substitution calculus.

2.4.1 Terms and the rewrite relation

Preterms. Let Σ be a signature. A Σ-*preterm* (or just "preterm" if Σ is clear from the context or not important) is a typed λ-term over Σ (see Sect. 2.3). Also, Σ-precontexts and Σ-presubstitutions are defined to be λ-contexts and λ-substitutions over Σ, respectively.

We want to consider preterms (and precontexts and presubstitutions) *modulo* β- and η-equivalence. Unique representatives of $\beta\eta$-equivalence classes can be computed using the union of β-reduction (presented in Sect. 2.3) and restricted η-expansion:

Definition 2.4.1 (Restricted η-expansion)**.** The relation of restricted η-expansion is the smallest relation such that $C[M] \to_{\overline{\eta}} C[\lambda x.Mx]$ if the following conditions are satisfied:

(i) M is of functional type;

(ii) x does not occur free in M;

(iii) M is not of the form $\lambda y.M_0$; and

(iv) the hole of C does not occur as the left part of an application.

Conditions (iii) and (iv) of the above definition are equivalent to the condition that a restricted η-expansion step can never create new β-redexes. Restricted η-expansion has the following important properties:

Lemma 2.4.2.

(i) The relation $\rightarrow_{\overline{\eta}}$ is confluent and terminating.

(ii) The set of $\overline{\eta}$-normal forms is closed under β-reduction.

(iii) Every $\beta\overline{\eta}$-equivalence class has a unique $\beta\overline{\eta}$-normal form.

Proof. See [53, Pag. 594, 595]. □

Terms. As written above, we want to consider $\beta\eta$-equivalence classes of preterms. By Lemma 2.4.2(iii) we can take $\beta\overline{\eta}$-normal forms as the unique representatives of the $\beta\eta$-equivalence classes and define:

Definition 2.4.3. Let Σ be a set of simply typed function symbols.

(i) A Σ-*term* is a Σ-preterm in $\beta\overline{\eta}$-normal form.

(ii) A Σ-*context* is a Σ-precontext in $\beta\overline{\eta}$-normal form.

(iii) A Σ-*substitution* is a Σ-presubstitution of which the codomain consists of Σ-terms.

As usual, the signature Σ will be omitted if clear from the context. Unless otherwise indicated, in the following the lowercase roman letters s, t will range over terms, the capital roman letters C, D, E will range over contexts and σ, τ will range over substitutions. In the following, a term of the form $as_1 \dots s_n$, where a is either a function symbol or a variable, will be written as $a(s_1, \dots, s_n)$ or even $a(\overline{s})$; a term of the form $\lambda x_1 \dots . \lambda x_n . s$ will be written as $\lambda x_1 \dots x_n . s$ or $\lambda \overline{x} . s$.

Because of Lemma 2.4.2 (ii), we can forget about restricted η-expansion altogether if we make sure that all 'input' terms are in $\overline{\eta}$-normal form. In the following, this will be implicitly done.

Contexts are often used in definitions and proofs to 'focus' on the important part of a term. However, they can have nasty nesting behavior which is undesired in such cases. Therefore, often contexts of a simple form are used. We define: a *base context* is a context in which the holes do no occur in the arguments of other holes.

Patterns. We restrict the left-hand sides of rules to be *patterns*. Patterns, as introduced by Miller [35], are terms in which every free variable has only distinct bound variables as arguments. Patterns have first-order-like properties with respect to unification: unification for patterns (and thus matching) is decidable, and if two patterns p, q are unifiable, then a unique most general unifier exists. Usually we will also require the left-hand sides to be linear (each free variable occurs at most once in it) and fully-extended (free variables have *all* bound variables in scope as argument). The notions of pattern, linear and fully extended pattern are formally defined below:

Definition 2.4.4 (Pattern)**.** Let Σ be a signature and $\overline{x}$ be a sequence of variables.

(i) A Σ-term s is a $(\Sigma, \overline{x})$-pattern if:

- $s = y(s_1, \ldots, s_n)$, where $y \notin \overline{x}$ and $s_1, \ldots, s_n$ is η-equivalent to a sequence of pairwise distinct variables from $\overline{x}$; or
- $s = a(s_1, \ldots, s_n)$, where $a \in \overline{x} \cup \Sigma$ and $s_1, \ldots, s_n$ are $(\Sigma, \overline{x})$-patterns; or
- $s = \lambda y.s_0$ and s_0 is a $(\Sigma, \overline{x}y)$-pattern.

(ii) A Σ-term s is *linear outside* $\overline{x}$ if each free variable not in $\overline{x}$ occurs in it at most once.

A Σ-term s is a *fully extended* $(\Sigma, \overline{x})$-pattern, if it is a $(\Sigma, \overline{x})$-pattern, and in the first case of item (i), it is always the case that $s_1, \ldots, s_n =_\eta \overline{x}$.

A Σ-term s is a *local* $(\Sigma, \overline{x})$-pattern if it is a fully extended $(\Sigma, \overline{x})$-pattern which is linear outside $\overline{x}$.

The Σ-parameter will usually be clear from the context, and is then omitted. The $\overline{x}$-parameter is omitted if $\overline{x} = \varnothing$, that is "$(\Sigma, \varnothing)$-pattern" may be abbreviated to "Σ-pattern" or even just to "pattern".

Example 2.4.5.

- Examples of local patterns are: $\mathsf{f}(x)$, $\mathsf{g}(\lambda xy.\mathsf{f}(z(x,y)))$ and $\mathsf{h}(\lambda x.z(x))$. Consider, as an example, the derivation of the fact that $\mathsf{g}(\lambda xy.\mathsf{f}(z(x,y)))$ is a local pattern:
  - $\mathsf{g}(\lambda xy.\mathsf{f}(z(x,y)))$ is a local pattern by the second clause, because:
  - $\lambda xy.\mathsf{f}(z(x,y))$ is a local pattern by the third clause (twice), because:
  - $\mathsf{f}(z(x,y))$ is a local x,y-pattern by the second clause, because:
  - $z(x,y)$ is a local x,y-pattern by the first clause.
- Examples of non-local patterns are: $\mathsf{lam}(\lambda x.\mathsf{app}(z,x))$[3] (not fully extended, because the bound variable x does not occur as an argument of z) and $\mathsf{h}(\lambda x.z(x), \lambda x.z(x))$ (not linear, because the free variable z occurs twice in the term).
- Examples of non-patterns are: $\mathsf{g}(z(\mathsf{a}))$ and $\mathsf{g}(z(x))$. In the first term, the free variable z has a function symbol a as argument, and in the second term a free variable x. In both cases, this is not a bound variable, as required.

The requirements of linearity and fully-extendedness have a similar purpose as the requirement of linearity for left-hand sides of first-order TRSs: they keep matching 'local' (hence the name "local pattern") in the following sense: to match a non-linear pattern, two subterms outside of the pattern need to be checked for equality; to match a non-fully extended pattern, a subterm outside of the pattern needs to be checked for the non-occurrence of variables. Many

[3] This non-local pattern corresponds to the left-hand side of the η-reduction rule, $\lambda x.Mx \to M$. The fact that the bound variable x does not occur as an argument of the free variable z represents the condition that x may not occur free in M.

theorems of first-order rewriting only hold if all left-hand sides are left-linear. Similarly, a lot of theory of higher-order rewriting is only applicable to HRSs in which all of the left-hand sides are local patterns.

The rewrite relation. We are now ready to define Higher-order Rewrite Systems (HRSs) and the rewrite relation:

Definition 2.4.6.

(i) Let Σ be a signature. A Σ-*rule* is a pair $\langle \lambda\overline{x}.l_0, \lambda\overline{x}.r_0\rangle$ (usually written $\lambda\overline{x}.l_0 \to \lambda\overline{x}.r_0$) of closed Σ-terms of the same type, such that l_0 is a pattern of the form $l_0 = f(s_1, \ldots, s_n)$, and all of $\overline{x}$ occur free in l_0.

(ii) A *Higher-order Rewrite System* (HRS) $\mathfrak{H}$ is a structure $\langle \Sigma, R\rangle$ such that R is a set of Σ-rules.

$\mathfrak{H}$ is a linear/fully extended/local HRS, if for all $\lambda\overline{x}.l_0 \to r \in R$, l_0 is a linear/fully extended/local pattern.

(iii) The rewrite relation $\to_{\mathfrak{H}}$ is defined as follows:

$$s \to_{\mathfrak{H}} t \quad \text{if} \quad s =_\beta C[l] \text{ and } t =_\beta C[r], \text{ where } l \to r \in R.$$

Note that, because l_0 is required to be of the form $f(s_1, \ldots, s_n)$ in item (i) of the above definition, it must be the case that l_0, and hence r_0, are of base type.

The rewrite relation is extended to a rewrite relation on substitutions in the following way: $\sigma \to_{\mathfrak{H}} \tau$ if:

- $Dom(\sigma) = Dom(\tau)$;
- there is a $x \in Dom(\sigma)$ such that $\sigma(x) \to_{\mathfrak{H}} \tau(x)$; and
- for all $y \in Dom(\sigma)$ such that $x \neq y$, it holds that $\sigma(y) = \tau(y)$.

The subscript $\mathfrak{H}$ of $\to_{\mathfrak{H}}$ will be omitted if clear from the context.

In this dissertation, we will mainly restrict our attention to local HRSs. However, in some cases (auxiliary) results can be extended to a more general class of HRSs.

Example 2.4.7. Let the HRS $\mathfrak{Map}$, implementing the higher-order function map, be defined by:

$$\lambda z.\mathsf{map}(\lambda x.z(x), \mathsf{nil}) \to \lambda z.\mathsf{nil}$$
$$\lambda zuv.\mathsf{map}(\lambda x.z(x), \mathsf{cons}(u, v)) \to \lambda zuv.\mathsf{cons}(z(u), \mathsf{map}(\lambda x.z(x), v))$$

Here, cons and nil are the list constructors, namely list composition and the

empty list, respectively. A reduction of two $\mathfrak{Map}$-steps is the following:

$$\begin{aligned}
&\mathsf{map}(\lambda x.\mathsf{f}(x), \mathsf{cons}(\mathsf{a}, \mathsf{nil})) \\
=_\beta \quad &\underline{(\lambda zuv.\mathsf{map}(\lambda x.z(x), \mathsf{cons}(u, v)))}(\lambda x'.\mathsf{f}(x'), \mathsf{a}, \mathsf{nil}) \\
\to_{\mathfrak{Map}} \quad &\underline{(\lambda zuv.\mathsf{cons}(z(u), \mathsf{map}(\lambda x.z(x), v)))}(\lambda x'.\mathsf{f}(x'), \mathsf{a}, \mathsf{nil}) \\
=_\beta \quad &\mathsf{cons}(\mathsf{f}(\mathsf{a}), \mathsf{map}(\lambda x.\mathsf{f}(x), \mathsf{nil})) \\
=_\beta \quad &\mathsf{cons}(\mathsf{f}(\mathsf{a}), \underline{(\lambda z.\mathsf{map}(\lambda x.z(x), \mathsf{nil}))}(\lambda x'.\mathsf{f}(x'))) \\
\to_{\mathfrak{Map}} \quad &\mathsf{cons}(\mathsf{f}(\mathsf{a}), \underline{(\lambda z.\mathsf{nil})}(\lambda x'.\mathsf{f}(x'))) \\
=_\beta \quad &\mathsf{cons}(\mathsf{f}(\mathsf{a}), \mathsf{nil})
\end{aligned}$$

Note how the (underlined) left-hand sides are *literally* replaced by the (also underlined) right-hand sides.

Remark 2.4.8. In the literature, often different but essentially equivalent definitions of HRSs are given. For example, a much more first-order-like definition is frequently found. Rules are tuples $l \to r$ of the same base type, such that $\mathrm{FV}(r) \subseteq \mathrm{FV}(l)$ and $l = f(l_1, \ldots, l_n)$ is a pattern, and the rewrite relation is defined as follows:

> $s \to t$ if there is a context C and substitution σ such that $C[l^\sigma] =_\beta s$ and $C[r^\sigma] =_\beta t$, for some rule $l \to r$.

We can translate both approaches to each other by removing/adding the leading abstractions. Which one to use is essentially a matter of taste. Let A be the variant with leading abstractions, and B the variant without them. Some comparisons:

- Variant B needs substitutions to be defined. This means that substitution is a primitive operation on two levels: on the preterm level and on the term level. This is not a theoretical problem but mostly an aesthetic one.
- In variant A a step is performed by *literally* replacing the left-hand side of a rule by the right-hand side (modulo β-equivalence).

We chose variant A here mainly because it makes it a bit easier to define proof terms later on (see Sect. 2.4.3). Sometimes, however, we change to variant B if this aids clarity.

For the sake of clarity, we will often employ the following convention in examples:

Convention 2.4.9. We will often not write the leading abstractions, and leave them implicit. When a rule with free variables in the left-hand side is encountered, the reader may add leading binders for the free variables in the order in which they occur in the left-hand side. For example:

$$\mathsf{h}(y, \lambda x.z(x)) \to z(y) \quad \text{is equal to} \quad \lambda yz.\mathsf{h}(y, \lambda x.z(x)) \to \lambda yz.z(y).$$

Collapsing rules. Intuitively, a rewrite rule is collapsing if it can bring previously unconnected parts of the context (and substitution) together. In first-order TRSs, the only way for a rule to be collapsing is if it has a single variable as right-hand side. Higher-order rewrite rules can also be collapsing due to nesting. We formally define the notion of collapsing rule by first defining what a collapsing *term*[4] is.

Definition 2.4.10. A term s is *collapsing*, if one of the following applies:

- *(context-subst):* $s = x(s_1, \ldots, s_n)$, where x is a free variable; or
- *(subst-subst):* $s = C[x(s_1, \ldots, s_n)]$, and for some k,

$$s_k = \lambda\overline{z}.y(t_1, \ldots, t_m)$$

  where C is a context, x is a free variable, and y a free or bound variable.

A rewrite rule $\lambda\overline{x}.l \to \lambda\overline{x}.r$ is collapsing, if r is collapsing, and an HRS is collapsing, if at least one of its rules is.

Example 2.4.11.

- The rules $\lambda x.\mathsf{f}(x) \to \lambda x.x$ and $\lambda z.\mathsf{mu}(\lambda x.z(x)) \to \lambda z.z(\mathsf{mu}(\lambda x.z(x)))$ are collapsing due to the (context-subst) condition.

  Suppose that $\mathsf{g}(\mathsf{a})$ is a redex, and consider the step $\mathsf{g}(\mathsf{f}(\mathsf{a})) \to \mathsf{g}(\mathsf{a})$. This step creates the redex $\mathsf{g}(\mathsf{a})$ without creating one of its function symbols.
- The rule $\lambda yz.\mathsf{g}(\lambda x.z(x), y) \to \lambda yz.\mathsf{f}(z(y))$ is collapsing due to the (subst-subst) condition.

  Suppose that $\mathsf{f}(\mathsf{a})$ is a redex. Then the step $\mathsf{g}(\lambda x.\mathsf{f}(x), \mathsf{a}) \to \mathsf{f}(\mathsf{a})$ creates the redex $\mathsf{f}(\mathsf{a})$ without creating one of its function symbols.
- The rule $\lambda yz.\mathsf{app}(\mathsf{lam}(\lambda x.z(x)), y) \to \lambda yz.z(y)$ is collapsing due to both the (context-subst) and the (subst-subst) conditions.

Termination. Mainly due to the possibility of nested variables, proving termination of HRSs is much more difficult than proving termination of first-order TRSs. Many well-known first-order termination techniques cannot be generalized to the higher-order case, or only in a much weaker form. In Chapter 4, we prove the property of Finite Family Developments, which is usable in some cases to prove termination of HRSs. Here we just mention the following convenient lemma:

Lemma 2.4.12 (Right-hand side lemma)**.** *Let $\mathfrak{H} = \langle \Sigma, R \rangle$ be an HRS. $\mathfrak{H}$ is terminating if and only if r^σ is terminating for every $\lambda\overline{x}.l \to \lambda\overline{x}.r \in R$ and terminating substitution σ.*

Proof. This is Lemma 8 of [41]. □

[4] Although the name "collapsing" is arguably a bad name for a property of a static concept like "term".

Let $\mathfrak{H} = \langle \Sigma, R \rangle$ and $\rho : l \to r \in R$.

$$\frac{\varphi_1 : s_1 \mathrel{-\!\!\circ\!\!\twoheadrightarrow_{\mathfrak{H}}} t_1 \cdots \varphi_n : s_n \mathrel{-\!\!\circ\!\!\twoheadrightarrow_{\mathfrak{H}}} t_n}{x(\varphi_1, \ldots, \varphi_n) : x(s_1, \ldots, s_n) \mathrel{-\!\!\circ\!\!\twoheadrightarrow_{\mathfrak{H}}} x(s_1, \ldots, s_n)}\ \mathsf{var}$$

$$\frac{\varphi_1 : s_1 \mathrel{-\!\!\circ\!\!\twoheadrightarrow_{\mathfrak{H}}} t_1 \cdots \varphi_n : s_n \mathrel{-\!\!\circ\!\!\twoheadrightarrow_{\mathfrak{H}}} t_n}{f(\varphi_1, \ldots, \varphi_n) : f(s_1, \ldots, s_n) \mathrel{-\!\!\circ\!\!\twoheadrightarrow_{\mathfrak{H}}} f(s_1, \ldots, s_n)}\ \mathsf{fun}$$

$$\frac{\varphi : s \mathrel{-\!\!\circ\!\!\twoheadrightarrow} \mathfrak{H}t}{\lambda x.\varphi : \lambda x.s \mathrel{-\!\!\circ\!\!\twoheadrightarrow_{\mathfrak{H}}} \lambda x.t}\ \mathsf{abs}$$

$$\frac{\varphi_1 : s_1 \mathrel{-\!\!\circ\!\!\twoheadrightarrow_{\mathfrak{H}}} t_1 \cdots \varphi_n : s_n \mathrel{-\!\!\circ\!\!\twoheadrightarrow_{\mathfrak{H}}} t_n}{\rho(\varphi_1, \ldots, \varphi_n) : l(s_1, \ldots, s_n) \mathrel{-\!\!\circ\!\!\to^*_{\mathfrak{H}}} r(t_1, \ldots, t_n)}\ \mathsf{rule}$$

$$\frac{\varphi : s \mathrel{-\!\!\circ\!\!\twoheadrightarrow_{\mathfrak{H}}} t \quad \psi : t \mathrel{-\!\!\circ\!\!\twoheadrightarrow_{\mathfrak{H}}} u}{\varphi \cdot \psi : s \mathrel{-\!\!\circ\!\!\twoheadrightarrow_{\mathfrak{H}}} u}\ \mathsf{trans}$$

Table 2.2: Higher-order rewrite logic.

2.4.2 Higher-order rewrite logic

Let $\mathfrak{H} = \langle \Sigma, R \rangle$ be an HRS. In this subsection we give an alternative definition of the rewrite relation of $\mathfrak{H}$ by means of a higher-order rewrite logic, that is, a higher-order equational logic (see for example [56, 38]) without the symmetry rule. Rewrite logics were introduced for the first-order case by Meseguer [34]. We give a rewrite logic with an extra component, namely an explicit witness, called a proof term, which is similar to the proof terms we defined for the λ-calculus. A proof term is considered as an ordinary higher-order term over an extended signature: besides all the normal function symbols from Σ, it has a binary, polymorphic function symbol:

$$\text{“}\cdot\text{”} : \alpha \to \alpha \to \alpha$$

(called the composition symbol, which is written infix) and for each rule $\lambda\overline{x}.l \to \lambda\overline{x}.r$, with label ρ, a function symbol (a so-called *rule symbol*) ρ of the same type as l and r. Since proof terms are considered as terms, they are viewed modulo $\alpha\beta\eta$-equality.

The rules of our higher-order rewriting logic are listed in Table 2.2. They have conclusion of the form $\varphi : s \mathrel{-\!\!\circ\!\!\twoheadrightarrow_{\mathfrak{H}}} t$. Proof terms in which no rule symbol occurs, are called empty and may be denoted by 1. Note that the var, fun and rule can function as axioms when $n = 0$.

We now can prove that rewrite logic induces the same rewrite relation as before:

Lemma 2.4.13. *Let s, t be terms. There is a proof term φ such that $\varphi : s \multimap\!\!\!\!\circ\!\!\twoheadrightarrow t$ if and only if $s \twoheadrightarrow t$.*

Proof. ($\Rightarrow$): By induction on the inference of $\varphi : s \multimap\!\!\!\!\circ\!\!\twoheadrightarrow t$. We distinguish the following cases:

- Suppose var or fun is the last rule of the inference. We show only fun; var is similar. We know that $\varphi = f(\varphi_1, \ldots, \varphi_n)$, where $\varphi_i : s_i \multimap\!\!\!\!\circ\!\!\twoheadrightarrow t_i$. By the induction hypothesis, $s_i \to^* t_i$. But then:

  $$f(s_1, s_2 \ldots, s_n) \to^* f(t_1, s_2 \ldots, s_n) \to^* \cdots \to^* f(t_1, t_2 \ldots, t_n)$$

  as required.
- Suppose abs is the last rule of the inference. This is an easy variant of the above case, because there is only one direct subterm, and is proved analogously.
- Suppose rule is the last rule of the inference. We know now that $\varphi = \rho(\varphi_1, \ldots, \varphi_n)$, where $\varphi_i : s_i \multimap\!\!\!\!\circ\!\!\twoheadrightarrow t_i$. By the induction hypothesis $s_i \to^* t_i$. But then:

  $$l(s_1, \ldots, s_n) \to^* l(t_1, \ldots, t_n) \to r(t_1, \ldots, t_n)$$

  as required.
- Suppose trans is the last rule of inference. We know now that $\varphi = \varphi_1 \cdot \varphi_n$, where $\varphi_1 : s \multimap\!\!\!\!\circ\!\!\twoheadrightarrow u$ and $\varphi_2 : u \multimap\!\!\!\!\circ\!\!\twoheadrightarrow t$, for some term u. By the induction hypothesis, $s \to^* u$ and $u \to^* t$. But then $s \to^* t$, as required.

($\Leftarrow$): By induction on the derivation of $s \to^* t$. If $s \to t$, then by definition:

$$s =_\beta C[l(s_1, \ldots, s_n)] \quad \text{and} \quad t =_\beta C[r(s_1, \ldots, s_n)]$$

The claim follows by a nested induction on the size of C and the observation that, for any term u without rule or composition symbols, it holds that $u : u \multimap\!\!\!\!\circ\!\!\twoheadrightarrow u$. If $s = t$, then $s : s \multimap\!\!\!\!\circ\!\!\twoheadrightarrow t$ by the same observation.

If, on the other hand, there is a u such that $s \to^* u$ and $u \to^* t$, then it holds by the induction hypothesis that there are proof terms φ_1, φ_2 such that $\varphi_1 : s \multimap\!\!\!\!\circ\!\!\twoheadrightarrow u$ and $\varphi_2 : u \multimap\!\!\!\!\circ\!\!\twoheadrightarrow t$. But then:

$$(\varphi_1 \cdot \varphi_2) : s \multimap\!\!\!\!\circ\!\!\twoheadrightarrow t$$

as required. □

Note that proof terms, as defined in this section, may have *nested* composition symbols, that is compositions which occur within the argument of a function symbol, abstraction or even a rule symbol. For example, in the HRS with the two rules:

$$\begin{aligned} \rho :&\quad \mathsf{f}(x) \to \mathsf{g}(x) \\ \theta :&\quad \mathsf{g}(x) \to \mathsf{h}(x) \end{aligned}$$

we can infer:

$$\lambda x.(\rho(x) \cdot \theta(x)) : \lambda x.\mathsf{f}(x) \multimap\!\!\!\!\rightarrow \lambda x.\mathsf{h}(x)$$

Nested compositions are allowed in [42, 43] for the first-order case. There, they are convenient because of the fact that, for some context C containing no rule symbols or compositions, $C[\varphi \cdot \psi]$ and $C[\varphi] \cdot C[\psi]$ are identified, which makes it easier to define relations between and operations on proof terms. However, the above identification is problematic in the higher-order case, because it conflicts with the intuitively valid equation $1 \cdot 1 = 1$: Let $\rho : \mathsf{a} \to \mathsf{b}$ be the only rule of an HRS. Then:

$$\rho =_\beta (\lambda x.x)\rho = (\lambda x.(x \cdot x))\rho =_\beta \rho \cdot \rho$$

The proof term ρ is well-formed, but the proof term $\rho \cdot \rho$ isn't because $\mathrm{tgt}(\rho) \neq \mathrm{src}(\rho)$. So we cannot consider proof terms modulo $\beta\eta$-equivalence.

Because we introduced proof terms in particular for easily defining meta-operations on reductions, we have two options: do not allow nested compositions or do not consider proof terms modulo $\beta\eta$-equivalence. I have opted for the first solution. It is left to further research to devise an elegant form of proof term which includes nested compositions but does not have the problem observed above.

2.4.3 Proof terms for higher-order multisteps

The solution is to define proof terms only for multisteps (steps that may contract any number of independent redexes at a time)[5] and not for reductions of such steps (reductions are created from multisteps in the way of Def. 2.2.2). In the rest of this dissertation, we will use the following definition for proof term:

Definition 2.4.14 (Proof term)**.** Let $\mathfrak{H} = \langle \Sigma, R \rangle$. We assume that all rules of R carry a unique label. A $\mathfrak{H}$-proof term is a higher-order term over the signature Σ^+, where

$$\Sigma^+ := \Sigma \cup \{\rho : \alpha \mid \rho : l \to r \in R,\ l : \alpha\}.$$

In other words, proof terms are higher order terms with an extended signature, which includes for each rule ρ a rule symbol for that rule of the same type as its left-hand side (and thus of the same type as its right-hand side). The function $\mathcal{R}\mathit{ules}(\varphi)$ returns the set of rule symbols occurring in φ.

To define the multistep rewrite relation, we leave out the composition rule of Table 2.2 (that is, we leave out trans, the last rule). This yields the inference rules of Table 2.3. Note again that the inference rules var, fun and rule function as axioms if $n = 0$. The judgements that are proven by this

[5] Multisteps are also sometimes called *parallel steps*. We follow the terminology of [43], where parallel steps are distinguished from multisteps, in that latter may contract two nested redexes in one go, while the former may not.

Let $\mathfrak{H} = \langle \Sigma, R \rangle$ and $\rho : l \to r \in R$.

$$\frac{\varphi_1 : s_1 \multimap_{\mathfrak{H}} t_1 \cdots \varphi_n : s_n \multimap_{\mathfrak{H}} t_n}{x(\varphi_1, \ldots, \varphi_n) : x(s_1, \ldots, s_n) \multimap_{\mathfrak{H}} x(s_1, \ldots, s_n)}\ \mathsf{var}$$

$$\frac{\varphi_1 : s_1 \multimap_{\mathfrak{H}} t_1 \cdots \varphi_n : s_n \multimap_{\mathfrak{H}} t_n}{f(\varphi_1, \ldots, \varphi_n) : f(s_1, \ldots, s_n) \multimap_{\mathfrak{H}} f(s_1, \ldots, s_n)}\ \mathsf{fun}$$

$$\frac{\varphi : s \multimap_{\mathfrak{H}} t}{\lambda x.\varphi : \lambda x.s \multimap_{\mathfrak{H}} \lambda x.t}\ \mathsf{abs}$$

$$\frac{\varphi_1 : s_1 \multimap_{\mathfrak{H}} t_1 \cdots \varphi_n : s_n \multimap_{\mathfrak{H}} t_n}{\rho(\varphi_1, \ldots, \varphi_n) : l(s_1, \ldots, s_n) \multimap_{\mathfrak{H}} r(t_1, \ldots, t_n)}\ \mathsf{rule}$$

Table 2.3: Proof terms for higher-order steps.

inference system are again of the form $\varphi : s \multimap t$, stating that φ is a proof term witnessing that t can be reached from s by performing one multistep. The multistep relation is defined, analogously to the case of the λ-calculus, as follows:

$$s \multimap t \text{ if there is a } \varphi \text{ such that } \varphi : s \multimap t$$

In the following, we will conflate (multi)steps and proof terms, that is, we will occasionally say "the step $\rho(\mathsf{a})$" if we really mean "the step witnessed by the proof term $\rho(\mathsf{a})$".

Example 2.4.15. Consider the HRS consisting of the single rule:

$$\mu : \mathsf{mu}(\lambda x.z(x)) \to z(\mathsf{mu}(\lambda x.z(x)))$$

The following inference shows that the proof term $\mu(\lambda x.\mathsf{f}(x))$ witnesses a step from $\mathsf{mu}(\lambda x.\mathsf{f}(x))$ to $\mathsf{f}(\mathsf{mu}(\lambda x.\mathsf{f}(x)))$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x : x \multimap x}\ \mathsf{var}}{\mathsf{f}(x) : \mathsf{f}(x) \multimap \mathsf{f}(x)}\ \mathsf{fun}}{\lambda x.\mathsf{f}(x) : \lambda x.\mathsf{f}(x) \multimap \lambda x.\mathsf{f}(x)}\ \mathsf{abs}}{\mu(\lambda x.\mathsf{f}(x)) : \mathsf{mu}(\lambda x.\mathsf{f}(x)) \multimap \mathsf{f}(\mathsf{mu}(\lambda x.\mathsf{f}(x)))}\ \mathsf{rule}$$

In the last rule, note that:

$$(\lambda z.\mathsf{mu}(\lambda x.z(x)))(\lambda x.\mathsf{f}(x)) =_\beta \mathsf{mu}(\lambda x.\mathsf{f}(x))$$
$$(\lambda z.z(\mathsf{mu}(\lambda x.z(x))))(\lambda x.\mathsf{f}(x)) =_\beta \mathsf{f}(\mathsf{mu}(\lambda x.\mathsf{f}(x)))$$

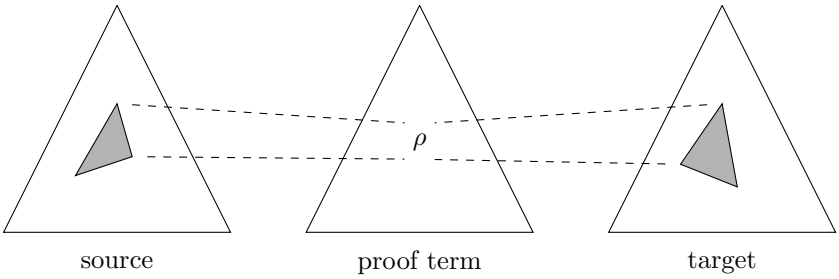


Figure 2.2: Intuition of rule symbols. *The source of a proof term is obtained by 'replacing' all rule symbols by the left-hand side of the corresponding rule, and the target is obtained by 'replacing' all rule symbols by the right-hand side.*

Example 2.4.16. Consider the HRS from Ex. 2.4.15 with the additional rule:

$$\rho : \mathsf{f}(x) \to \mathsf{g}(x)$$

The following inference shows that the proof term $\mu(\lambda x.\rho(x))$ witnesses a step from $\mathsf{mu}(\lambda x.\mathsf{f}(x))$ to $\mathsf{g}(\mathsf{mu}(\lambda x.\mathsf{g}(x)))$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x : x \multimap x}\,\mathsf{var}}{\rho(x) : \mathsf{f}(x) \multimap \mathsf{g}(x)}\,\mathsf{rule}}{\lambda x.\rho(x) : \lambda x.\mathsf{f}(x) \multimap \lambda x.\mathsf{g}(x)}\,\mathsf{abs}}{\mu(\lambda x.\rho(x)) : \mathsf{mu}(\lambda x.\mathsf{f}(x)) \multimap \mathsf{g}(\mathsf{mu}(\lambda x.\mathsf{g}(x)))}\,\mathsf{rule}$$

Intuitively, the rule symbols within the proof terms indicate where the 'action' takes place. Consider, for example, Fig. 2.2. In the middle of this figure we see the proof term, to the left its source and to the right its target. The rule symbol in the proof term indicates where the left-hand side of the rule in the source is replaced by the right-hand side of the rule.

We can show the following correspondence between the reduction relations $\to$ and $\multimap$:

Proposition 2.4.17. $\to \subseteq \multimap \subseteq \to^*$.

Proof. The first part, $\to \subseteq \multimap$, is shown by induction on the context C of the $\to$-step. If $C = a(C_1, \ldots, C_n)$ (where a is a function symbol or a variable) then the result follows easily from the induction hypothesis. If $C = \Box(s_1, \ldots, s_n)$, then $C[l] =_\beta l(s_1, \ldots, s_n)$ and $C[r] =_\beta r(s_1, \ldots, s_n)$, and so this in an instance of rule.

The second part, $\multimap \subseteq \to^*$, follows by induction on the inference of the $\multimap$-step φ. We distinguish the following cases:

- Suppose var or fun is the last rule of the inference. We show only fun. We know that $\varphi = f(\varphi_1, \ldots, \varphi_n)$, for $\varphi_i : s_i \multimap t_i$. By the induction hypothesis, $s_i \to^* t_i$. But then:

$$f(s_1, s_2 \ldots, s_n) \to^* f(t_1, s_2 \ldots, s_n) \to^* \cdots \to^* f(t_1, t_2 \ldots, t_n)$$

  as required.
- Suppose abs is the last rule of the inference. This is an easy variant of the above case, because there is only one direct subterm, and is proved analogously.
- Suppose rule is the last rule of the inference. We know now that $\varphi = \rho(\varphi_1, \ldots, \varphi_n)$, where $\varphi_i : s_i \multimap t_i$. By the induction hypothesis $s_i \to^* t_i$. But then:

$$l(s_1, \ldots, s_n) \to^* l(t_1, \ldots, t_n) \to r(t_1, \ldots, t_n)$$

  as required. □

Since, as indicated above, proof terms are terms over an extended alphabet, the definition of substitution on proof terms is inherited from terms. Again, we define, for a (proof term) substitution σ, the (term) substitution $\mathrm{src}(\sigma)$ and $\mathrm{tgt}(\sigma)$ as follows:

$$(\mathrm{src}(\sigma))(x) := \mathrm{src}(\sigma(x)) \quad \text{and} \quad (\mathrm{tgt}(\sigma))(x) := \mathrm{tgt}(\sigma(x)).$$

We can now prove the following useful fact, which is the HRS analogue of Lemma 2.3.5.

Lemma 2.4.18. *Let $\varphi\colon s \multimap t$ be a proof term, and σ a proof term substitution. Then $\varphi^\sigma : s^{\mathrm{src}(\sigma)} \multimap s^{\mathrm{tgt}(\sigma)}$.*

Proof. By induction on the structure of φ. □

We can now construct an ARS out of a HRS.

Definition 2.4.19. Let $\mathfrak{H} = \langle \Sigma, R \rangle$ be an HRS. The ARS of the HRS is defined as $\overline{\mathfrak{H}} = \langle \mathrm{Term}(\Sigma), \Phi, \mathrm{src}, \mathrm{tgt} \rangle$, where Φ is the set of proof terms of $\mathfrak{H}$ and src and tgt are defined such that:

$$\mathrm{src}(\varphi) = s \text{ and } \mathrm{tgt}(\varphi) = t \quad \text{if and only if} \quad \varphi : s \multimap t$$

In the following, we will conflate $\mathfrak{H}$ and $\overline{\mathfrak{H}}$. In particular, we will speak of $\mathfrak{H}$-reductions rather than $\overline{\mathfrak{H}}$-reductions. We introduce the following notations to project reduction without head steps to argument reductions:

Definition 2.4.20. Let, for $0 \leq i \leq n$, $\mathcal{R}_i = \varphi_{i0}, \varphi_{i1}, \ldots$ be (finite or infinite) reductions of the same length, f an n-ary function symbol, C an n-ary context

C and s a term of type $\alpha_1 \to \cdots \to \alpha_n \to \beta$. We define:

$$\begin{aligned} f^*(\mathcal{R}_1, \ldots, \mathcal{R}_n) &:= f(\varphi_{01}, \ldots, \varphi_{0n}), f(\varphi_{11}, \ldots, \varphi_{1n}), \ldots \\ \lambda x^*.\mathcal{R}_0 &:= \lambda x.\varphi_{00}, \lambda x.\varphi_{10}, \ldots \\ C^*(\mathcal{R}_1, \ldots, \mathcal{R}_n) &:= C(\varphi_{01}, \ldots, \varphi_{0n}), C(\varphi_{11}, \ldots, \varphi_{1n}), \ldots \\ s^*(\mathcal{R}_1, \ldots, \mathcal{R}_n) &:= s(\varphi_{01}, \ldots, \varphi_{0n}), s(\varphi_{11}, \ldots, \varphi_{1n}), \ldots \end{aligned}$$

Multisteps can contract any number of redexes in a single turn, because they may contain any number of rule symbols. We define the following subclasses of multisteps:

- An *empty* step is a step which contains no rule symbols. It is therefore equal to some term s and it holds that $s : s \multimap s$.
- A *proper* step is a step which contains precisely one rule symbol. Such a step is equivalent to the kind of steps defined in Def. 2.4.6, as can be shown by an easy inductive argument. A reduction which consists of only proper steps will be called a proper reduction.

There are two kinds of empty and infinite reductions. Reductions are sequences, so the notions of empty and infinite are inherited from sequences. However, it may be the case that an infinite reduction has only finitely many non-empty steps, or, on the other hand, a non-empty reduction consists of only empty steps. We want to explicitly distinguish these two forms of infinity and emptiness:

Definition 2.4.21. Let $\mathcal{R}$ be a reduction.

(i) $\mathcal{R}$ is *relatively finite* if it contains only finitely many non-empty steps.

(ii) $\mathcal{R}$ is *relatively empty* if it contains only empty steps.

The notions of "finite" and "empty" are directly inherited from sequences, that is a reduction is finite if it has finitely many empty or non-empty steps, and a reduction is empty if has no empty or non-empty steps at all. Observe that:

- Finiteness implies relative finiteness and emptiness implies relative emptiness.
- Since proper steps are by definition non-empty, for proper reductions, finiteness and relative finiteness coincide, as do emptiness and relative emptiness.
- A reduction which consists of infinitely many empty steps is both relatively empty and infinite.

The existence of infinite but relatively finite and non-empty but relatively empty reductions is a side-effect of our choice to view reductions as a sequence of (multi)steps and to allow empty multisteps. In practice they do not occur, but they need to be dealt with formally. To facilitate this, we define:

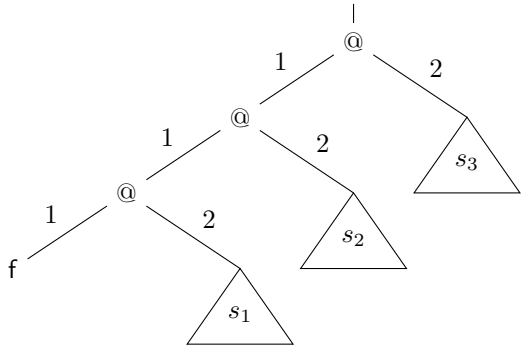


Figure 2.3: Tree representation of the term $\mathsf{f}(s_1, s_2, s_3)$.

Definition 2.4.22.

(i) Let $\mathcal{R}$ be a reduction. The reduction $[\![\mathcal{R}]\!]_{/1}$ is defined to be $\mathcal{R}$ from which all empty multisteps are removed.

(ii) Let $\mathcal{R}, \mathcal{S}$ be reductions. $\mathcal{R} =_{/1} \mathcal{S}$ if and only if $[\![\mathcal{R}]\!]_{/1} = [\![\mathcal{S}]\!]_{/1}$.

2.4.4 Positions and static tracing

The positions of terms are the same as the positions for λ-terms (we just look only at the $\beta\overline{\eta}$-normal form). We have to realize here, that our notational conventions complicate the correspondence between positions and parts of the term a little. For example, a term of the form $f(s_1, s_2, s_3)$ is really of the form $fs_1s_2s_3$: So, the position of the root of s_1 is 112, the position of the root of s_2 is 12, the position of the root of s_3 is 2 and the position of the function symbol f is 111 (see Fig. 2.3). In general, given a term $f(s_1, \ldots, s_n)$, the position of the root of the subterm s_i is $1^{n-i}2$ and the position of the function symbol is 1^n, where p^k is short for p repeated k times. This observation is important for understanding definitions concerning positions.

The trace relation of higher-order rewriting is a bit more complicated than the trace relation of β-reduction: because we consider terms modulo β-equivalence, we must trace the positions over the implicit β-reductions as well.

Definition 2.4.23. The higher-order trace relation over a proof term φ, denoted by $[\![\varphi\rangle\!\rangle$, is defined as follows:

$$
\begin{aligned}
[\![x(\varphi_1, \ldots, \varphi_n)\rangle\!\rangle &:= F([\![\varphi_1\rangle\!\rangle, \ldots, [\![\varphi_n\rangle\!\rangle) \\
[\![f(\varphi_1, \ldots, \varphi_n)\rangle\!\rangle &:= F([\![\varphi_1\rangle\!\rangle, \ldots, [\![\varphi_n\rangle\!\rangle) \\
[\![\lambda x.\varphi\rangle\!\rangle &:= \lambda.[\![\varphi\rangle\!\rangle \\
[\![\rho(\varphi_1, \ldots, \varphi_n)\rangle\!\rangle &:= ([\![\mathcal{M}\rangle\!\rangle_\beta)^{-1} ; \rho([\![\varphi_1\rangle\!\rangle, \ldots, [\![\varphi_n\rangle\!\rangle) ; [\![\mathcal{N}\rangle\!\rangle_\beta
\end{aligned}
$$

where, for relations on positions R_i:

- $\mathcal{M}$ and $\mathcal{N}$, in the 4th equation, are the standard[6] β-reductions from the preterms $l(\mathrm{src}(\varphi_1), \ldots, \mathrm{src}(\varphi_n))$ and $r(\mathrm{tgt}(\varphi_1), \ldots, \mathrm{tgt}(\varphi_n))$ to normal form, respectively;
- $\lambda.R$ is notation for the relation which relates ε to ε and $1p$ to $1q$ iff $p\ R\ q$;
- $F(R_1, \ldots, R_n)$ relates ε to ε and for each $1 \leq i \leq n$, $1^{n-i}2p$ to $1^{n-i}2q$ iff $p\ R_i\ q$; and
- $\rho(R_1, \ldots, R_n)$ relates, for each $1 \leq i \leq n$, $1^{n-i}2p$ to $1^{n-i}2q$ iff $p\ R_i\ q$.

Lemma 2.4.24. *Let $\varphi : s \multimap t$ be a multistep, and let $P = \mathcal{P}os(s)$. Then:*

$$[\![\varphi]\!] = \mathsf{id}{\restriction}P \quad \textit{if and only if} \quad \varphi = 1.$$

Proof. By induction on the size of φ, noting that non-empty steps do not trace the positions in the left-hand side of the rule to any position in the right-hand side of the rule. □

The trace relation traces positions that are not involved in the step (in [53], this kind of tracing relation is called a *static* tracing relation). Because of this, the positions of the target of a reduction which were involved in (touched by) the reduction can be distinguished from the other positions because they do not trace to any position in the source.

Definition 2.4.25. Let $\mathcal{R}$ be a finite reduction.

(i) A position $p \in \mathcal{P}os(\mathrm{tgt}(\mathcal{R}))$ is *touched* by $\mathcal{R}$ if there is no $q \in \mathrm{src}(\mathcal{R})$ such that $q\ [\![\mathcal{R}]\!]\ p$.

(ii) Suppose $\mathcal{R} = C^*[\mathcal{R}_1, \ldots, \mathcal{R}_n]$, where $\mathcal{R}_i : s_i \multimap\!\!\!\!\rightarrow t_i$. A position $p \in \mathcal{P}os(\mathrm{tgt}(\mathcal{R}))$ is touched by $\mathcal{R}_k$, if it touched by $C^*[t_1, \ldots, \mathcal{R}_k, \ldots, t_n]$.

In local HRSs, there is at most one position in the source of a step that traces to a given position in the target of a step.

Lemma 2.4.26. *Let $\mathfrak{H}$ be a local* HRS. *If $p\ [\![\varphi]\!]_{\mathfrak{H}}\ q$ and $p'\ [\![\varphi]\!]_{\mathfrak{H}}\ q$, then $p = p'$.*

Proof. By induction on φ. The cases that $\varphi = \lambda x.\varphi_0$ and $\varphi = f(\varphi_1, \ldots, \varphi_n)$ are easy. In the case that $\varphi = \rho(\varphi_1, \ldots, \varphi_n)$ we need Lemma 2.3.11(i) and Lemma 2.3.11(ii) together with the assumption that $\mathfrak{H}$ is linear to prove that, for each position q, there is at most one position p such that $p\ (\![\mathcal{M}]\!]_\beta)^{-1}\ q$ and at most one position p such that $p\ [\![\mathcal{N}]\!]_\beta\ q$. □

[6] We claim that all β-reductions from a term to normal form induce the same trace relation, so we could have said "arbitrary" here. We use "standard" because it is well-known from the literature that each β-reduction has a unique standard reduction (for example [25, Theorem 9.8.3]) and thus the definition is well-defined without the requirement to actually prove that all β-reductions to normal form induce the same trace relation.

In the following lemma an important auxiliary result is proved which will be used in following chapters. The lemma expresses that positions 'outside' the redex pattern of a step are not touched by that step. Formally:

Lemma 2.4.27. *Let $\varphi : s \to t$, with $\varphi = C[\rho(s_1, \ldots, s_n)]_p$, be a proper step. It holds that:*

(i) For any position q such that $p \leq q$, if $q \, [\![\varphi\rangle\!\rangle \, q'$, then $p \leq q$ if and only if $p \leq q'$;

(ii) For any other position q, $q \, [\![\varphi\rangle\!\rangle \, q'$ if and only if $q = q'$.

Proof. By a simple induction on the structure of the context C. □

2.4.5 Orthogonality of proper steps

In the familiar rewriting theory, orthogonality is a property of rewrite systems: a left-linear rewrite system is orthogonal, if none of its left-hand sides overlaps with another left-hand side. Here, we want to consider orthogonality as a property of *steps*: two steps are orthogonal if they are independent, that is, if the redex patterns they contract do not overlap. The two notions of orthogonality are connected in the following way: if a rewrite system is orthogonal (in the conventional sense), then all steps from the same source are orthogonal (in our sense). In this section we formally define the notion of orthogonality for higher-order, *proper* steps. The notion is easily generalized to multisteps.

Definition 2.4.28 (Redex pattern)**.** The set of redex positions, or *redex pattern*, of a proper step φ, denoted by $\mathcal{RP}os(\varphi)$, is defined as follows:

$$\begin{aligned}
\mathcal{RP}os(x(\varphi_1, \ldots, \varphi_n)) &:= \bigcup_{1 \leq i \leq n} 1^{n-i}2 \,;\, \mathcal{RP}os(\varphi_i) \\
\mathcal{RP}os(f(\varphi_1, \ldots, \varphi_n)) &:= \bigcup_{1 \leq i \leq n} 1^{n-i}2 \,;\, \mathcal{RP}os(\varphi_i) \\
\mathcal{RP}os(\lambda x.\varphi_0) &:= 1 \,;\, \mathcal{RP}os(\varphi_0) \\
\mathcal{RP}os(\rho(s_1, \ldots, s_n)) &:= \mathcal{P}os(l(\Box, \ldots, \Box))
\end{aligned}$$

where $\rho : l \to r$ is a rule.

As said, two proper steps are called orthogonal if their redex patterns do not overlap. Formally we can define:

Definition 2.4.29 (Orthogonal)**.**

(i) Let φ, ψ be proper steps. φ and ψ are *orthogonal* to each other if $\mathcal{RP}os(\varphi) \cap \mathcal{RP}os(\psi) = \varnothing$.

(ii) Let $\mathfrak{H}$ be a HRS. $\mathfrak{H}$ is orthogonal, if are coinitial proper steps are orthogonal.

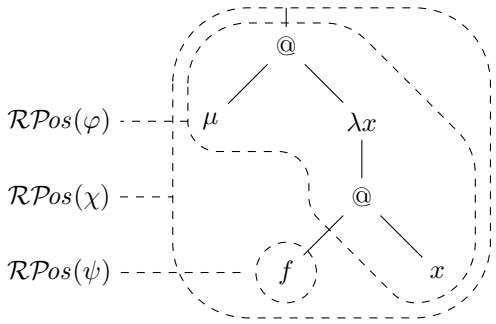


Figure 2.4: Redex patterns of $\varphi = \mu(\lambda x.\mathsf{f}(x))$, $\psi = \mathsf{mu}(\lambda x.\rho(x))$ and $\chi = \theta$.

Example 2.4.30. Consider the following (little bit contrived) HRS:

$$\begin{aligned}
\mu :&\quad \mathsf{mu}(\lambda x.z(x)) \to z(\mathsf{mu}(\lambda x.z(x))) \\
\rho :&\quad \mathsf{f}(x) \to \mathsf{g}(x) \\
\theta :&\quad \mathsf{mu}(\lambda x.\mathsf{f}(x)) \to \mathsf{a}
\end{aligned}$$

First of all, we note that the HRS is not orthogonal in the traditional sense of the word because of the overlap between rules μ and θ and rules ρ and θ. Consider the following steps:

$$\begin{aligned}
\varphi = &\quad \mu(\lambda x.\mathsf{f}(x)) : \mathsf{mu}(\lambda x.\mathsf{f}(x)) \to \mathsf{f}(\mathsf{mu}(\lambda x.\mathsf{f}(x))) \\
\psi = &\quad \mathsf{mu}(\lambda x.\rho(x)) : \mathsf{mu}(\lambda x.\mathsf{f}(x)) \to \mathsf{mu}(\lambda x.\mathsf{g}(x)) \\
\chi = &\quad \theta : \mathsf{mu}(\lambda x.\mathsf{f}(x)) \to \mathsf{a}
\end{aligned}$$

We calculate the positions of the redex patterns of all steps (see Fig. 2.4):

$$\begin{aligned}
\mathcal{RP}os(\varphi) &= \{\varepsilon, 1, 2, 21, 212\} \\
\mathcal{RP}os(\psi) &= \{211\} \\
\mathcal{RP}os(\chi) &= \{\varepsilon, 1, 2, 21, 211, 212\}
\end{aligned}$$

By definition, φ and ψ are orthogonal to each other, whereas φ and χ (and ψ and χ) are not. The HRS is not orthogonal because it allows non-orthogonal, coinitial, proper steps.

The definition of a set of redex positions of a proper step, also allows to define the *position* of a step. It is easy to see that, for a proper step φ, $\mathcal{RP}os(\varphi)$ has a unique $\sqsubseteq$-minimal position, which corresponds to the position of the hole of the context in the step $C[l] \to C[r]$. We define:

Definition 2.4.31. Let φ be a proper step. The step φ contracts a redex R at position p, if p is the $\sqsubseteq$-minimal position of $\mathcal{RP}os(\varphi)$.

2.5 Meta-equivalence and meta-rewriting

In definitions and proofs we will sometimes employ rewrite systems on the meta-level that operate on finite reductions rather than (higher-order) terms. In this section we will investigate some of the theory concerning such meta-rewrite systems. Since we represent reductions as sequences of (multi)steps, and strings are commonly defined as finite sequences, it is natural to use techniques from string rewriting to define our meta-equivalence and meta-rewriting, so that meta-rewriting inherits useful properties from string string rewriting, most notably the critical pair lemma, which establishes that a rewrite system is confluent if all its critical pairs can be joined.

Definition 2.5.1. Let $\mathfrak{A}$ be a rewrite system.

(i) A $\mathfrak{A}$-meta rewrite system is a set of tuples $\langle \mathcal{L}, \mathcal{R} \rangle$ such that $\mathcal{L}, \mathcal{R}$ are $\mathfrak{A}$-reductions.

(ii) Let Mrw be an $\mathfrak{A}$-meta rewrite system. The one-step rewrite relation $\Rightarrow_{\text{Mrw}}$ is the smallest relation such that:

- if $\langle \mathcal{R}, \mathcal{S} \rangle \in \text{Mrw}$ then $\mathcal{R} \Rightarrow_{\text{Meq}} \mathcal{S}$;
- if $\mathcal{R} \Rightarrow_{\text{Meq}} \mathcal{S}$, then $\mathcal{T}_1 ; \mathcal{R} ; \mathcal{T}_2 \Rightarrow_{\text{Mrw}} \mathcal{T}_1 ; \mathcal{S} ; \mathcal{T}_2$.

(iii) Let Meq be an $\mathfrak{A}$-meta rewrite system (in such a case also called $\mathfrak{A}$-meta equational system). The equivalence relation $\Leftrightarrow_{\text{Meq}}$ is defined as follows:

$$\Leftrightarrow_{\text{Meq}} := (\Rightarrow_{\text{Meq}} \cup \Rightarrow_{\text{Meq}}^{-1})$$

We will usually use meta-rewrite systems with infinitely many rules, where the rules are given by rule schemas.

Often we distinguish between steps of the reduction which are involved in a meta-rewrite step and steps which are not. This distinction can be formally defined as follows. The steps which are involved in the meta-rewrite step are the ones that where used to match against the left-hand side of the meta-rewrite rule.

2.6 Discussion

This chapter served two purposes: to introduce the notions of rewriting that are used in the proceeding chapters, in particular higher-order rewriting; and to define explicit witnesses to steps and reductions, called proof terms. Proof terms have two advantages:

- They enable us to easily use recursion to define notions of steps. For example, in Def. 2.4.23, the tracing relation of a step is defined by recursion on that step.
- They are convenient when defining operations which manipulate steps and reductions. Chapters 3 (Permutation equivalence), 5 (Standardization equivalence) and 6 (Residuals) make extensive use of proof terms.

Three

Permutation equivalence

3.1 Introduction

In this chapter we will investigate permutation equivalence for reductions in fully extended HRSs. Permutation equivalence states that two reductions are the same if the one can be transformed into the other by repeatedly permuting adjacent, independent steps. In later chapters two more methods of formalizing the notion of equivalence of reductions will be defined, to wit standardization equivalence and projection equivalence, and proven equivalent to permutation equivalence.

In the literature, permutation equivalence is often defined only for finite reductions. In the first part of the chapter (Sect. 3.2), we also consider only finite reductions. In the second part (Sect. 3.3), however, we also pay some attention to permutation equivalence for infinite reductions and the various problems which arise in this case.

3.2 Permutation equivalence for finite reductions

Unless otherwise indicated, in this section we restrict our attention to finite reductions. Also, we assume that HRSs are fully extended. We define permutation equivalence of finite reduction by means of a meta-equivalence system (see Sect. 2.5), each equivalence of which performs one permutation.

Definition 3.2.1. Let $\mathfrak{H} = \langle \Sigma, R \rangle$ be a HRS. The permutation equivalence meta-equivalence system $\Leftrightarrow_{\mathrm{P}}$ is defined as follows:

$$C[\rho(\overline{\varphi})] \Leftrightarrow C[\rho(\overline{s})], C[r(\overline{\varphi})] \qquad \text{(flat-l)}$$

$$C[\rho(\overline{\varphi})] \Leftrightarrow C[l(\overline{\varphi})], C[\rho(\overline{t})] \qquad \text{(flat-r)}$$

$$C[\varphi_1, \ldots, \varphi_n] \Leftrightarrow C[s_1, \ldots, \varphi_k, \ldots, s_n], C[\varphi_1, \ldots, t_k, \ldots, \varphi_n] \qquad (\mathrm{ser}_k\text{-l})$$

$$C[\varphi_1, \ldots, \varphi_n] \Leftrightarrow C[\varphi_1, \ldots, t_k, \ldots, \varphi_n], C[t_1, \ldots, \varphi_k, \ldots, t_n] \qquad (\mathrm{ser}_k\text{-r})$$

$$1 \Leftrightarrow \varepsilon \qquad \text{(unit)}$$

where:

- C is a base context containing no rule symbols;
- $\varphi\colon s \multimap t$, $\psi : u \multimap v$ and $\varphi_i\colon s_i \multimap t_i$, $\psi_i\colon u_i \multimap v_i$;
- $\rho\colon l \to r \in R$.

We define permutation equivalence as:

$$\mathcal{R} \approx \mathcal{S} \quad \text{if} \quad \mathcal{R} \Leftrightarrow_{\mathrm{P}}^{*} \mathcal{S}$$

Lemma 3.2.2. *The following are derived rules of* $\approx$*:*

$$C[l(\overline{\varphi})]\,;C[\rho(\overline{t})] \approx C[\rho(\overline{s})]\,;C[r(\overline{\varphi})] \qquad \text{(std)}$$
$$C[s,\psi]\,;C[\varphi,v] \approx C[\varphi,u]\,;C[t,\psi] \qquad \text{(par)}$$

where C is a base context containing no rule symbols, $\varphi_{(i)}\colon s_{(i)} \geq t_{(i)}$, $\psi_{(i)}\colon u_{(i)} \geq v_{(i)}$, and $\rho\colon l \to r \in R$.

Proof. By the following conversions:

$$C[l(\overline{\varphi})]\,;C[\rho(\overline{t})] \Longleftrightarrow_{(\text{flat-r})} C[\rho(\overline{\varphi})] \Longleftrightarrow_{(\text{flat-l})} C[\rho(\overline{s})]\,;C[r(\overline{\varphi})] \qquad \text{(std)}$$
$$C[s,\psi]\,;C[\varphi,v] \Longleftrightarrow_{(\text{ser}_1\text{-r})} C[\varphi,\psi] \Longleftrightarrow_{(\text{ser}_2\text{-l})} C[\varphi,u]\,;C[t,\psi] \qquad \text{(par)}$$

□

The derived rules (std) and (par) will in the following be used without reference to the lemma.

Example 3.2.3. Let the following HRS be given:

$$\begin{aligned} \mu :&\quad \mathsf{mu}(\lambda x.z(x)) \to z(\mathsf{mu}(\lambda x.z(x))) \\ \rho :&\quad \mathsf{f}(x) \to \mathsf{g}(x) \end{aligned}$$

and the following reductions:

$$\mathcal{R} : \mathsf{mu}(\lambda x.\mathsf{f}(x)) \to \mathsf{mu}(\lambda x.\mathsf{g}(x)) \to \mathsf{g}(\mathsf{mu}(\lambda x.\mathsf{g}(x)))$$
$$\mathcal{S} : \mathsf{mu}(\lambda x.\mathsf{f}(x)) \to \mathsf{f}(\mathsf{mu}(\lambda x.\mathsf{f}(x))) \to \mathsf{f}(\mathsf{mu}(\lambda x.\mathsf{g}(x))) \to \mathsf{g}(\mathsf{mu}(\lambda x.\mathsf{g}(x)))$$

The following conversion shows that both reductions are permutation equivalent:

$$\begin{aligned} &\mathsf{mu}(\lambda x.\rho(x)), \mu(\lambda x.\mathsf{g}(x)) \\ \Leftrightarrow_{(\text{par})} \quad &\mu(\lambda x.\mathsf{f}(x)), \rho(\mathsf{mu}(\lambda x.\rho(x))) \\ \Leftrightarrow_{(\text{flat-r})} \ &\mu(\lambda x.\mathsf{f}(x)), \mathsf{f}(\mathsf{mu}(\lambda x.\rho(x))), \rho(\mathsf{mu}(\lambda x.\mathsf{g}(x))) \end{aligned}$$

Lemma 3.2.4. *Let $\mathcal{R}, \mathcal{S}$ be finite reductions. If $\mathcal{R} \approx \mathcal{S}$, then* $\mathrm{src}(\mathcal{R}) = \mathrm{src}(\mathcal{S})$ *and* $\mathrm{tgt}(\mathcal{R}) = \mathrm{tgt}(\mathcal{S})$.

Proof. By induction on the derivation of $\mathcal{R} \Leftrightarrow_{\mathrm{P}}^{*} \mathcal{S}$. The induction step (the last rule of the derivation is reflexivity, symmetry or transitivity) is easy. In the base case we have that $\mathcal{R} \Leftrightarrow \mathcal{S}$ is an instance of an equation of Def. 3.2.1. An easy case analysis shows that, indeed, $\mathrm{src}(\mathcal{R}) = \mathrm{src}(\mathcal{S})$ and $\mathrm{tgt}(\mathcal{R}) = \mathrm{tgt}(\mathcal{S})$. □

The previous lemma makes it possible to compose reductions which are permutation equivalent to composable reductions. That is: if $\mathcal{R} \approx \mathcal{R}'$ and $\mathcal{S} \approx \mathcal{S}'$, and $\mathcal{R}\,;\mathcal{S}$ is well-defined, then $\mathcal{R}'\,;\mathcal{S}'$ is well-defined (and by definition, $\mathcal{R}\,;\mathcal{S} \approx \mathcal{R}'\,;\mathcal{S}'$). In the future, we will implicitly use this fact.

The following useful property of permutation equivalence that we will prove, is the fact that permutation equivalent reductions induce the same tracing relation.

Proposition 3.2.5. *Let $\mathfrak{H}$ be a local* HRS*, and $\mathcal{R}, \mathcal{S}$ $\mathfrak{H}$-reductions. If $\mathcal{R} \approx \mathcal{S}$ then $[\![\mathcal{R}\rangle\!\rangle = [\![\mathcal{S}\rangle\!\rangle$.*

Proof. The lemma follows from the fact that, by construction, $[\![\mathcal{L}\rangle\!\rangle = [\![\mathcal{R}\rangle\!\rangle$ for each equation $\mathcal{L} \approx \mathcal{R}$ from Def. 3.2.1. □

Remark 3.2.6. Permutation equivalence has some unexpected properties if we consider HRSs with erasing rules. In particular, it is not the case that if $\mathcal{R} \not\approx \mathcal{S}$, then $\mathcal{R}\,;\mathcal{T} \not\approx \mathcal{S}\,;\mathcal{T}$. Consider for example the HRS:

$$\begin{aligned} \rho &: \quad \mathsf{a} \to \mathsf{a} \\ \theta &: \quad \mathsf{f}(x) \to \mathsf{c} \end{aligned}$$

Then:

$$\mathsf{f}(\rho)\,;\theta(\mathsf{a}) \Longleftrightarrow_{(\mathrm{unit})} \mathsf{f}(\rho)\,;\theta(\mathsf{a})\,;\mathsf{c} \Longleftrightarrow_{(\mathrm{std})} \mathsf{f}(\rho)\,;\mathsf{f}(\rho)\,;\theta(\mathsf{a})$$

but $\mathsf{f}(\rho) \not\approx \mathsf{f}(\rho)\,;\mathsf{f}(\rho)$. Note that the inverse *does* hold: if $\mathcal{R} \approx \mathcal{S}$, then by definition $\mathcal{R}\,;\mathcal{T} \approx \mathcal{S}\,;\mathcal{T}$.

We now proceed by proving a useful auxiliary result, to wit that each finite reduction (possibly containing multisteps or empty steps) has a permutation equivalent proper reduction. In fact, the theorem is proved by explicitly constructing a proper reduction from an arbitrary reduction. The result is useful, because it can be used to disregard non-proper reductions in some proofs, making the proofs less cumbersome. It will be considerably strengthened in Chapter 5 (Standardization), where it will be shown that each reduction not only has a permutation equivalent proper reduction, but even a permutation equivalent *standard* reduction, where a standard reduction is a proper reduction in which the redexes are contracted visually from left to right.

Lemma 3.2.7. *Let $\mathcal{R}_1, \ldots, \mathcal{R}_n$ and $\mathcal{S}_1, \ldots, \mathcal{S}_n$ be reductions, such that $|\mathcal{R}_i| = |\mathcal{R}_j|$, $|\mathcal{S}_i| = |\mathcal{S}_j|$, and $\mathcal{R}_i \approx \mathcal{S}_i$, for $i, j \in \{1, \ldots n\}$. Then:*

$$C^*[\mathcal{R}_1, \ldots, \mathcal{R}_n] \approx C^*[\mathcal{S}_1, \ldots, \mathcal{S}_n].$$

Proof. First we show that if $\mathcal{R} \approx \mathcal{S}$, then $D^*[\mathcal{R}] \approx D^*[\mathcal{S}]$. This is proved by induction on the length of the derivation of $\mathcal{R} \approx \mathcal{S}$, and a nested induction on the size of D. The lemma follows from this and transitivity. □

Theorem 3.2.8.

(i) For each multistep φ, there exists a finite proper reduction $\mathcal{R}$ such that $\varphi \approx \mathcal{R}$. $\mathcal{R}$ is empty if and only if φ is.

(ii) For each finite reduction $\mathcal{R}$, there exists a finite proper reduction $\mathcal{R}'$ such that $\mathcal{R} \approx \mathcal{R}'$. $\mathcal{R}'$ is empty if and only if $\mathcal{R}$ is.

Proof. (i) By induction on the length of φ, distinguishing the following cases:

- If $\varphi = c$, for some nullary function symbol c, then we simply take $\mathcal{R}$ to be the empty reduction starting at c.
- Suppose $\varphi = C[\varphi_1, \ldots, \varphi_n]$ for some non-empty context C containing no rule symbols, and $\varphi_i : s_i \multimap t_i$. By the (ser-l) and (ser-r) equations

  $$\varphi \approx C[\varphi_1, s_2, \ldots, s_n], \ldots, C[t_1, \ldots, t_{n-1}, \varphi_n].$$

  By the induction hypothesis, there are finite $\mathcal{R}_i$ such that $\varphi_i \approx \mathcal{R}_i$ and $\mathcal{R}_i$ is empty if and only φ_i is. By repeated application of Lemma 3.2.7 we obtain

  $$\varphi \approx C^*[\mathcal{R}_1, s_2, \ldots, s_n] \,;\cdots;\, C^*[t_1, \ldots, t_{n-1}, \mathcal{R}_n].$$

  We take $\mathcal{R}$ to be the right part of this equation (with all empty steps removed). If φ is empty, all $\mathcal{R}_i$ are empty by induction hypothesis, and thus $\mathcal{R}$ is empty. If, on the other hand, φ is non-empty, then some $\mathcal{R}_i$ must be non-empty, and thus $\mathcal{R}$ is non-empty.
- Suppose $\varphi = \rho(\varphi_1, \ldots, \varphi_n)$ for $\varphi_i : s_i \multimap t_i$. By (flat-r) it holds that

  $$\varphi \approx l(\varphi_1, \ldots, \varphi_n), \rho(t_1, \ldots, t_n).$$

  We act as in the previous case to obtain a finite proper reduction $\mathcal{S}$ such that $\mathcal{S} \approx l(\varphi_1, \ldots, \varphi_n)$, and then we take $\mathcal{R} := \mathcal{S} \,;\, \rho(t_1, \ldots, t_m)$ (with all empty steps removed). Both φ and $\mathcal{R}$ are non-empty.

(ii) By induction on the length of $\mathcal{R}$, using (i). □

The proof of the previous theorem gives rise to the following useful definitions:

- for a multistep φ, we call the proper reduction created by the construction in the proof of Theorem 3.2.8(i) the *canonical development* of φ;
- the canonical development of a reduction $\mathcal{R} = \varphi_0, \varphi_1, \ldots$ is the proper reduction $\mathcal{S}_0 \,;\, \mathcal{S}_1 \,;\cdots$, where $\mathcal{S}_i$ is the canonical development of the step φ_i.

Example 3.2.9. The canonical development of the multistep $\mu(\lambda x.\rho(x))$ of the HRS from Ex. 3.2.3 is:

$$\mathsf{mu}(\lambda x.\rho(x)), \mu(\lambda x.\mathsf{f}(x))$$

Proposition 3.2.10. *The canonical development of a reduction $\mathcal{R}$ is infinite if and only if $\mathcal{R}$ is relatively infinite*

Proof. Directly from the claim of Theorem 3.2.8 that the canonical development of a multistep φ is empty if and only if φ is empty. □

3.3 Permutation equivalence for infinite reductions

Defining permutation equivalence for infinite reductions is not as easy as defining permutation equivalence for finite reductions. The problem is not defining the notion of one permutation, because it is a completely local operation. The problem is that it is not always clear what the result of performing an infinite amount of permutations should be. In fact, it is not even intuitively clear what the desired properties of permutation equivalence for infinite reductions are. Consider, for example, the following TRS:

$$\begin{aligned}\mathsf{f}(x) &\to \mathsf{g}(\mathsf{f}(x))\\ \mathsf{a} &\to \mathsf{b}\end{aligned}$$

and the following infinite reductions:

$$\begin{aligned}\mathcal{R} &: \mathsf{f}(\mathsf{a}) \to \mathsf{f}(\mathsf{b}) \to \mathsf{g}(\mathsf{f}(\mathsf{b})) \to \mathsf{g}(\mathsf{g}(\mathsf{f}(\mathsf{b}))) \to \cdots\\ \mathcal{S} &: \mathsf{f}(\mathsf{a}) \to \mathsf{g}(\mathsf{f}(\mathsf{a})) \to \mathsf{g}(\mathsf{g}(\mathsf{f}(\mathsf{a}))) \to \cdots\end{aligned}$$

Should $\mathcal{R}$ and $\mathcal{S}$ be equivalent? On the one hand, it seems obvious that they should not, because the descendants of the redex contracted by the first step of $\mathcal{R}$ are not contracted by any step of $\mathcal{S}$. On the other hand, the 'result' of both reductions is the infinite term $\mathsf{g}(\mathsf{g}(\mathsf{g}(\cdots)))$, and the first step of $\mathcal{R}$ does not participate in producing it. So, the situation is similar to the erased steps situation in the previous section.

In this section we consider an infinite reduction as the limit of its finite prefixes, and say that two infinite reductions are permutation equivalent if any finite prefix of the one can be extended so that it's permutation equivalent to a finite prefix of the other, and vice versa. This is merely a design choice, which is sufficient for our purposes.

Definition 3.3.1. Let $\mathcal{R}, \mathcal{S}$ be reductions.

(i) $\mathcal{R} \leq_\infty \mathcal{S}$ if for each finite $\mathcal{R}_0$ such that $R_0 \sqsubseteq R$, there is a finite $\mathcal{S}_0$ such that $\mathcal{S}_0 \approx \mathcal{R}_0 ; \mathcal{T}$ for some $\mathcal{T}$ and $\mathcal{S}_0 \sqsubseteq \mathcal{S}$.

(ii) $\mathcal{R} \approx_\infty \mathcal{S}$ if $\mathcal{R} \leq_\infty \mathcal{S}$ and $\mathcal{S} \leq_\infty \mathcal{R}$.

Note that in the example above, $\mathcal{R}$ and $\mathcal{S}$ are not permutation equivalent according to this definition. Let $\mathcal{R}_0$ be the reduction consisting of only the first step of $\mathcal{R}$. Because no descendant of the redex a is contracted in $\mathcal{S}$, there is no prefix $\mathcal{S}_0$ of $\mathcal{S}$ which is permutation equivalent to $\mathcal{R}\,;\mathcal{T}$, for some reduction $\mathcal{T}$.

The new notion of permutation equivalence is conservative over the old, finitistic one. The proof of this depends on the following property. We prove the property by using standardization from the next chapter (Of course, we have made sure that standardization itself does not need this lemma.)

Lemma 3.3.2. *Let $\mathcal{R}$ be a finite reduction. If $\mathcal{R} \approx \mathcal{R}\,;\varphi$, then $\varphi = 1$.*

Proof. (Sketch) Without loss of generality, we assume that $\mathcal{R}$ is a standard reduction. Now we calculate the standard reduction equivalent to $\mathcal{R};\varphi$. This is done by iteratively permuting φ to its correct place. Thus, $\mathit{Std}(\mathcal{R};\varphi) = \mathcal{R}';\varphi';\mathcal{S}$, such that $\mathcal{R} = \mathcal{R}';\mathcal{T}$. The only way that $\mathcal{R} = \mathit{Std}(\mathcal{R}';\varphi';\mathcal{S})$ is when $\varphi = 1$. □

Remark 3.3.3. The symmetrical claim to Lemma 3.3.2, viz. that $\mathcal{R} \approx \varphi\,;\mathcal{R}$ implies $\varphi = 1$, does *not* hold, because the HRS may contain erasing rules. Consider again the HRS from Remark 3.2.6. In this HRS it holds that

$$\mathsf{f}(\rho)\,;\theta(\mathsf{a}) \approx \theta(\mathsf{a})\,;\mathsf{c} \approx \theta(\mathsf{a}).$$

This asymmetry is due to the fact that the variables of right-hand sides may be a strict subset of the variables of the left-hand sides of rules; in other words, it is (again) caused by erasing rules.

Proposition 3.3.4. *Let $\mathcal{R},\mathcal{S}$ be finite reductions. Then $\mathcal{R} \approx \mathcal{S}$ if and only if $\mathcal{R} \approx_\infty \mathcal{S}$.*

Proof. ($\Rightarrow$): Suppose $\mathcal{R} \approx \mathcal{S}$. First we show $\mathcal{R} \leq_\infty \mathcal{S}$. Suppose $\mathcal{R}_0 \sqsubseteq \mathcal{R}$. The conditions of the definition are satisfied by taking $\mathcal{T}$ to be the reduction such that $\mathcal{R}_0\,;\mathcal{T} = \mathcal{R}$, and $\mathcal{S}_0 := \mathcal{S}$. The requirement that $\mathcal{S} \leq_\infty \mathcal{R}$ is proved symmetrically.

($\Leftarrow$): Suppose $\mathcal{R} \approx_\infty \mathcal{S}$. Since $\mathcal{R} \sqsubseteq \mathcal{R}$, there is some $\mathcal{S}_0 \sqsubseteq \mathcal{S}$ such that $\mathcal{S}_0 \approx \mathcal{R}$. Symmetrically, there is some $\mathcal{R}_0 \sqsubseteq \mathcal{R}$ such that $\mathcal{R}_0 \approx \mathcal{S}$. So, we have $\mathcal{T}_1,\mathcal{R}_2$ such that: $\mathcal{R}_0\,;\mathcal{T}_1 = \mathcal{R}$ and $\mathcal{S}_0\,;\mathcal{T}_2 = \mathcal{S}$. Substitution yields $\mathcal{R}\,;\mathcal{T}_2\,;\mathcal{T}_1 \approx \mathcal{R}$ and $\mathcal{S}\,;\mathcal{T}_1\,;\mathcal{T}_2 \approx \mathcal{S}$. By (repeated application of) Lemma 3.3.2, $\mathcal{T}_1$ and $\mathcal{T}_2$ consist of empty steps, and thus: $\mathcal{R} \approx \mathcal{R}_0 \approx \mathcal{S}$. □

Theorem 3.3.5. *For every relatively infinite reduction $\mathcal{R}$, there exists a permutation equivalent, infinite, proper reduction $\mathcal{S}$ with the same source.*

Proof. Let $\mathcal{R} = \varphi_1,\varphi_2,\ldots$. By Theorem 3.2.8, each φ_i is mapped to a finite proper reduction $\mathcal{S}_i$. Since $\mathcal{R}$ is relatively terminating, infinitely many of these reductions are non-empty, and so $\mathcal{S} = \mathcal{S}_1\,;\mathcal{S}_2\,;\ldots$ is an infinite proper reduction. Trivially, $\mathcal{R} \approx_\infty \mathcal{S}$. The fact that $\mathrm{src}(\mathcal{S}) = \mathrm{src}(\mathcal{R})$ follows from the fact that $\mathcal{S}_1 \approx \varphi_1$ and Lemma 3.2.4. □

3.4 Discussion

In this chapter we have defined a notion of equivalence of reductions, permutation equivalence. This notion expresses that two reductions are equivalent to each other, if the one can be transformed into the other by repeatedly permuting independent steps.

This notion of equivalence does only consider the 'needed' parts of the reductions, the parts that, as it were, help in producing the end result. This property implies that it is *not* the case that each permutation equivalence class of reductions has an upper bound on the length (the equivalence in the example on page 43 can be repeated indefinitely). The termination proofs of the standardization procedures of Chapter 5, however, do depend on such an upper bound. For this reason, the proofs in that chapter are a bit more involved, because we have to restrict the equivalence class a bit.

Alternatively, we could devise a notion of permutation equivalence which does not forget unneeded parts of the reduction. For this, a possible solution would be to add 'memory symbols' to the signature, whose function it is to 'remember' the parts of the terms and reductions that have been erased. For example, for each finite sequence of types $\boldsymbol{\alpha} = \alpha_1, \ldots, \alpha_n$, we could add a function symbol:

$$\mathsf{mem}_{\boldsymbol{\alpha}} : \alpha_1 \to \cdots \to \alpha_n \to \alpha_1.$$

The first argument of mem corresponds to 'normal' term; the other arguments are called the memory part and correspond to the erased terms. Every erasing rule can now be mapped to a non-erasing one. The rule $\mathsf{f}(x) \to \mathsf{c}$ can, for example, be replaced by $\mathsf{f}(x) \to \mathsf{mem}(\mathsf{c}, x)$. Rewriting with memory symbols can be elegantly defined, see for instance [25, Chap. II.4]. It does have different problems, however. For example, the non-erasing reductions can never be uniquely associated to the erasing reductions, because steps in the memory part of the reduction are erased.

Four

Finite Family Developments

4.1 Introduction

In this chapter, we prove that HRSs enjoy the property of Finite Family Developments (FFD). We prove the property by first proving the Prefix Property for HRSs, and then reducing FFD to it. The Prefix Property, in turn, is proved by reducing it to a prefix property for a λ-calculus with explicit substitutions, the λx-calculus.

The result of this chapter is not directly related to equivalence of reductions, but it is required to prove the results of the next two chapters.

Prefix property. The Prefix Property says that, given a step, the ancestor of a prefix of the target is a prefix of the source. Consider, as an example, the (first-order) rewriting system with the single rule $\mathsf{f}(x) \to \mathsf{g}(\mathsf{f}(x), x)$ and the step

$$\mathsf{f}(\mathsf{h}(\mathsf{a})) \to \mathsf{g}(\mathsf{f}(\mathsf{h}(\mathsf{a})), \mathsf{h}(\mathsf{a})).$$

Now, $p = \mathsf{g}(\mathsf{f}(\Box), \mathsf{h}(\Box))$ is a prefix of the target. Intuitively, its ancestor is $\mathsf{f}(\mathsf{h}(\Box))$, because it is the smallest prefix q of the source such that $q \to p'$ and p is contained in p'. Observe:

$$q = \mathsf{f}(\mathsf{h}(\Box)) \to \mathsf{g}(\mathsf{f}(\mathsf{h}(\Box)), \mathsf{h}(\Box)) = p'$$

Indeed, $q = \mathsf{f}(\mathsf{h}(\Box))$ is a prefix of the source.

Many different prefix properties are possible: we can, for example, vary in how the notions of prefix and ancestor are formalized, and we may impose additional conditions on the form of the prefixes. Prefix properties are already known for first-order TRSs [5, 53] and (a labelling of) the λ-calculus with β-reductions [5], and have many applications, such as (head) needed reductions [53, Chap. 8] and normalization of outermost-fair reductions [53, Chap. 9]. A similar property is proved in Van Daalen's Square Brackets Lemma [11].

Finite Family Developments. The Finite Family Developments property states that each reduction in which the creation depth, or family, of function symbols is bounded, is finite. The intuition behind the notion of creation depth is that in a step $C[l^\sigma] \to C[r^\sigma]$, the symbols of r depend on the symbols of l, and therefore have a higher creation depth, while the symbols in the context C and substitution σ do not take part in the step and have the same creation depth in both source and target.

Consider, for example, the following infinite (first-order) reduction, using the rewrite system above, where the function symbols are labelled with their creation depth:

$$\mathsf{f}^0(\mathsf{a}^0) \to \mathsf{g}^1(\mathsf{f}^1(\mathsf{a}^0), \mathsf{a}^0) \to \mathsf{g}^1(\mathsf{g}^2(\mathsf{f}^2(\mathsf{a}^0), \mathsf{a}^0)) \to \\ \mathsf{g}^1(\mathsf{g}^2(\mathsf{g}^3(\mathsf{f}^3(\mathsf{a}^0), \mathsf{a}^0), \mathsf{a}^0)) \to \mathsf{g}^1(\mathsf{g}^2(\mathsf{g}^3(\mathsf{g}^4(\mathsf{f}^4(\mathsf{a}^0), \mathsf{a}^0), \mathsf{a}^0), \mathsf{a}^0)) \to \cdots$$

Clearly, in this infinite reduction, the creation depth of the f's grows without bound. In a rewriting paradigm which enjoys the FFD property, restricting the creation depth to a finite bound yields finite reductions. FFD is a useful tool to prove various properties of rewrite systems, such as termination (for example, termination of the simply typed λ-calculus follows from FFD; see Sect. 4.6.2) and the existence of standard reductions (see Chapter 5), etc.

Remark. The material of this chapter was previously published as [9].

4.2 Labelling HRSs with natural numbers

In the introduction, we already labelled the function symbols with their creation depth. In this section, we formalize this idea.

Labelling (or marking) rewriting systems is a well-known method to formalize the notion of redex family; see for example [29, 30]. Here, we give a labelling, in the sense of [42, 43], for HRSs, analogous to the labelling for the λ-calculus used by Hyland [20] and Wadsworth [54]. Each function symbol is labelled by a natural number, representing the creation depth of the function symbol, and the rules are labelled such that every function symbol of the right-hand side is labelled with the largest label of the left-hand side plus one.

Definition 4.2.1 (ω-labelling)**.** Let $\mathfrak{H} = \langle \Sigma, R \rangle$ be an HRS.

(i) The *ω-labelling* of the signature Σ is defined as:

$$\Sigma^\omega := \{f^\ell \mid f \in \Sigma, \ell \in \mathbb{N}\}$$

The projection operation $[\![\cdot]\!]_\omega^\Sigma$ is the mapping from Σ^ω to Σ given by $[\![f^\ell]\!]_\omega^\Sigma := f$. The operation is homomorphically extended to terms.

(ii) The *creation depth* of a term s, denoted $\mathsf{D}(s)$, is the largest label of s, that is:

$$\mathsf{D}(s) := \max\{\ell \mid f^\ell \in \mathcal{S}ym(s)\}$$

(iii) Let s be a term, and $\ell \in \mathbb{N}$ a label. Then:

$$x(s_1, \ldots, s_n)^\ell := x(s_1^\ell, \ldots, s_n^\ell)$$
$$f(s_1, \ldots, s_n)^\ell := f^\ell(s_1^\ell, \ldots, s_n^\ell)$$
$$(\lambda x.s_0)^\ell := \lambda x.s_0^\ell$$

The *initial labelling* of a term s is given by the map I^ω, which is defined as follows: $\mathrm{I}^\omega(s) := s^0$.

(iv) The ω-labelling R^ω of the set of rules R contains, for every rule $\rho \in R$, where $\rho : l \to r$, and every Σ^ω-term l' such that $[\![l']\!]_\omega^\Sigma = l$, the rule $\rho_{l'} : l' \to r^{(\mathsf{D}(l')+1)}$.

The projection operation $[\![\cdot]\!]_\omega^R$ is the mapping from R^ω to R given by $[\![\rho_s]\!]_\omega^R := \rho$.

(v) The ω-labelled version of $\mathfrak{H}$ is defined as: $\mathfrak{H}^\omega := \langle \Sigma^\omega, R^\omega \rangle$

(vi) The creation depth of a step φ is the highest label of its target, and the creation depth of a reduction $\mathcal{R}$ is the highest label of its steps:

$$\mathsf{D}(\varphi) := \mathsf{D}(\mathrm{tgt}(\varphi))$$
$$\mathsf{D}(\mathcal{R}) := \max_{\varphi \in \mathcal{R}} \mathsf{D}(\varphi)$$

We write $[\![\cdot]\!]_\omega$ for $[\![\cdot]\!]_\omega^\Sigma \cup [\![\cdot]\!]_\omega^R$. This map is homomorphically extended to terms (just as in item (i) of the definition above), proof terms and reductions. Note that $\mathsf{D}(\mathcal{R})$ may be undefined (viz. if $\mathcal{R}$ is an infinite reduction). However, for terms s and steps φ, $\mathsf{D}(s)$ and $\mathsf{D}(\varphi)$ are always defined.

The ω-labelling only labels *function symbols*, not variables, abstractions or applications. The reason for this is that we want the ω-labelling of an HRS to be an HRS itself (otherwise it would not be a labelling in the sense of [42, 43]). Labelling variables is impossible, because α-equivalent terms are identified. Labelling abstractions and applications is impossible because we have fixed the (unlabeled) simply typed λ-calculus as substitution calculus. The consequence of not labelling abstractions and applications is that some extra work has to be done for HRSs which contain collapsing rules. The reason for this is that applying such a rule may create a redex which does not contain any function symbols whose label were increased by the application of the rule. In Sect. 4.4, this problem will be investigated in more detail and solved. Until then, we often restrict our attention to non-collapsing HRSs.

Example 4.2.2. Consider the HRS $\mathfrak{Map}_{\mathsf{e}}$:

$$\mathsf{map}(\lambda x.z(x), \mathsf{nil}) \to \mathsf{nil}$$
$$\mathsf{map}(\lambda x.z(x), \mathsf{cons}(u, v)) \to \mathsf{cons}(z(\mathsf{e}(u)), \mathsf{map}(\lambda x.z(x), v))$$

This HRS is a slight adaptation of the HRS $\mathfrak{Map}$ from page 2.4.7. The e function symbol is included in the second rule in order to make sure that the HRS is

non-collapsing. (In Sect. 4.5, a general method to produce a non-collapsing HRS out of a collapsing one is developed.) The labelled HRS $\mathfrak{M}ap^{\omega}_{\mathsf{e}}$ consists, among others, of the following rules:

$$\begin{aligned}
\mathsf{map}^0(\lambda x.z(x), \mathsf{nil}^0) &\to \mathsf{nil}^1\\
\mathsf{map}^1(\lambda x.z(x), \mathsf{nil}^1) &\to \mathsf{nil}^2\\
&\dots\\
\mathsf{map}^0(\lambda x.z(x), \mathsf{cons}^0(u,v)) &\to \mathsf{cons}^1(z(\mathsf{e}^1(u)), \mathsf{map}^1(\lambda x.z(x), v))\\
\mathsf{map}^0(\lambda x.z(x), \mathsf{cons}^1(u,v)) &\to \mathsf{cons}^2(z(\mathsf{e}^2(u)), \mathsf{map}^2(\lambda x.z(x), v))\\
&\dots
\end{aligned}$$

A labelled reduction is the following:

$$\begin{aligned}
&\mathsf{map}^0(\lambda x.\mathsf{f}^0(x), \mathsf{cons}^0(\mathsf{a}^0, \mathsf{nil}^0))\\
=_\beta\ &\underline{(\lambda zuv.\mathsf{map}^0(\lambda x.z(x), \mathsf{cons}^0(u,v)))}(\lambda x'.\mathsf{f}^0(x'), \mathsf{a}^0, \mathsf{nil}^0)\\
\to_{\mathfrak{M}ap}\ &\underline{(\lambda zuv.\mathsf{cons}^1(z(\mathsf{e}^1(u)), \mathsf{map}^1(\lambda x.z(x), v)))}(\lambda x'.\mathsf{f}^0(x'), \mathsf{a}^0, \mathsf{nil}^0)\\
=_\beta\ &\mathsf{cons}^1(\mathsf{f}^0(\mathsf{e}^1(\mathsf{a}^0)), \mathsf{map}^1(\lambda x.\mathsf{f}^0(x), \mathsf{nil}^0))\\
=_\beta\ &\mathsf{cons}^1(\mathsf{f}^0(\mathsf{e}^1(\mathsf{a}^0)), \underline{(\lambda z.\mathsf{map}^1(\lambda x.z(x), \mathsf{nil}^0))}(\lambda x'.\mathsf{f}^0(x')))\\
\to_{\mathfrak{M}ap}\ &\mathsf{cons}^1(\mathsf{f}^0(\mathsf{e}^1(\mathsf{a}^0)), \underline{(\lambda z.\mathsf{nil}^2)}(\lambda x'.\mathsf{f}^0(x')))\\
=_\beta\ &\mathsf{cons}^1(\mathsf{f}^0(\mathsf{e}^1(\mathsf{a}^0)), \mathsf{nil}^2)
\end{aligned}$$

Notice how only the labels of function symbols involved in the step (that is, the underlined ones) are increased.

The following two lemmas provide a correspondence between labelled and unlabeled reductions:

Lemma 4.2.3. *Let $\mathfrak{H}$ be an* HRS. *$\mathfrak{H}^{\omega}$ is orthogonal/collapsing/erasing, if and only if $\mathfrak{H}$ is.*

Proof. By induction on the length of s we easily prove that s^{ℓ} is orthogonal/collapsing/erasing, if and only if s is, where in the case of orthogonality we take into account that s^{ℓ} is uniformly labelled. □

Lemma 4.2.4. *Let $\mathfrak{H} = \langle \Sigma, R\rangle$ be an* HRS.

(i) Let s be a Σ-term and s' a Σ^{ω}-term with $[\![s']\!]_{\omega} = s$. For each $\mathfrak{H}$-step φ with source s, there is a unique $\mathfrak{H}^{\omega}$-step φ' with source s' such that $[\![\psi]\!]_{\omega} = \varphi$.

(ii) If $\varphi : s \multimap_{\mathfrak{H}^{\omega}} t$, then $[\![\varphi]\!]_{\omega} : [\![s]\!]_{\omega} \multimap_{\mathfrak{H}} [\![t]\!]_{\omega}$.

Proof. (i) By induction on the inference of $\varphi : s \multimap_{\mathfrak{H}} t$. The interesting case is when $\varphi = \rho(\varphi_1, \dots, \varphi_n)$, for some $\rho : l \to r$ and $\varphi_i : s_i \multimap_{\mathfrak{H}} t_i$. It must be the case that $s = l(s_1, \dots, s_n)$ and $s' = l'(s'_1, \dots, s'_n)$, where $[\![l']\!]_{\omega} = l$

and $[\![s'_i]\!]_\omega = s_i$. By induction hypothesis there are unique φ'_i with source s'_i such that $[\![\varphi'_i]\!]_\omega = \varphi_i$. We take $\varphi' := \rho_{l'}(\varphi'_1, \ldots, \varphi'_n)$, which satisfies the requirements of the lemma.

(ii) By induction on the inference of $\varphi : s \multimap_{\mathfrak{H}^\omega} t$. The interesting case is when $\varphi = \rho_l(\varphi_1, \ldots, \varphi_n)$, for some $\rho_l : l \to r$ and $\varphi_i : s_i \multimap_{\mathfrak{H}^\omega} t_i$, $i \in \{1, \ldots, n\}$. By induction hypothesis, $[\![\varphi_i]\!]_\omega : [\![s_i]\!]_\omega \multimap_{\mathfrak{H}} [\![t_i]\!]_\omega$. From this and the fact that $[\![\rho_l]\!]_\omega : [\![l]\!]_\omega \to [\![r]\!]_\omega$, it follows that

$$[\![s]\!]_\omega = [\![l(s_1, \ldots, s_n)]\!]_\omega \multimap_{\mathfrak{H}} [\![r(t_1, \ldots, t_n)]\!]_\omega = [\![t]\!]_\omega. \qquad \square$$

The results of Lemma 4.2.4 easily generalize to reductions. This allows us to canonically map unlabeled reductions to labelled ones: we take the one with the initial labelling as source. Formally:

Definition 4.2.5. Let $\mathfrak{H}$ be an HRS, and $\mathcal{R}$ a $\mathfrak{H}$-reduction. The canonical ω-labelling $\mathcal{R}^\omega$ is the unique labelling such that $[\![\mathcal{R}^\omega]\!]_\omega = \mathcal{R}$ and $\mathrm{src}(\mathcal{R}^\omega) = \mathrm{I}^\omega(\mathrm{src}(\mathcal{R}))$.

We can use this canonical labelling to extend the notion of creation depth to unlabeled reductions: if $\mathcal{R}$ is an unlabeled reduction, then $\mathsf{D}(\mathcal{R}) := \mathsf{D}(\mathcal{R}^\omega)$.

Until now, we included multisteps as well as proper steps. In the rest of this chapter, however, we will often restrict our attention to proper steps and proper reductions. (By Theorems 3.2.8 and 3.3.5 the results are nonetheless easily generalized to multistep reductions.)

4.3 The Prefix Property

We call p a prefix of term t, if it is a pattern, and there exists a substitution σ such that $p^\sigma = t$. Given a step $s \to t$, a subterm q of s is the ancestor of a subterm p of t, if the symbols of t 'trace to' the symbols of s. This notion is formalized here using labelling together with the rewrite relation: q is an ancestor of p, if $\mathsf{D}(p) \geq \mathsf{D}(q)$ and $q \twoheadrightarrow_{\mathfrak{H}^\omega} p^\upsilon$. The substitution υ is necessary because q might reduce to a 'bigger' term than p; typically, υ has only function symbols which are also in p. Using these formalizations, we prove in this section the following theorem (the proof itself begins on page 64):

Theorem 4.3.1 (Prefix Property)**.** *Let $\mathfrak{H}^\omega$ be the ω-labelling of a non-collapsing* HRS*, s a term, p a local $\overline{x}$-pattern and σ a substitution. If $s \to_{\mathfrak{H}^\omega} p^\sigma$, then there exist a local $\overline{x}$-pattern q and a substitution τ, such that $s = q^\tau$, $\mathsf{D}(p) \geq \mathsf{D}(q)$, and either:*

- *$q \to_{\mathfrak{H}^\omega} p^\upsilon$, for some substitution υ such that $\upsilon ; \tau = \sigma$; or* *(trm)*
- *$q = p$ and $\tau \to_{\mathfrak{H}^\omega} \sigma$.* *(sub)*

The theorem states that, given a prefix of the target, its ancestor is a prefix of the source. There are two possibilities: either the prefix takes part in the step, or the step occurred fully in the substitution. Note that, in the first case,

we do not only require that its ancestor is a prefix, but also that the suffix stays the same (except for duplicated subterms). In this regard, the lemma is stronger than the prefix property (for the λ-calculus) proved in [5, Prop. 7.4].

Example 4.3.2. Consider the following $\mathfrak{Map}^\omega$-step (see page 26):

$$\mathsf{h}^1(\mathsf{map}^3(\lambda x.\mathsf{f}^2(x), \mathsf{cons}^2(\mathsf{a}^5, \mathsf{nil}^1)) \to \\ \mathsf{h}^1(\mathsf{cons}^4(\mathsf{f}^2(\mathsf{e}^4(\mathsf{a}^5)), \mathsf{map}^4(\lambda x.\mathsf{f}^2(x), \mathsf{nil}^1)))$$

First, let the following prefix p and suffix σ be given:

$$p := \mathsf{h}^1(\mathsf{cons}^4(\mathsf{f}^2(y_1), y_2))$$
$$\sigma := [y_1 \mapsto \mathsf{e}^4(\mathsf{a}^5), y_2 \mapsto \mathsf{map}^4(\lambda x.\mathsf{f}^2(x), \mathsf{nil}^1)]$$

Then the following assignments satisfy the conditions of the (trm) case:

$$q := \mathsf{h}^1(\mathsf{map}^3(\lambda x.\mathsf{f}^2(x), \mathsf{cons}^2(y_1, z_2)))$$
$$\upsilon := [y_1 \mapsto \mathsf{e}^4(z_1), y_2 \mapsto \mathsf{map}^4(\lambda x.\mathsf{f}(x), z_2)]$$
$$\tau := [z_1 \mapsto \mathsf{a}^5, z_2 \mapsto \mathsf{nil}^1]$$

Second, let:

$$p := \mathsf{h}(y)$$
$$\sigma := [y \mapsto \mathsf{cons}^4(\mathsf{f}^2(\mathsf{e}^4(\mathsf{a}^5)), \mathsf{map}^4(\lambda x.\mathsf{f}^2(x), \mathsf{nil}^1)))]$$

Then:

$$q := \mathsf{h}^1(y)$$
$$\tau := [y \mapsto \mathsf{map}^3(\lambda x.\mathsf{f}^2(x), \mathsf{cons}^2(\mathsf{a}^5, \mathsf{nil}^1))]$$

satisfy the conditions of the (sub) case.

The interesting case in the proof of the Prefix Property is the case that the step $s \to_{\mathfrak{H}^\omega} p^\sigma$ occurs at the head. In this case we have that $s = l^\rho$ and $p^\sigma = r^\rho$, for some rule $l \to r$ and substitution ρ. This situation is depicted in Fig. 4.1. In this case, we want to construct an ancestor q that satisfies the (trm) case. It makes sense to try to do this by adding to the pattern l the parts of the pattern p that are not in r (depicted in the figure by υ), and showing that the resulting term is a pattern again. However, due to the implicit β-conversions, these "parts of p that are not in r" are not easily obtained.

The key observation is that the β-reduction from p^σ to normal form is of a particularly simple form; basically, it carries out a variable renaming, because p is a pattern and has only bound variables as arguments of free variables. The trick is to translate the prefix and suffix in such a way, that the variable renamings, in a sense, are already carried out (we need variable capturing, first-order substitutions for this, called *graftings*), trace the prefix back over

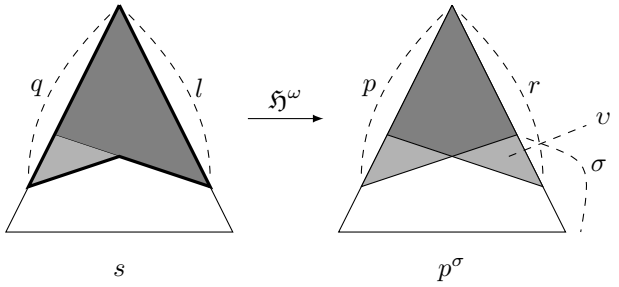


Figure 4.1: The interesting case in the proof of the Prefix Property for HRSs: the step occurs at the head. *We want to construct the prefix q of s by taking the left-hand side of the rule l plus the "parts of p that are not in r", denoted by υ in the figure.*

the β-reduction from r^ρ to normal form, and find the prefix's ancestor, which is a prefix of r^ρ. Now, we are dealing with terms that are exactly equal, instead of only equal up to β-equality, and now the problem can be solved by using first-order unification techniques, which are well investigated in the existing literature.

The above sketch of a proof technique suggests that we need to prove a prefix property for β-reductions in the λ-calculus. This is difficult, however, since the λ-calculus does not cope well with graftings, because of the global nature of substitution. Let me illustrate the problem with the following example. Let $C := (\lambda x.\Box)\mathsf{a}$, $D := \Box$ and $s := x$. Then $C \to_\beta D$, and $C[s] \to_\beta \mathsf{a}$, because the x in s is captured by the abstraction in the context and substituted for. However, $D[s] = x$ and thus $C[s] \not\to_\beta D[s]$.

To tackle this problem, we use a λ-calculus with explicit substitutions (a variant of the λx-calculus), and prove a prefix property for it. Then, we simulate β-equality with this new calculus. In [12] a similar approach is taken with respect to higher-order unification.

4.3.1 The Prefix Property of the λx-calculus

We use a variant of the λx-calculus [7], with explicit renamings. The calculus has both object variables (x, y, z) and metavariables (X, Y, Z). In the following, we will refer to this calculus simply as "λx-calculus". The terms of the λx-calculus over some signature Σ are first-order terms derivable by the following grammar:

$$\Lambda_{\mathrm{x}} := x \mid X \mid f \mid \Lambda_{\mathrm{x}}\Lambda_{\mathrm{x}} \mid \lambda x.\Lambda_{\mathrm{x}} \mid \Lambda_{\mathrm{x}}\{x\backslash\Lambda_{\mathrm{x}}\}$$

where $f \in \Sigma$ and the object variables are considered as constants or names. M, N will range over λx-terms. Terms of the form $M\{x\backslash N\}$ will be called *explicit substitutions*, and the $\{x\backslash N\}$ part of an explicit substitution is called a *closure*. With $\mathrm{MV}(M)$ we will denote the set of metavariables of M, and

with $\mathcal{S}ym(M)$ the set of function symbols of M. The reduction rules of the λx-calculus are:

$$\begin{aligned}
(\lambda x.M)N &\to_{\mathrm{B}} M\{x\backslash N\}\\
x\{x\backslash N\} &\to_{\mathrm{x}} N\\
y\{x\backslash N\} &\to_{\mathrm{x}} y\\
f\{x\backslash N\} &\to_{\mathrm{x}} f\\
(M_1M_2)\{x\backslash N\} &\to_{\mathrm{x}} M_1\{x\backslash N\}M_2\{x\backslash N\}\\
(\lambda x.M)\{x\backslash N\} &\to_{\mathrm{x}} \lambda x.M\\
(\lambda y.M)\{x\backslash N\} &\to_{\mathrm{x}} \lambda z.M\{y\backslash z\}\{x\backslash N\}
\end{aligned}$$

where $x \neq y$ and z is a fresh object variable. The subcalculus x consists of all rules except the B-rule. The reduction relations $\to_{\mathrm{Bx}}$ and $\to_{\mathrm{x}}$ are the contextual closures of the above steps. Note that there is no reduction rule for terms of the form $X\{x\backslash N\}$, where X is a metavariable, and thus x-normal forms are characterized by the fact that sequences of closures are only applied to metavariables.

A λx-term is called *passive* if metavariables are never applied to something, that is, the λx-term does not contain any subterms of the form $X\overline{\mu}(M_1,\ldots,M_n)$, where X is a metavariable, $\overline{\mu}$ is a sequence of closures and $n \geq 1$; it is called *linear* if every metavariable occurs in it at most once. In the following P, Q will range over linear, passive λx-terms.

Lemma 4.3.3. *The* x*-calculus is confluent and terminating.*

Proof. Confluence follows because the calculus is clearly orthogonal. Termination follows by the observation that each x-rule strictly decreases the size of the argument of the explicit substitution, and hence a strictly decreasing multiset ordering can be devised. □

Remark 4.3.4. Although, by the previous lemma, the x-calculus is confluent, it is well-known that the λx-calculus is *not* confluent on terms containing metavariables, as is witnessed by the following counterexample:

$$\begin{array}{lll}
(\lambda x.(\lambda y.Z)Y)X & & (\lambda x.(\lambda y.Z)Y)X\\
\twoheadrightarrow_{\mathrm{Bx}} & (\lambda y'.Z\{y\backslash y'\}\{x\backslash X\})Y\{x\backslash X\} & \twoheadrightarrow_{\mathrm{Bx}} \quad (\lambda x.Z\{y\backslash Y\})X\\
\twoheadrightarrow_{\mathrm{Bx}} & Z\{y\backslash y'\}\{x\backslash X\}\{y'\backslash Y\{x\backslash X\}\} & \twoheadrightarrow_{\mathrm{Bx}} \quad Z\{y\backslash Y\}\{x\backslash X\}
\end{array}$$

At first sight, non-confluence seems problematic, because we're trying to use the λx-calculus to simulate the (confluent) λ-calculus. However, the translation to λ-calculus (see page 60) will remove all closures, and will project normal forms of the same λx-term to the same λ-term (modulo α-equivalence).

A *grafting* is a mapping from metavariables to λx-terms. The greek lowercase letters $\zeta, \eta, \theta, \kappa$ will range over graftings. Applying a grafting ζ to a term M,

written $M[\zeta]$, is defined exactly as first order substitution, that is:

$$\begin{aligned} x[\zeta] &:= x \\ X[\zeta] &:= \zeta'(X) \\ f[\zeta] &:= f \\ (M_1 M_2)[\zeta] &:= M_1[\zeta] M_2[\zeta] \\ (\lambda x.M)[\zeta] &:= \lambda x.M[\zeta] \\ (M\{x\backslash N\})[\zeta] &:= M[\zeta]\{x\backslash N[\zeta]\} \end{aligned}$$

where $\zeta'(X) := \zeta(X)$, if $X \in Dom(\zeta)$, and $\zeta'(X) := X$, otherwise. A grafting is called *linear*, if every metavariable occurs in its codomain only once, that is, its codomain consists of linear λx-terms with mutually disjoint metavariables. A grafting is called *passive*, if all the terms of its codomain are passive.

Because λx-terms are first-order terms, unification is decidable, by using one of the various well-known first-order unification algorithms. In the proof of the Prefix Property, we need the following property: if two λx-terms are unifiable, there exists a most general unifier (mgu), and, if we assume the unifiable terms to be linear and passive, then the mgu applied to one of the terms is a linear, passive λx-term again:

Lemma 4.3.5. *Let M, N be linear* λx*-terms, where* $\mathrm{MV}(M)$ *and* $\mathrm{MV}(N)$ *are disjoint, and let ζ, η be graftings such that $M[\zeta] = N[\eta]$. There exist graftings ζ_0, η_0, κ such that $M[\zeta_0] = N[\eta_0]$, $\zeta_0 ; \kappa = \zeta$, $\eta_0 ; \kappa = \eta$, $\mathcal{S}ym(\zeta_0) \subseteq \mathcal{S}ym(N)$ and $\mathcal{S}ym(\eta_0) \subseteq \mathcal{S}ym(M)$. Moreover, if M (N) is passive, then η_0 (ζ_0) consists of passive* λx*-terms.*

Proof. Since we consider λx-terms here as first-order terms, the (first part) of the lemma is essentially an instance of first-order unification.

By assumption, M and N are unified by $\langle \mu, \nu \rangle$ (the assumption that the metavariables are disjoint allows us to consider ζ and η together as a unifier), and thus have a most general unifier (mgu). We take $\langle \zeta_0, \eta_0 \rangle$ to be this mgu. The desired κ exists because any unifier is an instance of an mgu.

Assume $f \in \mathcal{S}ym(\eta_0)$, but $f \notin \mathcal{S}ym(M)$. Then it must be the case that $f \in \mathcal{S}ym(\zeta_0)$. Since all the subterms of f must be in the graftings too, there exist graftings $\zeta_0', \eta_0', \kappa'$ such that $\zeta_0' ; \kappa = \zeta_0$ and $\eta_0' ; \kappa = \eta_0$. It follows by linearity that $\langle \zeta_0', \eta_0' \rangle$ is a unifier, contradicting the assumption. For the assumption that $f \in \mathcal{S}ym(\zeta_0)$, but $f \notin \mathcal{S}ym(N)$, a contradiction is derived through a symmetrical argument.

For the second part of the lemma, suppose M is passive, but $\eta_0(X)$ is not. Then $N[\eta_0]$ is not passive, and contains a subterm of the form $N_0 = X\overline{\mu}(N_1, \ldots, N_n)$. Because $M[\zeta_0] = N[\eta_0]$, so does $M[\zeta_0]$. There are two possibilities: if the root application of N_0 is in M, then M is not passive, contradicting the assumption; otherwise, N_0 must be a subterm of $\zeta_0(Y)$, for some $Y \in Dom(\mu_0)$, but then $\langle \zeta_0, \eta_0 \rangle$ is not a mgu. Again, a symmetrical arguments yields the desired result if N is passive, but $\mu_0(X)$ is not. □

Example 4.3.6. Let:

$$M := \lambda x.\mathsf{g}(\mathsf{f}_1(X_1), X_2) \qquad N := \lambda x.\mathsf{g}(Y_1, \mathsf{f}_2(Y_2))$$
$$\zeta := [X_1 \mapsto \mathsf{a}, X_2 \mapsto \mathsf{f}_2(\mathsf{a})] \qquad \eta := [Y_1 \mapsto \mathsf{f}_1(\mathsf{a}), Y_2 \mapsto \mathsf{a}]$$

Then $M[\zeta] = \lambda x.\mathsf{g}(\mathsf{f}_1(\mathsf{a}), \mathsf{f}_2(\mathsf{a})) = N[\eta]$. We take:

$$\zeta_0 := [X_2 \mapsto \mathsf{f}_2(Z_1)]$$
$$\eta_0 := [Y_1 \mapsto \mathsf{f}_1(Z_2)]$$
$$\kappa := [Z_i \mapsto \mathsf{a}]$$

to satisfy the conditions of the lemma.

In the next theorem, we prove the Prefix Property of the λx-calculus. P is a prefix of a λx-term M, if it is a linear, passive λx-term, and there exists a grafting ζ such that $P[\zeta] = M$. The notion of ancestor is again formalized using labelling and the rewrite relation; however, because we do not count creation depth in Bx-reductions, now the labels, or more generally, the function symbols of the prefix must be the same as those of its ancestor. Just like in Theorem 4.3.1, a prefix can either take part in the step, or not, resulting in two cases. Item (ii) is the extension of the Prefix Property to Bx-*reductions*.

Theorem 4.3.7 (λx-Prefix Property)**.** *Let M be a closed λx-term, P a linear, passive λx-term and ζ a grafting.*

(i) If $M \to_{\mathrm{Bx}} P[\zeta]$, then there exist a linear, passive λx-term Q and a grafting η such that $M = Q[\eta]$, $\mathcal{S}ym(Q) = \mathcal{S}ym(P)$ and either:

- *$Q \to_{\mathrm{Bx}} P[\kappa]$ where κ is some grafting such that $\kappa\,;\eta = \zeta$; or (trm)*
- *$Q = P$ and $\eta \to_{\mathrm{x}} \zeta$. (sub)*

(ii) If $M \twoheadrightarrow_{\mathrm{Bx}} P[\zeta]$, then there exist a linear, passive λx-term Q and a grafting η such that $M = Q[\eta]$, $\mathcal{S}ym(Q) = \mathcal{S}ym(P)$ and $Q \twoheadrightarrow_{\mathrm{Bx}} P[\kappa]$, where κ is some grafting such that $\kappa\,;\eta \twoheadrightarrow_{\mathrm{Bx}} \zeta$.

Proof. (i) By induction on the context of the step $M \to_{\mathrm{Bx}} P[\zeta]$. In this proof, let id_ζ be the identity grafting on the domain of a grafting ζ, that is $\mathsf{id}_\zeta := [X \mapsto X \mid X \in Dom(\zeta)]$.

If $P = X$, then we take $Q := X$ and $\eta := \zeta[X \mapsto M]$, satisfying the (sub) case of the lemma. If the step does not occur at the head of the term, then the lemma follows simply from the induction hypothesis. Otherwise we look at the reduction rule which was applied (at the head).

The interesting case is when a closure is distributed over an application, that is

$$M = (M_1 M_2)\{x\backslash N\} \to_{\mathrm{x}} (M_1\{x\backslash N\})(M_2\{x\backslash N\}) = P[\zeta].$$

Since P is not a metavariable, $P = P_1 P_2$, where $P_1[\zeta] = M_1\{x\backslash N\}$ and $P_2[\zeta] = M_2\{x\backslash N\}$. For P_2, there are two possible cases:

(a) If $P_2 = X$, for some metavariable X, then let $\kappa' := \mathsf{id}_\zeta ; [X \mapsto Y\{x\backslash Z\}]$ and $\zeta' := \zeta[X \mapsto X, Y \mapsto M_2, Z \mapsto N]$, where Y, Z are fresh metavariables.

(b) Otherwise, let $\kappa' := \mathsf{id}_\zeta$ and $\zeta' := \zeta$.

In both cases, $P_2[\kappa'] = Q_2\{x\backslash R_2\}$, for some passive λx-terms Q_2 and R_2, and $\kappa' ; \zeta' = \zeta$.

Since P is passive, we know that P_1 is not of the form $X\overline{\mu}$, where $\overline{\mu}$ is a list of substitutions. In particular, P is not of the form X. Thus, $P_1 = Q_1\{x\backslash R_1\}$. Because P is linear, and the $\kappa'(X) = X$ for $X \in \mathrm{MV}(P_1)$, $P_1[\kappa'] = P_1$. Also, because in case (a), Y and Z are fresh metavariables, and in case (b), $\zeta' = \zeta$, it holds that $P_1[\zeta'] = P_1[\zeta]$.

Because P is linear, and R_1 and R_2 are different subterms of P, R_1 and R_2 are linear and have no metavariables in common. Since $R_1[\zeta'] = N = R_2[\zeta']$, we can apply Lemma 4.3.5, and obtain graftings ζ_1, ζ_2, η such that:

$$\begin{array}{ll} R_1[\zeta_1] = R_2[\zeta_2] & \mathcal{S}ym(\zeta_1) \subseteq \mathcal{S}ym(R_2) \\ \zeta_1 ; \eta = \zeta' & \mathcal{S}ym(\zeta_2) \subseteq \mathcal{S}ym(R_1) \\ \zeta_2 ; \eta = \zeta' & \end{array}$$

Let $R := R_1[\zeta_1]$ $(= R_2[\zeta_2])$. Because it is the case that $\mathcal{S}ym(\zeta_1) \subseteq \mathcal{S}ym(R_2)$ and $\mathcal{S}ym(\zeta_2) \subseteq \mathcal{S}ym(R_1)$, it holds that $\mathcal{S}ym(R) = \mathcal{S}ym(R_1) \cup \mathcal{S}ym(R_2)$. Next, let $\kappa_i = (\kappa' ; \zeta_i)\restriction \mathrm{FV}(P_i)$, for $i \in \{1, 2\}$. Since P is linear, κ_1 and κ_2 have disjoint domains. Let $\kappa := \kappa_1 \cup \kappa_2$. We define $Q := (Q_1Q_2)\{x\backslash R\}$. Now,

$$Q = (Q_1Q_2)\{x\backslash R\} \to_{\mathrm{x}} (Q_1\{x\backslash R\})(Q_2\{x\backslash R\}) = P_1[\kappa' ; \zeta_1]P_2[\kappa' ; \zeta_2] = P[\kappa].$$

Also, suppose $X \in \mathit{Dom}(\kappa)$. There are two subcases. If $X \in \mathrm{MV}(P_2)$, then $(\kappa ; \eta)(X) = (\kappa' ; \zeta_2 ; \eta)(X) = (\kappa' ; \zeta')(X) = \zeta(X)$. Otherwise, $\kappa'(X) = X$ and $\zeta'(X) = \zeta(X)$. So $\kappa ; \eta = \zeta$. Finally,

$$\mathcal{S}ym(P) = \mathcal{S}ym(Q_1) \cup \mathcal{S}ym(Q_2) \cup \mathcal{S}ym(R_1) \cup \mathcal{S}ym(R_2) = \\ \mathcal{S}ym(Q_1) \cup \mathcal{S}ym(Q_2) \cup \mathcal{S}ym(R) = \mathcal{S}ym(Q).$$

and so the requirements of the (trm) case are satisfied.

(ii) By induction on the length of the reduction sequence $M \twoheadrightarrow_{\mathrm{Bx}} P[\zeta]$. In the base case, if $M = P[\zeta]$, we take $Q := P$ and $\eta := \zeta$, satisfying the lemma with $\kappa := \varnothing$.

Otherwise, let $M \twoheadrightarrow_{\mathrm{Bx}} N \to_{\mathrm{Bx}} P[\zeta]$. We apply item (i), obtaining a passive λx-term P' and grafting ζ' such that $M = P'[\zeta']$, $\mathcal{S}ym(P') = \mathcal{S}ym(P)$ and either the (trm) or (sub) case applies. Then we apply the induction hypothesis on the reduction $M \twoheadrightarrow_{\mathrm{Bx}} P'[\zeta']$, obtaining a passive λx-term Q and grafting η such that $M = Q[\eta]$, $\mathcal{S}ym(Q) = \mathcal{S}ym(P')$ $(= \mathcal{S}ym(P))$, $Q \twoheadrightarrow_{\mathrm{Bx}} P'[\kappa]$ for some grafting κ such that $\kappa ; \eta \twoheadrightarrow_{\mathrm{Bx}} \zeta'$. Now we distinguish the following cases:

- *(trm)* If $P' \to_{\mathrm{Bx}} P[\kappa_0]$, for some κ_0 such that $\kappa_0 ; \zeta' = \zeta$, then

  $$Q \twoheadrightarrow_{\mathrm{Bx}} P'[\kappa] \twoheadrightarrow_{\mathrm{Bx}} P[\kappa_0 ; \kappa] \quad \text{and} \quad \kappa_0 ; \kappa ; \eta \twoheadrightarrow_{\mathrm{Bx}} \kappa_0 ; \zeta' = \zeta,$$

  as required.

- *(sub)* If $P' = P$ and $\zeta' \to_{\mathrm{x}} \zeta$, then

$$Q \twoheadrightarrow_{\mathrm{Bx}} P'[\kappa] = P[\kappa] \quad \text{and} \quad \kappa\,;\eta \twoheadrightarrow_{\mathrm{Bx}} \zeta' \twoheadrightarrow_{\mathrm{x}} \zeta,$$

  as required. □

Example 4.3.8. Consider the Bx-reduction

$$(\lambda x.\mathsf{g}(x,x))(\mathsf{f}(\mathsf{a})) \twoheadrightarrow_{\mathrm{Bx}} \mathsf{g}(\mathsf{f}(\mathsf{a}),\mathsf{f}(\mathsf{a}))$$

and the prefix $P = \mathsf{g}(\mathsf{f}(X), Y)$ of the target. This means that the suffix is $\zeta = [X \mapsto \mathsf{a}, Y \mapsto \mathsf{f}(\mathsf{a})]$. We can take:

- $Q := (\lambda x.\mathsf{g}(x,x))(\mathsf{f}(Y))$,
- $\kappa := [Y \mapsto \mathsf{f}(X)]$ and
- $\eta := [X \mapsto \mathsf{a}]$,

satisfying the conditions of Theorem 4.3.7 (ii).

4.3.2 Translating between terms, preterms and λx-terms

We are now dealing with three types of expressions: terms, preterms and λx-terms. Terms are equivalence classes of preterms, preterms are just λ-terms over a signature of function symbols, and λx-terms are preterms with explicit substitutions. In the following, it is often desired to make precise which type of expression we are talking about, and as a result, to define an explicit translation between the three types of expressions. Here, we define the operations which carry out these translations:

- $\cdot^\flat$ and $\cdot^\sharp$ from terms to preterms and back;
- $\cdot^\ominus$ and $\cdot^\oplus$ from preterms to λx-terms and back.

Translating between terms and preterms.

We introduce a pair of operations which interpret terms as preterms, and vice versa: with $s^\flat$ (s-flat), we denote (the unique representative of) the term s, interpreted as a preterm, and with $s^\sharp$ (s-sharp), we denote (the $\beta\overline{\eta}$-normal form of) the preterm s, interpreted as a higher-order term. These operations naturally generalize to substitutions. We will call a preterm s a (fully extended/linear/local) $\overline{x}$-prepattern, if it is in $\beta\overline{\eta}$-normal form, and $s^\sharp$ is a (linear/fully extended/local) $\overline{x}$-pattern.

Translating between preterms and λx-terms.

We assume a bijection between the metavariables of λx-terms and the variables of preterms. The following convention is employed: a metavariable X is associated with x, Y with y, etc. If necessary, α-Conversions will be carried out to avoid name clashes.

Now we can introduce the operations $\cdot^{\ominus}_{\overline{x}}$ and $\cdot^{\oplus}$, which map preterms to λx-terms and vice versa, as follows:[1]

$$
\begin{aligned}
y^{\ominus}_{\overline{x}} &:= Y \quad \text{if } y \notin \overline{x} & M^{\oplus} &:= (M{\downarrow_{\mathrm{x}}})^{\oplus}_{\mathrm{N}} \\
x^{\ominus}_{\overline{x}} &:= x \quad \text{if } x \in \overline{x} & (Y\overline{\sigma})^{\oplus}_{\mathrm{N}} &:= y \\
f^{\ominus}_{\overline{x}} &:= f & x^{\oplus}_{\mathrm{N}} &:= x \\
(\lambda y.s)^{\ominus}_{\overline{x}} &:= \lambda y.s^{\ominus}_{\overline{x}y} & f^{\oplus}_{\mathrm{N}} &:= f \\
(s_1 s_2)^{\ominus}_{\overline{x}} &:= (s_1)^{\ominus}_{\overline{x}}(s_2)^{\ominus}_{\overline{x}} & (\lambda y.M)^{\oplus}_{\mathrm{N}} &:= \lambda y.M^{\oplus} \\
& & (M_1 M_2)^{\oplus}_{\mathrm{N}} &:= M_1^{\oplus} M_2^{\oplus}
\end{aligned}
$$

Note that $\cdot^{\oplus}$ also normalizes the term to x-normal form and removes explicit substitutions, and that, for each preterm s and sequence of variables $\overline{x}$, $(s^{\ominus}_{\overline{x}})^{\oplus} = s$. The operations above are naturally generalized to translations between substitutions and graftings.

Lemma 4.3.9. *Let M, N be λx-terms. $M \twoheadrightarrow_{\mathrm{Bx}} N$ if and only if $M^{\oplus} \twoheadrightarrow_{\beta} N^{\oplus}$.*

Proof. ($\Rightarrow$) and ($\Leftarrow$) are proved by induction on the length of the reductions $M \twoheadrightarrow_{\mathrm{Bx}} N$ and $M^{\oplus} \twoheadrightarrow_{\beta} N^{\oplus}$, respectively, with a nested induction on the context of the step in the base case. □

Although the above lemma suggests that Bx-reduction in the λx-calculus can easily simulate β-reduction, there is still a problem: $\cdot^{\oplus}$ does not distribute properly over grafting application. The problem is similar to the problem given on page 55. Consider the following λx-term and grafting:

$$
\begin{aligned}
M &:= (\lambda x.\mathsf{f}(Y))\mathsf{a} \\
\zeta &:= [Y \mapsto x]
\end{aligned}
$$

Now it holds that:

$$
\begin{aligned}
M[\zeta]^{\oplus} &= (\lambda x.\mathsf{f}(x))\mathsf{a} \\
M^{\oplus} &= (\lambda x.\mathsf{f}(y))\mathsf{a} \\
\zeta^{\oplus} &= [y \mapsto x]
\end{aligned}
$$

Note that $(M^{\oplus})^{(\zeta^{\oplus})} = \lambda z.\mathsf{f}(x)$, because substitutions are capture-avoiding, and thus $M[\zeta]^{\oplus} \neq_{\beta} (M^{\oplus})^{(\zeta^{\oplus})}$.

The solution is to add as arguments to the free variables of the preterms as many (bound) variables as necessary (or more) to make the distribution

[1] Actually, the operation $\cdot^{\ominus}_{\overline{x}}$ is not a mapping, because α-equivalent preterms preterms are identified as usual, and thus identical preterms can be mapped to different λx-terms. This is not a problem in practice, and can be fixed in theory by considering only preterms in 'α-normal form', for example by consecutively numbering the bound variables from left to right.

work. In the example above we could take:

$$s := (\lambda x.\mathsf{f}(y(x)))\mathsf{a}$$
$$\sigma := [y \mapsto \lambda x.x]$$

Now, s and σ are, in a way that will be formalized in the next definition, 'similar' to M and ζ, but now $M[\zeta]^{\oplus} =_\beta s^\sigma$.

Definition 4.3.10. Let M be a λx-term and ζ a grafting. A tuple $\langle s, \sigma\rangle$ of preterm and substitution is a λ-*extension* of $\langle M, \zeta\rangle$ if there are graftings θ_1, θ_2 such that:

- $s = M[\theta_1]^{\oplus}$ and $\sigma = (\theta_2 \,;\, \zeta)^{\oplus}$;
- for each $X \in \mathrm{MV}(M)$, $\theta_1(X) = X(\overline{z})$ and $\theta_2(X) = \lambda\overline{z}.X$, where $\overline{z}$ is a list of variables containing at least the bound variables of M in scope that occur in $\zeta(X)$ (in arbitrary order).

The notion of λ-extension is, again, naturally generalized to graftings and substitutions as the first component of the tuples.

Lemma 4.3.11. *Let $\langle s, \sigma\rangle$ be a λ-extension of $\langle M, \zeta\rangle$. Then:*

(i) $s^\sigma =_\beta M[\zeta]^{\oplus}$*;*

(ii) for each λx*-term* N *such that* $M \twoheadrightarrow_{\mathrm{Bx}} N$, $s^\sigma =_\beta N[\zeta]^{\oplus}$.

Proof. (i) Follows from the fact that for all $X \in \mathrm{MV}(M)$ it holds that

$$(\theta_1 \,;\, \theta_2)\downarrow_{\mathrm{x}}(X) = X\{z_1\backslash z_1\}\cdots\{z_n\backslash z_n\}$$

and thus $s^\sigma =_\beta M[\theta_1 \,;\, \theta_2 \,;\, \zeta]^{\oplus} =_\beta M[\zeta]^{\oplus}$.

(ii) Assume that $M \twoheadrightarrow_{\mathrm{Bx}} N$. Let $t = N[\theta_1]^{\oplus}$. From Lemma 4.3.9, the fact that $M[\theta_1] \twoheadrightarrow_{\mathrm{Bx}} N[\theta_1]$, because $M \twoheadrightarrow_{\mathrm{Bx}} N$ by assumption, and the facts that $s^\sigma =_\beta M[\zeta]^{\oplus}$ and $t = N[\theta_1]^{\oplus}$, it follows that

$$s \twoheadrightarrow_\beta t. \tag{$\star$}$$

Now observe that, by definition, if $N_1\{x\backslash N_2\}$ is a subterm of N, then for all $X \in \mathrm{MV}(N_1)$, x does only occur bound in $\theta_2 \,;\, \zeta$, and thus

$$\ulcorner M[\theta_1]\downarrow_{\mathrm{x}}[(\theta_2 \,;\, \zeta)\downarrow_{\mathrm{x}}]\urcorner = \ulcorner M[\theta_1 \,;\, \theta_2 \,;\, \zeta]\downarrow_{\mathrm{x}}\urcorner.$$

where $\ulcorner K\urcorner$ denotes K with all closures removed, for arbitrary K. Therefore:

$$t^\sigma =_\beta N[\theta_1 \,;\, \theta_2 \,;\, \zeta]^{\oplus} =_\beta N[\zeta]^{\oplus}. \tag{$\star\star$}$$

We conclude the proof by noting that, since $s \twoheadrightarrow_\beta t$ by ($\star$), it follows by ($\star\star$) that:

$$s^\sigma =_\beta N[\zeta]^{\oplus},$$

as required. □

The lemma works, because the arguments of the free variables in the term and the leading abstractions in the substitution, take over the role of the explicit substitutions, as can be seen in the following example:

Example 4.3.12. Let the following pair of λx-term and grafting be given:

$$M := (\lambda x.(\lambda y.Z)\mathsf{b})\mathsf{a}$$
$$\zeta := [Z \mapsto \mathsf{f}(x, y)]$$

Now, according to Def. 4.3.10, $\langle s, \sigma\rangle$, where:

$$s := (\lambda x.(\lambda y.z(x, y))\mathsf{b})\mathsf{a}$$
$$\sigma := [z \mapsto \lambda xy.\mathsf{f}(x, y)]$$

is a λ-extension of $\langle M, \zeta\rangle$, with, $\theta_1 = [Z \mapsto Z(x, y)]$ and $\theta_2 = [\lambda xy.Z]$. We check both cases of Lemma 4.3.11:

(i) $s^\sigma = (\lambda x.(\lambda y.(\lambda xy.\mathsf{f}(x, y))(x, y))\mathsf{b})\mathsf{a} =_\beta (\lambda x.(\lambda y.\mathsf{f}(x, y))\mathsf{b})\mathsf{a} = M[\zeta]^\oplus$.

(ii) Let $N = Z\{y\backslash\mathsf{b}\}\{x\backslash\mathsf{a}\}$. Then $M \twoheadrightarrow_{\mathrm{x}} N$. Let $t = z(\mathsf{a}, \mathsf{b})$. Now:

$$t^\sigma =_\beta \mathsf{f}(\mathsf{a}, \mathsf{b}) = M[\zeta]^\oplus$$

Since $s =_\beta t$, this means that $s^\sigma =_\beta M[\zeta]^\oplus$, as required.
(Note that the $\cdot^\oplus$ operation also reduces to x-normal form.)

Translating (pre)patterns.

Now we define a translation between pairs of prepatterns and presubstitutions, on the one hand, and linear, passive λx-terms and graftings on the other. For this translation we use the same bijection between metavariables of λx-terms and variables of preterms as before. Because the notion of pattern is parametrized by a sequence of variables, the translation operations must be as well.

Definition 4.3.13. Let $\mathbf{P}^+_{\overline{x}}$, a mapping which maps pairs of linear, passive λx-terms containing no explicit substitutions and graftings to pairs of local $\overline{x}$-prepatterns and substitutions, be defined as follows:

$$\mathbf{P}^+_{\overline{x}}\langle Y, [Y \mapsto M]\rangle := \langle y(\overline{x}), [y \mapsto \lambda\overline{x}.M^\oplus]\rangle$$
$$\mathbf{P}^+_{\overline{x}}\langle y(\overline{P_n}), \zeta\rangle := \langle y(\overline{p_n}), \sigma\rangle \quad \text{if } \mathbf{P}^+_{\overline{x}}\langle P_i, \zeta{\restriction}P_i\rangle = \langle p_i, \sigma{\restriction}p_i\rangle$$
$$\mathbf{P}^+_{\overline{x}}\langle f(\overline{P_n}), \zeta\rangle := \langle f(\overline{p_n}), \sigma\rangle \quad \text{if } \mathbf{P}^+_{\overline{x}}\langle P_i, \zeta{\restriction}P_i\rangle = \langle p_i, \sigma{\restriction}p_i\rangle$$
$$\mathbf{P}^+_{\overline{x}}\langle \lambda y.P, \zeta\rangle := \langle \lambda y.p, \sigma\rangle \quad \text{if } \mathbf{P}^+_{\overline{x}y}\langle P, \zeta\rangle = \langle p, \zeta_\star\rangle$$

where $\zeta{\restriction}P$ denotes the restriction of grafting/substitution ζ to the metavariables/variables of λx-term/preterm P. With $\mathbf{P}^-_{\overline{x}}$ we denote the inverse of $\mathbf{P}^+_{\overline{x}}$.[2]

[2] Like $\cdot^{\ominus}_{\overline{x}}$, the $\mathbf{P}^-_{\overline{x}}$ operation is not actually a mapping, but this is no problem for the same reason. See footnote 1.

As usual, we generalize the above operations to (local pattern) substitutions and (linear, passive) graftings, in the obvious way. Note that the first element of the result of $\mathbf{P}^-_{\overline{x}}\langle\cdot,\cdot\rangle$ and $\mathbf{P}^+_{\overline{x}}\langle\cdot,\cdot\rangle$ does not depend on the second.

Lemma 4.3.14. *Let P be a linear, passive λx-term and ζ a grafting, both containing no explicit substitutions. Then $\mathbf{P}^+_{\overline{x}}\langle P,\zeta\rangle$ is a λ-extension of $\langle P,\zeta\rangle$, for an arbitrary list of variables $\overline{x}$.*

Proof. By the fact that *all* bound variables in scope are added as arguments to the free variables. □

Example 4.3.15. Consider the linear, local λx-terms:

$$P := \mathsf{f}(\lambda xy.\mathsf{g}(Z,x))$$
$$Q := \mathsf{map}(\lambda x.Z, \mathsf{nil})$$

and the grafting $\zeta = [Z \mapsto \mathsf{f}(x)]$. Then:

$$\mathbf{P}^+_{\varnothing}\langle P,\zeta\rangle = \langle \mathsf{f}(\lambda xy.\mathsf{g}(z(x,y),x)), [z \mapsto \lambda xy.\mathsf{f}(x)]\rangle$$
$$\mathbf{P}^+_{\varnothing}\langle Q,\zeta\rangle = \langle \mathsf{map}(\lambda x.Z(x), \mathsf{nil}), [z \mapsto \lambda x.\mathsf{f}(x)]\rangle$$

Both satisfy the requirements of being a λ-extension, as can be easily checked.

4.3.3 Proof of the Prefix Property

We have now set up the necessary tools to prove the Prefix Property. We repeat the statement of Theorem 4.3.1 for convenience:

Theorem 4.3.1 (Prefix Property)**.** *Let $\mathfrak{H}^\omega$ be the ω-labelling of a non-collapsing* HRS*, s a term, p a local $\overline{x}$-pattern and σ a substitution. If $s \to_{\mathfrak{H}^\omega} p^\sigma$, then there exist a local $\overline{x}$-pattern q and a substitution τ, such that $s = q^\tau$, $\mathsf{D}(p) \geq \mathsf{D}(q)$, and either:*

- *$q \to_{\mathfrak{H}^\omega} p^\upsilon$, for some substitution υ such that $\upsilon\,;\tau = \sigma$; or* *(trm)*
- *$q = p$ and $\tau \to_{\mathfrak{H}^\omega} \sigma$.* *(sub)*

Proof. In the course of this proof we use letters subscripted with a $\star$ (for example $s_\star, P_\star, \sigma_\star \ldots$) to denote preterms and presubstitutions. Letters without $\star$-subscript denote terms and substitutions on the term level, or λx-terms and graftings.

We prove the theorem by induction on the context of the step $s \to_{\mathfrak{H}^\omega} p^\sigma$. If p has a variable as head, then the (sub) case is trivially satisfied. If the step does not occur at the head, then the result follows easily from the induction hypothesis.

So, assume $s = l^\rho$ and $p^\sigma = r^\rho$, for some rule $\lambda\overline{x}.l \to \lambda\overline{x}.r \in R$ and substitution ρ. We 'cast' everything down to the preterm level, that is:

$$s_\star := s^\flat \quad l_\star := l^\flat \quad \rho_\star := \rho^\flat \quad r_\star := r^\flat \quad p_\star := p^\flat \quad \text{and} \quad \sigma_\star := \sigma^\flat.$$

Furthermore, let $\langle P, \zeta \rangle := \mathbf{P}^{-}_{\overline{x}} \langle p_\star, \sigma_\star \rangle$. Now, by definition, P is a linear, passive λx-term, and $P[\zeta]^{\oplus} =_\beta p^{\sigma_\star}_\star$.

Because p has no β-redexes, and the $\mathbf{P}^{-}_{\overline{x}} \langle \cdot, \cdot \rangle$ operation does not add closures, we know that $P[\zeta]$ is a Bx-normal form and

$$r^{\rho_\star}_\star =_\beta p^{\sigma_\star}_\star =_\beta P[\zeta]^{\oplus}.$$

Therefore, $r^{\rho_\star}_\star \twoheadrightarrow_\beta P[\zeta]^{\oplus}$, and from this and Lemma 4.3.9 it follows that

$$R[\mu] \twoheadrightarrow_{\mathrm{Bx}} P[\zeta],$$

where $R := (r_\star)^{\ominus}_{\overline{x}}$ and $\mu := (\rho_\star)^{\ominus}_{\overline{x}}$. Now, we can apply Theorem 4.3.7(ii) and obtain a linear, passive λx-term P', with $\mathcal{S}ym(P') = \mathcal{S}ym(P)$, and graftings η, κ such that

$$P'[\eta] = R[\mu], \quad P' \twoheadrightarrow_{\mathrm{Bx}} P[\kappa] \quad \text{and} \quad \kappa \,;\, \eta \twoheadrightarrow_{\mathrm{Bx}} \zeta$$

Because of the first equality above, we can apply Lemma 4.3.5 and obtain a linear, passive grafting μ' and graftings η', κ' with $\mathcal{S}ym(\eta') \subseteq \mathcal{S}ym(R)$ and $\mathcal{S}ym(\mu') \subseteq \mathcal{S}ym(P')$, such that

$$P'[\eta'] = R[\mu'], \quad \eta' \,;\, \kappa' = \eta \quad \text{and} \quad \mu' \,;\, \kappa' = \mu.$$

We now have the situation as in Fig. 4.2. Arrows with open arrow heads represent graftings, arrows with closed arrow heads represent Bx-reductions. The substitution required by the theorem is (the translation of) the grafting which is represented by the dashed arrow. We can however not directly compose κ and η'. However, by translation back to the term level, the Bx-reductions become equalities (modulo β-equivalence), and then the the translations κ and η' *can* be composed, yielding the desired substitution υ. The translation, using the techniques developed in the previous subsection, is, in principle, not hard, but it is a cumbersome operation.

Let $\langle \rho'_\star, \tau_\star \rangle := \mathbf{P}^{+}_{\varnothing} \langle \mu', \kappa' \rangle$. Then $\rho'_\star$ is a local pattern substitution, and by Lemma 4.3.14, $\langle \rho'_\star, \tau_\star \rangle$ is a λ-extension of $\langle \mu', \kappa' \rangle$. Let θ be the grafting such that $\rho'_\star = (\mu' \,;\, \theta)^{\oplus}$ (θ exists by definition of λ-extension), and let $\eta'' := \eta' \,;\, \theta$.

Because the (object) variables of $r_\star$ and $\tau_\star$ can be assumed, without loss of generality, to be disjoint, and because $\langle \rho'_\star, \tau_\star \rangle$ is a λ-extension of $\langle \mu', \kappa' \rangle$, it follows that $\langle r^{\rho'_\star}_\star, \tau_\star \rangle$ is a λ-extension of $\langle R[\mu'], \kappa' \rangle$ and thus of $\langle P'[\eta'], \kappa' \rangle$.

Next, let $\langle P'_\star, \eta''_\star \rangle := \mathbf{P}^{+}_{\overline{x}} \langle P', \eta'' \rangle$. By definition, there exists a grafting θ' such that $P'_\star = P'[\theta']^{\oplus}$, and because $P' \twoheadrightarrow_{\mathrm{Bx}} P[\kappa]$ it is, by definition of λ-extension, the case that

$$P'_\star =_\beta P[\kappa \,;\, \theta']^{\oplus}.$$

Now, let $\langle P_\star, \kappa_\star \rangle := \mathbf{P}^{+}_{\overline{x}} \langle P, \kappa \,;\, \theta' \rangle$. Since $P_\star$ does not depend on $\kappa \,;\, \theta'$, $P_\star = p_\star$. We know now that

$$P^{\kappa_\star}_\star =_\beta P[\kappa \,;\, \theta']^{\oplus} =_\beta P'[\theta']^{\oplus} =_\beta P'_\star.$$

Finally, we take: $q_\star := l^{\rho'_\star}_\star$, $\upsilon_\star := \kappa_\star \,;\, \eta''_\star$. Then:

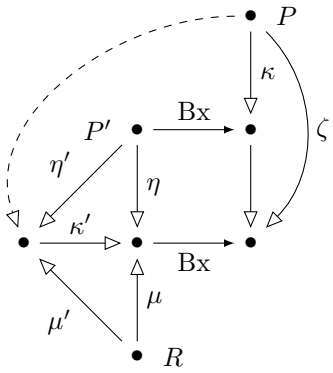


Figure 4.2: Proof of the Prefix Property. *This figure shows the constructed* λx*-terms and graftings in the proof of the Prefix Property and their relationships. Open arrows represent graftings, closed arrows represent* Bx*-reductions. The dashed open arrow represents the grafting which is transformed back into the required substitution* υ*.*

- $q_\star$ is a local $\overline{x}$-prepattern, because $l_\star$ is a local $\overline{x}$-prepattern and $\rho'_\star$ is a local pattern substitution;
- $r_\star^{\rho'_\star} =_\beta P'^{\eta''_\star}_\star =_\beta P_\star^{\kappa_\star;\eta''_\star} =_\beta p_\star^{\upsilon_\star}$;
- $s_\star = l_\star^{\rho_\star} =_\beta l_\star^{\rho'_\star;\tau_\star} =_\beta q_\star^{\tau_\star}$;
- $\upsilon_\star\,;\tau_\star =_\beta \kappa_\star\,;\eta''_\star\,;\tau_\star =_\beta \sigma_\star$;
- $\mathsf{D}(p_\star) = \mathsf{D}(P') = \mathsf{D}(P'[\eta']) = \mathsf{D}(R[\rho']) \geq \mathsf{D}(l_\circ[\rho']) = \mathsf{D}(q_\star)$, where the second equality holds because $\mathcal{S}ym(\eta') \subseteq \mathcal{S}ym(r)$, all labels in r are the same, p and r have at least one symbol in common because r is non-collapsing, and thus $\mathcal{S}ym(\eta') \subseteq \mathcal{S}ym(p)$.

Now we take $q := q_\star^\sharp$, $\upsilon := \upsilon_\star^\sharp$ and $\tau := \tau_\star$ to satisfy the (trm) case. □

4.4 Finite Family Developments for non-collapsing HRSs

In this section we apply the prefix property of the previous section to prove that all family developments of HRSs are finite: the Finite Family Developments property. We restrict our attention to non-collapsing HRSs first. In the next section, we will describe a way to generalize the result to collapsing HRSs as well.

Families are formalized by labelling all function symbols with natural numbers, as defined in Def. 4.2.1. We prove that the resulting system is terminating if we restrict the labels to some finite bound. The proof is inspired

by the proof by Van Oostrom [41]. The differences between this proof and the one by Van Oostrom are the following:

- We use a different method of labelling. Our labelling has the property that one step of the labelled HRS corresponds exactly to one step in the original. Also, our notion of labelling is an instance of the abstract notion of labelling put forth in [42, 43].
- In Van Oostrom's paper, the proof of Lemma 15 is omitted. Here, we give a proof of that lemma (it is Lemma 4.4.1), adapted for the different method of labelling, by reducing it to the Prefix Property.

Lemma 4.4.1. *Let $\mathfrak{H}^\omega$ be the labelling of a non-collapsing* HRS*, s be a term, p a local pattern, $\ell \in \mathbb{N}$ a label and τ and σ substitutions such that for any $x \in Dom(\sigma)$, $\sigma(x)$ has a function symbol labelled with ℓ as head. If $s^\sigma \twoheadrightarrow_{\mathfrak{H}^\omega} p^\tau$, then either:*

- $\mathsf{D}(p) \geq \ell$*; or* *(int)*
- $s \twoheadrightarrow_{\mathfrak{H}} p^\upsilon$*, for some υ such that $\upsilon\,;\sigma \twoheadrightarrow_{\mathfrak{H}^\omega} \tau$.* *(ext)*

Proof. By induction on the length of the reduction $s^\sigma \twoheadrightarrow_{\mathfrak{H}^\omega} p^\tau$. If the length is 0, there are two cases: if a substitution υ exists such that $s = p^\upsilon$, then the conditions of the (ext) case are trivially satisfied; otherwise, we show (int) by induction on p, using the assumption that $\sigma(x)$ has a function symbol labelled with ℓ as head, for all $x \in Dom(\sigma)$.

Otherwise, suppose $s^\sigma \twoheadrightarrow_{\mathfrak{H}^\omega} s' \rightarrow_{\mathfrak{H}^\omega} p^\tau$. By Theorem 4.3.1, there exist a local pattern q and substitution σ' such that $s' = q^{\sigma'}$, $\mathsf{D}(p) \geq \mathsf{D}(q)$ and either (trm) $q \twoheadrightarrow_{\mathfrak{H}^\omega} p^{\upsilon'}$ and $\upsilon'\,;\sigma' = \tau$; or (sub) $p = q$ and $\sigma' \twoheadrightarrow_{\mathfrak{H}^\omega} \tau$. Applying the induction hypothesis to $s^\sigma \twoheadrightarrow_{\mathfrak{H}^\omega} q^{\sigma'}$ yields that one of the following cases must apply:

- *(int)* $\mathsf{D}(q) \geq \ell$, but then $\mathsf{D}(p) \geq \ell$ by transitivity of $\geq$.
- *(ext)* $s \twoheadrightarrow_{\mathfrak{H}^\omega} q^\upsilon$ and $\upsilon;\sigma \twoheadrightarrow_{\mathfrak{H}^\omega} \sigma'$, for some substitution υ. We distinguish the following cases:
  - *(trm)* $s \twoheadrightarrow_{\mathfrak{H}^\omega} q^\upsilon \twoheadrightarrow_{\mathfrak{H}^\omega} p^{\upsilon';\upsilon}$ and $\upsilon'\,;\upsilon\,;\sigma \twoheadrightarrow_{\mathfrak{H}^\omega} \upsilon'\,;\sigma' = \tau$.
  - *(sub)* $s \twoheadrightarrow_{\mathfrak{H}^\omega} q^\upsilon = p^\upsilon$ and $\upsilon\,;\sigma \twoheadrightarrow_{\mathfrak{H}^\omega} \sigma' \twoheadrightarrow_{\mathfrak{H}^\omega} \tau$.

  In both cases the (ext) case of the lemma is satisfied. □

The above lemma allows us to prove the Finite Family Developments property for non-collapsing HRSs.

Theorem 4.4.2 (FFD)**.** *Let $\mathfrak{H}^\omega$ be the labelling of a non-collapsing* HRS*, and let $\mathcal{R} : s_1 \rightarrow_{\mathfrak{H}^\omega} s_2 \rightarrow_{\mathfrak{H}^\omega} \cdots$ be a $\mathfrak{H}^\omega$-reduction. $\mathcal{R}$ is relatively finite, if and only if there is a $\ell_{\max} \in \mathbb{N}$ such that $\mathsf{D}(s_i) \leq \ell_{\max}$ for all s_i.*

Proof. ($\Rightarrow$): An infinite but relatively finite reduction must have an infinite tail of empty steps. These empty steps are all the same, so removing them does not change which labels occur in the reduction and yields a finite reduction. Now, the result is trivial, because a finite reduction has a finite amount of steps and therefore a finite amount of labels, by definition.

($\Leftarrow$): We prove the theorem by showing that $\mathfrak{H}^\omega = \langle \Sigma^\omega, R^\omega \rangle$ is terminating if we restrict it to rules $l \to r \in R^\omega$ where $\mathsf{D}(r) \leq \ell_{\max}$. By the Right-hand Side Lemma (Lemma 2.4.12), it suffices to show termination of $(s^\ell)^\sigma$, for every right-hand side s, terminating substitution σ and label $\ell \in \mathbb{N}$. We prove a stronger result: we assume of s only that it is not (subst-subst)-collapsing.[3] We prove this by induction on $(\ell_{\max} - \ell)$.

Let $(s^\ell)^\sigma$ be a minimal term from which a relatively infinite reduction $\mathcal{R}$ exists. By Prop. 3.2.10 there exists an infinite proper reduction from $(s^\ell)^\sigma$, so we can assume without loss of generality that $\mathcal{R}$ is a proper reduction. Since $(s^\ell)^\sigma$ is minimal, we can assume that $\mathcal{R}$ contains at least one head step, and that $s = a(s_1, \ldots, s_n)$, so $s^\ell = a'((s_1^\ell)^\sigma, \ldots, (s_n^\ell)^\sigma)$. By minimality, the s_i^ℓ are terminating. We distinguish the following cases:

- If a is the function symbol f, then $a' = f^\ell$. Since the first head step strictly increases the label, termination follows from the induction hypothesis.

- If a is a variable, then it must be in the domain of σ (otherwise, a head step would not be possible, contradicting minimality). Suppose

  $$\sigma(a) = \lambda\overline{x}.b(t_1, \ldots, t_m)$$

  and thus $(s^\ell)^\sigma = b(t_1^{\sigma'}, \ldots, t_m^{\sigma'})$, where $\sigma' = [x_1 \mapsto s_1^\ell, \ldots, x_n \mapsto s_n^\ell]$. By a nested induction on the order $\to_{\mathfrak{H}^\omega}$ starting from $\sigma(a)$, which is well-founded by the assumption that σ is terminating, we prove that $(s')^{\sigma'}$ is terminating if $\sigma(a) \twoheadrightarrow_{\mathfrak{H}^\omega} s'$. The following cases can be distinguished:

  - Suppose $b = f^{\ell'}$. Then an infinite reduction from $(s^\ell)^\sigma$ looks like:

    $$f^{\ell'}(t_1^{\sigma'}, \ldots, t_m^{\sigma'}) \twoheadrightarrow_{\mathfrak{H}^\omega} f^{\ell'}(t_1, \ldots, t_m) = l^\tau \to_{\mathfrak{H}^\omega} (r^{\ell''})^\tau \twoheadrightarrow_{\mathfrak{H}^\omega} \cdots$$

    where $l \to r^{\ell'} \in R^\omega$. Since a is a variable and s is not (subst-subst)-collapsing, we know that the s_i^ℓ have function symbols labelled with ℓ as head, and we can apply Lemma 4.4.1 to l^τ and σ':

    * *(int):* $\mathsf{D}(l) \geq \ell$: It follows by the fact that, by construction, $\ell'' = \mathsf{D}(l) + 1$, that $\ell'' \geq \ell$. Thus, by the outermost induction hypothesis, $(r^{\ell''})^\tau$ is terminating, contradicting the assumption that $(s^\ell)^\sigma$ is not.

[3] We drop the (context-subst) condition of Def 2.4.10, because subterms of non (context-subst)-collapsing terms can be (constext-subst)-collapsing, meaning that an infinite reduction from a minimal counter example might not contain a head step.

∗ *(ext):* $a^\sigma \twoheadrightarrow_{\mathfrak{H}^\omega} l^\upsilon$ and $\upsilon\,;\sigma' \twoheadrightarrow_{\mathfrak{H}^\omega} \tau$: We know that

$$a^\sigma \twoheadrightarrow_{\mathfrak{H}^\omega} l^\upsilon \to_{\mathfrak{H}^\omega} (r^{\ell''})^\upsilon \quad \text{and} \quad (r^{\ell''})^{(\upsilon;\sigma')} \twoheadrightarrow_{\mathfrak{H}^\omega} (r^{\ell''})^\tau.$$

Since the left reduction consists of at least one step (the last one), $(r^{\ell''})^{(\upsilon;\sigma')}$ is terminating by the nested induction hypothesis, which yields termination of $(r^{\ell''})^\tau$ by the right reduction, contradicting the assumption that $(s^\ell)^\sigma$ is not terminating.

– Suppose b is a variable, it must be in the domain of σ. Suppose $b = x_i$, and $s_i = \lambda\overline{y}.c(u_1,\ldots,u_l)$. Then $(s^\ell)^\sigma = c(u_1,\ldots,u_l)^{(\sigma;\tau)}$ where $\tau = \bigcup_{1\le i\le m}[y_i \mapsto t_i^{\sigma'}]$. Now σ is terminating by assumption, and the $t_i^{\sigma'}$ are terminating by the nested induction hypothesis, so $\sigma\,;\tau$ is a terminating substitution. Since $c(u_1,\ldots,u_l)$ is a proper subterm of s, termination of $c(u_1,\ldots,u_l)^{(\sigma;\tau)}$ follows by minimality. □

4.5 Dealing with collapsing HRSs

In the previous sections we restricted our attention to non-collapsing HRSs. This is not without reason: both the Prefix Property and FFD do not hold for collapsing HRSs, as is witnessed by the following two counterexamples:

Example 4.5.1 (Prefix Property). Consider the collapsing HRS $\mathfrak{M}u$:

$$\mathsf{mu}(\lambda x.z(x)) \to z(\mathsf{mu}(\lambda x.z(x))$$

and the following $\mathfrak{M}u^\omega$-step:

$$\mathsf{mu}^3(\lambda x.\mathsf{f}^2(x)) \to_{\mathfrak{M}u^\omega} \mathsf{f}^2(\mathsf{mu}^4(\lambda x.\mathsf{f}^2(x)))$$

It is easy to check that the prefix $p = \mathsf{f}^2(u)$ of the target of the step has no ancestor q that satisfies the requirements of the Prefix Property (Theorem 4.3.1).

Example 4.5.2 (FFD). Consider the collapsing HRS $\mathfrak{L}am$:

$$\mathsf{app}(\mathsf{lam}(\lambda x.z(x), y)) \to z(y)$$

Then one $\mathfrak{L}am^\omega$-step is the following:

$$\begin{aligned}&\mathsf{app}^1(\mathsf{lam}^1(\lambda x.\mathsf{app}^1(x,x)), \mathsf{lam}^1(\lambda x.\mathsf{app}^1(x,x)))\\ &\to_{\mathfrak{L}am^\omega} \mathsf{app}^1(\mathsf{lam}^1(\lambda x.\mathsf{app}^1(x,x)), \mathsf{lam}^1(\lambda x.\mathsf{app}^1(x,x)))\end{aligned}$$

So we see that $\mathfrak{L}am^\omega$ has a one-step cycle, and thus an infinite reduction with bounded labels.

The problem in both cases is that, because of applying a collapsing rule, a function symbol can be directly connected to a previously unconnected function symbol from the context or substitution, or to the root of the term, without

the rule leaving any trace in between, in the form of a (labelled) function symbol. This can be remedied by including 'empty' function symbols, named ϵ_α, for each base type α, in the right-hand sides of rules, and 'saturating' the left-hand sides of rules with those empty function symbols. The same approach is taken for the first-order case in [42, 43].

Definition 4.5.3 (ϵ-lifting)**.** Let $\mathfrak{H} = \langle \Sigma, R \rangle$ be an HRS.

(i) The ϵ-lifting Σ^ϵ of the signature Σ, consists of all function symbols of Σ, and, for each base type α, a function symbol $\epsilon_\alpha : \alpha \to \alpha$.

(ii) The ϵ-lifting of a term s of type α, written s^ϵ, is defined as follows:

$$\begin{aligned} x(s_1, \ldots, s_n)^\epsilon &:= \epsilon_\alpha(x(s_1^\epsilon, \ldots, s_n^\epsilon))) \\ f(s_1, \ldots, s_n)^\epsilon &:= f(s_1^\epsilon, \ldots, s_n^\epsilon) \\ (\lambda x.s_0)^\epsilon &:= \lambda x.s_0^\epsilon \end{aligned}$$

(iii) The set of ϵ-saturations of a pattern p, denoted by $\mathrm{Sat}^\epsilon(s)$, is defined as $\mathrm{Sat}^\epsilon(p) = \mathrm{Sat}^\epsilon_{\mathrm{out}}(p)$, where:

$$\begin{aligned} \mathrm{Sat}^\epsilon_{\mathrm{out}}(x(\overline{y_n})) &:= \{x(\overline{y_n})\} \\ \mathrm{Sat}^\epsilon_{\mathrm{out}}(f(\overline{p_n})) &:= \{f(\overline{q_n}) \mid q_i \in \mathrm{Sat}^\epsilon_{\mathrm{in}}(p_i)\} \\ \mathrm{Sat}^\epsilon_{\mathrm{out}}(\lambda x.p_0) &:= \{\lambda x.q_0 \mid q_0 \in \mathrm{Sat}^\epsilon_{\mathrm{in}}(p_0)\} \\ \mathrm{Sat}^\epsilon_{\mathrm{in}}(x(\overline{y_n})) &:= \{x(\overline{y_n})\} \\ \mathrm{Sat}^\epsilon_{\mathrm{in}}(f(\overline{p_n})) &:= \{\epsilon^m_\alpha(f(\overline{q_n})) \mid m \in \mathbb{N}, q_i \in \mathrm{Sat}^\epsilon_{\mathrm{in}}(p_i)\} \\ \mathrm{Sat}^\epsilon_{\mathrm{in}}(\lambda x.p_0) &:= \{\lambda x.q_0 \mid q_0 \in \mathrm{Sat}^\epsilon_{\mathrm{in}}(p_0)\} \end{aligned}$$

where α is the type of p, and $\epsilon^m_\alpha(p)$ is inductively defined by $\epsilon^0_\alpha(p) = p$, $\epsilon^{m+1}_\alpha(p) = \epsilon_\alpha(\epsilon^m_\alpha(p))$.

(iv) The ϵ-lifting of an HRS $\mathfrak{H} = \langle \Sigma, R \rangle$ is defined as $\mathfrak{H}^\epsilon = \langle \Sigma^\epsilon, R^\epsilon \rangle$, where

$$R^\epsilon := \bigcup_{\rho : l \to r \in R} \{\rho_{l'} : l' \to r^\epsilon \mid l' \in \mathrm{Sat}^\epsilon(l)\}.$$

(v) The projection operation $[\![\cdot]\!]_\epsilon$ is the mapping from Σ^ϵ-terms to Σ-terms and $\mathfrak{H}^\epsilon$-proof terms to $\mathfrak{H}$-proof terms, which removes all ϵ_α-symbols, that is, the operation is given by:

$$\begin{aligned} [\![\epsilon_\alpha(\varphi)]\!]_\epsilon &:= [\![\varphi]\!]_\epsilon \\ [\![\lambda x.\varphi]\!]_\epsilon &:= \lambda x.[\![\varphi]\!]_\epsilon \\ [\![x(\varphi_1, \ldots, \varphi_n)]\!]_\epsilon &:= x([\![\varphi_1]\!]_\epsilon, \ldots, [\![\varphi_n]\!]_\epsilon) \\ [\![f(\varphi_1, \ldots, \varphi_n)]\!]_\epsilon &:= f([\![\varphi_1]\!]_\epsilon, \ldots, [\![\varphi_n]\!]_\epsilon) \\ [\![\rho_l(\varphi_1, \ldots, \varphi_n]\!]_\epsilon &:= \rho([\![\varphi_1]\!]_\epsilon, \ldots, [\![\varphi_n]\!]_\epsilon) \end{aligned}$$

(vi) For each Σ^ϵ-term s, we define the relation $\Updownarrow_s$ between $\mathcal{P}os([\![s]\!]_\epsilon)$ and $\mathcal{P}os(s)$ as follows:

- if $s = x(\overline{s})$, then $\varepsilon \Updownarrow_s \varepsilon$ and $ip \Updownarrow_s iq$ if $p \Updownarrow_{s_i} q$;
- if $s = f(\overline{s})$, then $\varepsilon \Updownarrow_s \varepsilon$ and $ip \Updownarrow_s iq$ if $p \Updownarrow_{s_i} q$;
- if $s = \lambda x.s'$, then $\varepsilon \Updownarrow_s \varepsilon$ and $1p \Updownarrow_s 1q$ if $p \Updownarrow_{s'} q$;
- if $s = \epsilon(s')$, then $p \Updownarrow_s 1q$ if $p \Updownarrow_{s'} q$.

Observe that, above, there is a distinction between ε, the empty sequence (here, in particular, the root position), and ϵ, the empty function symbol. Note that the mapping $\Updownarrow_s$ is a function because it is total and injective. The ϵ-lifting produces non-collapsing HRSs and terms, steps and reduction in such reductions, as follows from the following proposition:

Proposition 4.5.4. *For any* HRS $\mathfrak{H}$*, its ϵ-lifting $\mathfrak{H}^\epsilon$ is a non-collapsing* HRS*.*

Proof. By definition, all right-hand sides of $\mathfrak{H}^\epsilon$ are of the form s^ϵ, for some s. It is easy to see, that for all collapsing terms s, s^ϵ is non-collapsing. □

The ϵ-liftings of the two counter examples introduced earlier in this section do not have the same problems as their originals:

Example 4.5.5. The ϵ-lifting of $\mathfrak{Mu}$ is the following (types of ϵ's omitted):

$$\mathsf{mu}(\lambda x.z(x)) \to \epsilon(z(\epsilon(\mathsf{mu}(\lambda x.\epsilon(z(\epsilon(x))))))).$$

A $(\mathfrak{Mu}^\epsilon)^\omega$ step corresponding to the step of Ex. 4.5.1 is:

$$\mathsf{mu}^3(\lambda x.\mathsf{f}^2(x)) \to_{(\mathfrak{Mu}^\epsilon)^\omega} \epsilon^4(\mathsf{f}^2(\epsilon^4(\mathsf{mu}^4(\lambda x.\epsilon(\mathsf{f}^2(\epsilon(x))))))).$$

Take the corresponding prefix $p = \epsilon^4(\mathsf{f}^2(y))$. Now, the Prefix Property is satisfied with $q = \mathsf{mu}^3(\lambda x.\mathsf{f}^2(x))$, $\tau = \varnothing$ and $\upsilon = [z \mapsto \epsilon^4(\mathsf{mu}^4(\lambda x.\epsilon(\mathsf{f}^2(\epsilon(x))))))]$.

Example 4.5.6. The ϵ-lifting of $\mathfrak{Lam}$ consists of (among others) the following rules:

$$\begin{aligned}
\mathsf{app}(\mathsf{lam}(\lambda x.z(x), y)) &\to \epsilon(z(\epsilon(y))) \\
\mathsf{app}(\epsilon(\mathsf{lam}(\lambda x.z(x))), y) &\to \epsilon(z(\epsilon(y))) \\
\mathsf{app}(\epsilon(\epsilon(\mathsf{lam}(\lambda x.z(x)))), y) &\to \epsilon(z(\epsilon(y)))
\end{aligned}$$

Then a $(\mathfrak{Lam}^\epsilon)^\omega$-step corresponding to the step of Ex. 4.5.2 is the following:

$$\begin{aligned}
&\mathsf{app}^1(\mathsf{lam}^1(\lambda x.\mathsf{app}^1(x,x)), \mathsf{lam}^1(\lambda x.\mathsf{app}^1(x,x))) \\
&\to_{(\mathfrak{Lam}^\epsilon)^\omega} \quad \epsilon^2(\mathsf{app}^1(\epsilon^2(\mathsf{lam}^1(\lambda x.\mathsf{app}^1(x,x))), \epsilon^2(\mathsf{lam}^1(\lambda x.\mathsf{app}^1(x,x)))))).
\end{aligned}$$

Now, all redex patterns have a maximum label of 2, instead of 1.

Lemma 4.5.7. *Let $\mathfrak{H} = \langle \Sigma, R \rangle$ be an* HRS*.*

(i) Let s be a Σ-term and s' a Σ^ϵ-term with $[\![s']\!]_\epsilon = s$. For each $\mathfrak{H}$-step φ with source s, there is a unique $\mathfrak{H}^\epsilon$-step φ' with source s' such that $[\![\psi]\!]_\epsilon = \varphi$.

(ii) If $\varphi : s \multimap\!\!\rightarrow_{\mathfrak{H}^\epsilon} t$, *then* $[\![\varphi]\!]_\omega : [\![s]\!]_\epsilon \multimap\!\!\rightarrow_{\mathfrak{H}} [\![t]\!]_\epsilon$.

Proof. (i) By induction on the inference of the step $\varphi : s \multimap\!\!\rightarrow_{\mathfrak{H}} t$. The interesting case is if $\varphi = \rho(\varphi_1, \ldots, \varphi_n)$ for some rule $\rho : l \to r$. Then it is the case that $s = l(s_1, \ldots, s_n)$ and $s' = \epsilon^n(l'(s'_1, \ldots, s'_n))$. By the induction hypothesis, there are unique φ'_i with source s'_i such that $[\![\varphi']\!]_\epsilon = \varphi$. We take $\varphi' := \epsilon^n(\rho_{l'}(\varphi_1, \ldots, \varphi_n))$ to satisfy the requirements of the lemma.

(ii) By induction on the inference of the step $\varphi : s \multimap\!\!\rightarrow_{\mathfrak{H}^\epsilon} t$. The interesting case is the case that $\varphi = \rho_l(\varphi_1, \ldots, \varphi_n)$, for some $\rho_l : l \to r$ and $\varphi_i : s_i \multimap\!\!\rightarrow t_i$. By the induction hypothesis $[\![\varphi_i]\!]_\epsilon : [\![s_i]\!]_\epsilon \to [\![t_i]\!]_\epsilon$. From this and the fact that $[\![\rho_l]\!]_\epsilon : [\![l]\!]_\epsilon \to [\![r]\!]_\epsilon$ we conclude that

$$[\![s]\!]_\epsilon = [\![l(s_1, \ldots, s_n)]\!]_\epsilon \multimap\!\!\rightarrow_{\mathfrak{H}} [\![r(t_1, \ldots, t_n)]\!]_\epsilon = [\![t]\!]_\epsilon. \qquad \square$$

Just as in Section 4.2, the results of Lemma 4.5.7 can be easily generalized to reductions. Now, we can use the ϵ-lifting to prove FFD for arbitrary HRSs:

Theorem 4.5.8 (FFD)**.** *Let* $(\mathfrak{H}^\epsilon)^\omega$ *be the* $\epsilon\omega$*-labelling of an* HRS*, and let* $\mathcal{R} : s_1 \to_{(\mathfrak{H}^\epsilon)^\omega} s_2 \to_{(\mathfrak{H}^\epsilon)^\omega} \cdots$ *be a* $(\mathfrak{H}^\epsilon)^\omega$*-reduction.* $\mathcal{R}$ *is relatively finite, if and only if there is a* $\ell_{\max} \in \mathbb{N}$ *such that* $\mathsf{D}(s_i) \leq \ell_{\max}$ *for all* s_i*.*

Proof. By Prop. 4.5.4 and Theorem 4.4.2. $\square$

Again, we want to canonically map reductions of a potentially collapsing HRS to reductions of a non-collapsing HRS: we take the ϵ-lifting starting in the source of the reduction itself. Formally:

Definition 4.5.9. Let $\mathfrak{H}$ be an HRS, and $\mathcal{R} : s \multimap\!\!\twoheadrightarrow \cdots$ a reduction.

(i) The canonical ϵ-lifting of $\mathcal{R}$ is the $\mathfrak{H}^\epsilon$-reduction $\mathcal{S}$ such that $[\![\mathcal{S}]\!]_\epsilon = \mathcal{R}$ and $\mathrm{src}(\mathcal{S}) = s$.

(ii) The canonical $\epsilon\omega$-labelling of $\mathcal{R}$ is the $(\mathfrak{H}^\epsilon)^\omega$-reduction $\mathcal{S}$ such that

$$[\![[\![\mathcal{S}]\!]_\epsilon]\!]_\omega = \mathcal{R} \quad \text{and} \quad \mathrm{src}(\mathcal{S}) = s^0.$$

Def. 4.5.9 is well-defined by Lemma 4.2.4 and Lemma 4.5.7. In the following, we will often refer to the creation depth or family of a function symbol, term or step in a reduction of an unlabeled and possibly collapsing HRS. In such cases, we actually mean the creation depth or family of the corresponding function symbol (given by the map $\Updownarrow_s$), term or step in the canonical $\epsilon\omega$-reduction.

4.6 Applications

The result of this chapter will be applied in several places in the rest of this thesis. In this section we highlight two applications that have no direct relevance to other parts of this work: finite developments and a proof of termination of the simply typed λ-calculus.

4.6.1 Finite Developments

Finiteness of Developments is a well-known and important property of the λ-calculus, Combinatory Logic and first-order TRSs, and has proven to be a useful technique for proving, for example, confluence. The property says that any reduction which only contracts (descendants of) redexes that occur in the source term, is finite.

The Finite Developments property is usually proved by a technique called *marking* or *underlining* (see for example Prop. 4.4.5 and Theorem 4.5.4 of [53]). In this section we define a similar technique for HRSs and show that finite developments are essentially equivalent to reductions with a maximum creation depth of 1, and therefore finite by a simple application of the Finite Family Developments theorem. We restrict our attention to non-collapsing HRSs, again. Collapsing HRSs may be lifted to non-collapsing HRSs by using the technique developed in Sect. 4.5.

Definition 4.6.1 (Underlining)**.** Let $\mathfrak{H} = \langle \Sigma, R \rangle$ be an HRS. The *underlining* $\underline{\mathfrak{H}} = \langle \underline{\Sigma}, \underline{R} \rangle$, is defined as follows:

(i) $\underline{\Sigma} := \Sigma \cup \{\underline{f} : \alpha \mid f : \alpha \in \Sigma\}$

(ii) $\underline{R} := \{\underline{l} \to r \mid l \to r \in R\}$
where $\underline{l}$ is obtained by replacing all function symbols f in l by their underlined version $\underline{f}$.

Theorem 4.6.2 (Finite Developments)**.** *The underlining* $\underline{\mathfrak{H}}$ *of every* HRS $\mathfrak{H}$ *is terminating.*

Proof. By the observation that each $\underline{\mathfrak{H}}$-reduction maps to a $\mathfrak{H}^\omega$-reduction if 'normal' function symbols are changed to function symbols with creation depth 0, and underlined function symbols are changed to function symbols with creation depth 1. By Theorem 4.5.8, this $\mathfrak{H}^\omega$-reduction is finite because every creation depth is smaller or equal than 1, and therefore also the original reduction is finite. □

4.6.2 Termination of the simply typed λ-calculus

The most well-known proof of termination of the simply typed λ-calculus is the proof using strong computability due to Tait [52]. Here we present a termination proof of an HRS which encodes the simply typed λ-calculus, using the Finite Family Developments result of the present paper.[4]

[4] It is noted in [25, p. 31] that termination of simply typed λ-calculus follows from termination of the Hyland–Wadsworth-labelling, a variation of which we use to formalize FFD.

The HRS we consider has an infinite number of rules. Let α, β range over (codes of) types, and let the following signature be given:

$$\begin{aligned} \mathsf{T}_\alpha &: \mathit{term} \to \mathit{term} \\ \mathsf{app} &: \mathit{term} \to \mathit{term} \to \mathit{term} \\ \mathsf{lam} &: (\mathit{term} \to \mathit{term}) \to \mathit{term} \end{aligned}$$

The simply typed λ-calculus can be encoded as the following HRS, $\mathfrak{Lam}^{\to}$:

$$\mathsf{app}(\mathsf{T}^n_{\alpha\to\beta}(\mathsf{lam}(\lambda x.Z(x))), Y) \to \mathsf{T}_\beta(Z(\mathsf{T}_\alpha(Y)))$$

for every $n > 0$, where $f^n(s)$ is defined (here) as: $f^0(s) = s$ and $f^{n+1}(s) = f(f^n(s))$. Termination of $\mathfrak{Lam}^{\to}$ cannot be proved by any higher-order termination technique that the author is aware of. However, it does follow from Finite Family Developments:

Proposition 4.6.3. *$\mathfrak{Lam}^{\to}$ is terminating.*

Proof. Note that $\mathfrak{Lam}^{\to}$ is non-collapsing, and consider the ω-labelling of $\mathfrak{Lam}^{\to}$, which consists of rules of the form:

$$\mathsf{app}^\ell(((\mathsf{T}^k_{\alpha\to\beta})^n(\mathsf{lam}^j(\lambda x.Z(x))), Y) \to \mathsf{T}^p_\beta(Z(\mathsf{T}^p_\alpha(Y)))$$

where ℓ, j are labels representing the creation depth of the symbols, $\overline{k}$ is a sequence of n such labels and $p = \max(\ell, \overline{k}, j)$. The symbols of the form T_α will be called *type symbols*.

Let the *height* of a type be defined as follows: $ht(a) = 1$ (where a is a base type); $ht(\alpha \to \beta) = \max(ht(\alpha), ht(\beta)) + 1$. Now we can define the *value* of a type symbol as $\mathit{Val}(\mathsf{T}^\ell_\alpha) = \ell + ht(\alpha)$ and the value of a term as $\mathit{Val}(s) = \max\{\mathit{Val}(f) \mid f \in \mathcal{S}ym(s)\}$. Now, a simple induction on the context of the step reveals that, if $s \to_{\mathfrak{Lam}^{\to}} t$, then $\mathit{Val}(s) \geq \mathit{Val}(t)$.

Let $\mathfrak{R} : s_0 \to s_1 \to s_2 \to \cdots$ be an arbitrary $\mathfrak{Lam}^{\to}$-reduction. Because $\mathit{Val}(s_i) \geq \mathit{Val}(s_j)$ if $i < j$, the value of every term in $\mathcal{R}$ is less or equal to $\mathit{Val}(s_0)$, and thus every symbol occurring in the reduction has a creation depth less than or equal to $\mathit{Val}(s_0)$. Therefore, by Finite Family Developments (Theorem 4.4.2), $\mathfrak{R}$ is finite. □

We conclude this subsection by showing that the HRS introduced above encodes the simply typed λ-calculus in the following sense: every reduction in the simply typed λ-calculus can be lifted to a $\mathfrak{Lam}^{\to}$-reduction. For this, we use the following map h, which maps each simply typed λ-term to a set of possible $\mathfrak{Lam}^{\to}$-encodings of it:

$$\begin{aligned} \mathrm{h}(x_\alpha) &:= \{\mathsf{T}^n_\alpha(x) \mid n > 0\} \\ \mathrm{h}(f_\alpha) &:= \{\mathsf{T}^n_\alpha(f) \mid n > 0\} \\ \mathrm{h}(M_{\alpha\to\beta}N_\alpha) &:= \{\mathsf{T}^n_\beta(\mathsf{app}(s,t)) \mid s \in \mathrm{h}(M), t \in \mathrm{h}(N), n > 0\} \\ \mathrm{h}(\lambda x_\alpha.M_\beta) &:= \{\mathsf{T}^n_{\alpha\to\beta}(\mathsf{lam}(\lambda x.s)) \mid s \in \mathrm{h}(M), n > 0\} \end{aligned}$$

It follows directly from the definition that $\mathrm{h}(M) \neq \varnothing$, for all simply typed λ-terms M. Additionally, we show that the map h commutes over substitution:

Lemma 4.6.4. $\mathrm{h}(M[x \mapsto N]) = \mathrm{h}(M)[x \mapsto \mathrm{h}(N)]$.

Proof. By induction on M. □

We have the following correspondence between the simply typed λ-calculus and $\mathfrak{Lam}^{\to}$:

Lemma 4.6.5. *If $M \to_\beta N$ and $s \in \mathrm{h}(M)$, then there is a $t \in \mathrm{h}(N)$ such that $s \to_{\mathfrak{Lam}^{\to}} t$.*

Proof. By induction on the derivation of $M \to_\beta N$. The interesting case is if the β-step occurs at the head, that is:

$$M = (\lambda x_\alpha.M_1)M_2 \quad \text{and} \quad N = M_1[x \mapsto M_2]$$

By definition, this means that $s = \mathsf{T}^n_\beta(\mathsf{app}(\mathsf{T}^m_{\alpha\to\beta}(\mathsf{lam}(\lambda x.t_1)), t_2))$, where $t_i \in \mathrm{h}(M_i)$, for $i \in \{1, 2\}$, and $m, n > 0$. Now:

$$s \to \mathsf{T}^{n+1}_\beta(t_1[x \mapsto \mathsf{T}_\alpha(t_2)]) \in \mathrm{h}(N)$$

where the last inclusion follows from Lemma 4.6.4 and the facts that $\mathsf{T}^{n+1}_\beta \in \mathrm{h}(M_1)$ because $t_1 \in \mathrm{h}(M_1)$, and $\mathsf{T}_\alpha(t_2) \in \mathrm{h}(M_2)$ because $t_2 \in \mathrm{h}(M_2)$. □

Theorem 4.6.6. *The simply typed λ-calculus is terminating.*

Proof. By Lemma 4.6.5, the existence of an infinite reduction in the simply typed λ-calculus implies the existence of an infinite $\mathfrak{Lam}^{\to}$-reduction, which is impossible by Prop. 4.6.3. □

At first glance, the proof above looks like cheating: we already use the simply typed λ-calculus as the substitution calculus of higher-order rewriting. However, in the proof of FFD, we only require *normalization*: we assume that from each preterm a β-reduction to normal form exists, but not that every β-reduction is finite.

4.7 Related work

As mentioned before, the result of this chapter builds on the work of Van Oostrom [41]. In first-order TRSs, Finiteness of Family Developments follows from a simple argument involving a recursive path ordering. Interestingly, to my knowledge, all higher-order variants of the recursive path ordering (for example [21, 45]) are too weak to prove higher-order FFD.

The property of FFD is also similar to results in other rewriting paradigms: match bounded string rewrite systems and dependency pairs in first-order TRSs.

4.7.1 Match-bounds

Geser, Hofbauer & Waldmann introduced the class of *match-bounded* String Rewrite Systems (SRSs) [13]. Proving that a SRS is match-bounded, is a very successful technique for (automatically) proving termination of SRSs. Just like in our approach, symbols are labelled with natural numbers, called *match heights*, which are increased during the rewriting process. The SRS is terminating if there is a bound on the match heights of the reductions of the SRS. In this respect the match-bound criterion is a stronger criterion than the FFD property: FFD requires that every reduction has a bound, while the match-bounds approach requires there to be a global bound which holds for all reductions.

There is a second notable difference between match bounds and FFD: where in our approach symbols of the right-hand side are labelled with a creation depth higher than the *maximum* creation depth of the left-hand side, in the match-bounds approach the symbols of the right-hand side are labelled with a match height higher than the *minimum* match height of the left-hand side.

Note, that in HRSs, and even first-order TRSs, it is not possible to obtain a termination result when labelling the right-hand side with the minimum creation depth of the left-hand side plus one; it is essential that the maximum creation depth is used. Consider, as a counter-example the one-rule TRS $\mathsf{f}(x, \mathsf{a}) \to \mathsf{f}(x, x)$. Then:

$$\mathsf{f}^0(\mathsf{a}^0, \mathsf{a}^0) \to \mathsf{f}^1(\mathsf{a}^0, \mathsf{a}^0) \to \mathsf{f}^1(\mathsf{a}^0, \mathsf{a}^0) \to \cdots$$

is an infinite reduction in the labelled TRS.

The problem is seemingly caused by the duplication behavior of the TRS. In fact, the technique has been generalized to TRSs in [14], and it is shown there that, for linear TRSs, it is sufficient to label the right-hand sides with the minimum label of the corresponding left-hand side plus one, while for non-linear TRSs the maximum label plus one is required.

4.7.2 Dependency pairs

Arts & Giesl introduced the Dependency Pair approach [1], which turned out to be a very successful technique for (automatically) proving termination of first-order TRSs. In the Dependency Pair approach termination is proved by showing that infinite *dependency chains* cannot exist, which is, in many cases, easier than showing termination of the TRS directly.

Although the similarity is harder to spot than in the case of match-bounds, the Dependency Pairs approach and the Finite Family Development approach follow from the same basic principle: that in every finite reduction there is a bound on the creation depth of the function symbols in the term. In the (original, first-order) Dependency Pair approach only the defined symbols (the head symbols of the left-hand sides of the rules) need to be considered, which is why dependency chains are often compared to chains of function calls. The existence of infinite dependency chains is essentially the Dependency

Pair equivalent of the unboundedness of the creation depths of the function symbols.

In order to generalize Dependency Pairs to the higher-order case, some method must be devised to consider other symbols than the defined symbols as well. Many attempts have been made for HRSs and other higher-order rewriting paradigms (for example [47, 46, 15, 6]) but most are rather weak in comparison to first-order dependency pairs, even if restricted to 'first-order' HRSs, that is HRSs in which all variables are of base type.

It is my opinion that the similarities between Finite Family Developments and the techniques mentioned above warrant a closer investigation into their relationships. Perhaps a termination technique can be found which is powerful for both first- and higher-order rewriting paradigms.

4.8 Discussion

In this chapter we gave a proof of the Finite Family Developments property of HRSs. Interestingly, the proof of Finite Family Developments is much more involved in the higher-order case than in the first-order case. In the first-order case it follows, even for collapsing TRSs, by a simple application of the Recursive Path Ordering. Higher-order Recursive Path Orderings do exist, but are, to my knowledge, presently not strong enough to prove higher-order Finite Family Developments.

Finiteness of Family Developments is a very useful technical property: in Chapter 5 it helps to show termination of two higher-order standardization procedures, by the following observation: if some operation on (finite) reductions preserves the bound on the labels of the function symbols, then by König's Lemma there is a bound on the length of the reductions obtained by the operation (see for example Lemma 5.4.8).

Finite Family Developments can also be useful to prove termination of HRSs, as is demonstrated in Sect. 4.6.2, and since many rewriting paradigms can be encoded as HRSs, the result of this chapter may be useful to other forms of rewriting as well.

Five

Standardization

5.1 Introduction

Standardization is the property that for each reduction, there exists an equivalent reduction in which the redexes are contracted in a pre-defined, standard order (usually the outermost-innermost, left-right order, also called the standard order). Such a reduction is called a *standard reduction*. Standardization is useful in various cases:

- Standardization often makes it possible to simplify proofs considerably: because each reduction has an equivalent standard reduction, it suffices in many cases to consider only standard reductions. For example, if we want to find out that some term t can be reached from s, we now only have to find a standard reduction from s to t, which considerably narrows the search space.
- Standardization is helpful to study the semantics of (functional) programming languages. In lazy functional programming languages such as Haskell, for example, program traces roughly correspond to standard reductions in the sense of this chapter.
- If it can be shown that each (finite) reduction has a *unique* equivalent standard reduction, which is the case in this chapter, standard reductions can be considered as the unique representatives of (permutation) equivalent reductions, and standardization becomes a method to decide equivalence of reductions. It is this useful fact that motivated the study of standardization in this dissertation.

Here, we consider a reduction to be standard if the redexes are contracted from left to right.

In his dissertation, Klop [25] identified two methods to 'calculate' an equivalent reduction for a given reduction. One is a deterministic procedure which finds the step contracting the left-most redex and moves this step to the beginning of the reduction, which we call *Selection Standardization* here. The

other, called *Inversion Standardization* here, is a non-deterministic procedure which permutes adjacent steps which are in the wrong order (modulo an equivalence which permutes parallel steps). Here, we translate both standardization methods to HRSs. We also show that each reduction has a *unique* equivalent standard reduction, making our notion of standard suitable for formalizing equivalence of reductions.

Standardization theorems have been proved for, among others, the λ-calculus [10, 44, 25], first-order TRSs [19, 53] and Combinatory Reduction Systems [25, 55]. Also, standardization has been studied from an abstract point of view [16, 33]. We will spend some time comparing our approach to other approaches in Sect. 5.8.

We restrict our attention to local (that is linear and fully extended) HRSs. This is because only these kinds of HRSs are compatible with our notion of standardness and permutation equivalence. In Sect. 5.6 we show this by giving two counter examples.

5.2 Standard reductions

We will call a reduction "standard" if the redexes are contracted in strictly left-to-right order. We can formally define the notion of standardness as follows:

Definition 5.2.1.

(i) Let $\varphi : s \to t$ be a proper step contracting redex pattern R at position p, and $\mathcal{R} = \psi_0, \psi_1, \ldots$ a proper reduction starting at t. $\mathcal{R}$ is *standard* for φ, if there is no index k such that:

- ψ_k contracts a redex pattern S such that $S <_{\text{lex}} R$;
- for all $i < k$, ψ_i contracts a redex pattern T such that $R \leq_{\text{lex}} T$; and
- $p \notin S$.

(ii) Let $\varphi : s \to t$ be a proper step contracting redex pattern R at position p, and $\mathcal{R}$ a reduction containing multisteps. $\mathcal{R}$ is standard for φ, if the canonical development of $\mathcal{R}$ is standard for φ.

(iii) A reduction $\mathcal{R} = \varphi_1, \varphi_2, \ldots$ is *standard* if it is proper and, for each i, the reduction starting at φ_{i+1} is standard for φ_i.

Item (i) of the definition can be informally stated as follows: a proper reduction $\mathcal{R}$ is standard for a proper step φ contracting a redex at position p, if for the first step ψ_k which contracts a redex to the left of the redex of φ, it holds that $p \in \mathcal{RP}os(\psi)$. The intuition behind this is the following. We want that the redexes in a reduction are contracted from left to right. A step ψ_k may only contract a redex to the left of the redex contracted by φ, if φ is involved in creating the redex.

Example 5.2.2. Let the following HRS be given:

$$\begin{aligned} \rho: &\quad \mathsf{let}(\lambda x.z(x), y) \to z(y) \\ \theta: &\quad \mathsf{f}(x) \to \mathsf{g}(x) \\ \eta: &\quad \mathsf{a} \to \mathsf{b} \end{aligned}$$

The reduction

$$\mathsf{let}(\lambda x.\mathsf{f}(x), \eta)\,;\,\mathsf{let}(\lambda x.\theta(x), \mathsf{b})\,;\,\rho(\lambda x.\mathsf{g}(x), \mathsf{b})$$

is not standard because, because the second step $\mathsf{let}(\lambda x.\mathsf{f}(x), \eta)$ contracts the redex $R_2 = \{121, 1211\}$, the first step $\mathsf{let}(\lambda x.\theta(x), \mathsf{b})$ contracts a redex $R_1 = \{2\}$ at position 2, and while $R_2 \leq_{\text{lex}} R_1$, it is also the case that $2 \notin R_2$. Alternatively, the last step $\rho(\lambda x.\mathsf{g}(x), \mathsf{b})$ contracts the redex $R_3 = \{\epsilon, 1, 11, 12\}$, and although $R_3 \leq_{\text{lex}} R_1$, it is also the case that $121 \notin R_3$.

On the other hand, the reduction

$$\rho(\lambda x.\mathsf{f}(x), \mathsf{a}), \theta(\mathsf{a}), \mathsf{g}(\eta)$$

is standard (and, as a matter of fact, equivalent to the first one).

Example 5.2.3. Let the following HRS be given:

$$\begin{aligned} \rho: &\quad \mathsf{let}(\lambda x.z(x), y) \to z(y) \\ \theta: &\quad \mathsf{f}(\mathsf{f}(x)) \to \mathsf{f}(x) \end{aligned}$$

Consider the steps:

$$\begin{aligned} \varphi_1 := &\quad \mathsf{f}(\rho(\lambda x.\mathsf{f}(x), \mathsf{a})) : \mathsf{f}(\mathsf{let}(\lambda x.\mathsf{f}(x), \mathsf{a})) \to \mathsf{f}(\mathsf{f}(\mathsf{a})) \\ \varphi_2 := &\quad \theta(\mathsf{a}) : \mathsf{f}(\mathsf{f}(\mathsf{a})) \to \mathsf{f}(\mathsf{a}) \end{aligned}$$

These steps contract the following redexes, resp. R_1 and R_2:

$$\begin{aligned} R_1 &= \{2, 21, 211, 212\} \\ R_2 &= \{\epsilon, 1, 2, 21\} \end{aligned}$$

Although $R_3 \leq_{\text{lex}} R_2$, the reduction $\varphi_1\,;\,\varphi_2$ is standard, because $2 \in R_2$.

The following lemma expresses that the labellings of the previous chapter preserve the property of standardness:

Lemma 5.2.4. *Let $\mathcal{R}$ be a relatively finite reduction standard for φ, and suppose $\mathcal{R} \approx \mathcal{S}$. Then $\mathcal{S}$ is a reduction standard for φ.*

Proof. By induction on the derivation of $\mathcal{R} \approx \mathcal{S}$. □

The main result of this chapter is the Standardization Theorem (Theorem 5.5.1), which states that each reduction has a unique, permutation equivalent standard reduction. This theorem will be proved by giving two standardization procedures which transform an arbitrary reduction into an equivalent standard one, selection standardization (Sect. 5.3) and inversion standardization (Sect. 5.4).

5.3 Selection standardization

In this section we define a procedure which, given an arbitrary reduction, produces a standard reduction equivalent to the input reduction. The procedure corresponds to weak standardization of Klop [25], but is called the *Selection Standardization procedure* here, after [43], due to its similarity to the selection sort algorithm. In selection sort, the smallest item of the list is selected and moved to the beginning of the list, after which the procedure is recursively called in order to sort the rest of the list. Similarly, the Selection Standardization procedure selects the left-most, outermost step (or, rather, the step which contracts the left-most and outermost redex) in the reduction and permutes it to the beginning of the reduction. Then the procedure is recursively applied to the rest of the reduction, producing a standard reduction in the end.

5.3.1 The standardization procedure

We define the standardization procedure by defining an algorithm which finds the left-most, outermost step of a reduction, and removes it from the reduction. This yields two objects: the left-most, outer-most step, and the tail of the reduction, that is, the reduction from which the left-most, outermost step has been removed. Now, the procedure is recursively applied to the tail of the reduction, and so on. The sequence steps which were removed in each iteration, now forms a standard reduction.

The result is defined here as the limiting reduction of a series of reductions which are only standard up to a certain point. This has the advantage that the heart of the procedure is a normal, finitistic algorithm, but still the method is well-defined for infinite reductions also. In Sect. 5.7 we investigate standardization of infinite reductions in some more detail.

The procedure presented in this section is a straight-forward adaption of the procedures 8.5.14 and 8.5.46 of [43], where only the first-order case is handled. However, the proof that the procedure terminates is much more involved due to the following reason. In first order term rewriting, the residuals of two parallel, un-nested redexes are parallel, un-nested redexes again. This fact helps in obtaining a well-founded ordering in which the recursive calls of the standardization function have a strictly smaller argument. In higher-order rewriting, however, applying a rule may nest two previously un-nested redexes, which makes the first-order proof method unavailable, because contracting a redex can now duplicate one of the redex's own 'family members'. Consider the HRS

$$\begin{aligned} \mathsf{h}(\lambda x.z(x), y) &\to z(z(y)) \\ \mathsf{f}(x) &\to \mathsf{g}(x, x) \end{aligned}$$

and the reduction

$$\mathsf{h}(\lambda x.\mathsf{f}(x), \mathsf{a}) \to \mathsf{f}(\mathsf{f}(x)) \to \mathsf{g}(\mathsf{f}(x), \mathsf{f}(x)).$$

We see that the second step duplicates the redex $\mathsf{f}(x)$, although the redex contracted in this step is a descendant of the same redex as the redex it duplicates. Because of the presence of these nestings, we require here Finite Family Developments to obtain an upper bound on the length of the reductions which can occur in the calculation of the standard reduction.

Definition 5.3.1 (Selection standardization)**.** Let $\mathcal{R}$ be a reduction starting at s.

(i) We define the functions *Lmc* and *Nlmc*, returning the (step contracting the) left-most contracted redex, and the rest of the reduction, respectively.

- If $\mathcal{R}$ contains no head steps, then one of the following two cases applies:
  Case 1A: Suppose $\mathcal{R} = f^*(\mathcal{R}_1, \ldots, \mathcal{R}_n)$, for reductions $\mathcal{R}_i$ starting at s_i. Let $\mathcal{R}_k$ be the left-most non-empty reduction among those. Then:
  $$\begin{aligned} \mathit{Lmc}(\mathcal{R}) &:= f(s_1, \ldots, s_{k-1}, \mathit{Lmc}(\mathcal{R}_k), s_{k+1}, \ldots, s_n) \\ \mathit{Nlmc}(\mathcal{R}) &:= f^*(s_1, \ldots, s_{k-1}, \mathit{Nlmc}(\mathcal{R}_k), \mathcal{R}_{k+1}, \ldots, \mathcal{R}_n) \end{aligned}$$
  Case 1B: Suppose $\mathcal{R} = \lambda x^*.\mathcal{R}'$. Then:
  $$\begin{aligned} \mathit{Lmc}(\mathcal{R}) &:= \lambda x.\mathit{Lmc}(\mathcal{R}') \\ \mathit{Nlmc}(\mathcal{R}) &:= \lambda x^*.\mathit{Nlmc}(\mathcal{R}') \end{aligned}$$
- Otherwise, $\mathcal{R}$ can be written as $\mathcal{S}\,;\varphi\,;\mathcal{T}$, where $\varphi = \rho(\varphi_1, \ldots, \varphi_n)$, for some rule $\rho : l \to r$ and $\varphi_i : u_i \multimap v_i$, is the first head step of $\mathcal{R}$. Let C be the largest context such that $\mathcal{P}os(C) \subseteq \mathcal{P}at(l)$ and $\mathcal{S} = C^*[\mathcal{S}_1, \ldots, \mathcal{S}_m]$, for reductions $\mathcal{S}_i : s_i \multimap\!\!\!\!\twoheadrightarrow t_i$.
  Case 2A: If $\mathcal{P}os(C) = \mathcal{P}at(l)$, then $\mathcal{S} = l^*(\mathcal{S}_1, \ldots, \mathcal{S}_n)$, and
  $$\begin{aligned} \mathit{Lmc}(\mathcal{R}) &:= \rho(s_1, \ldots, s_n) \\ \mathit{Nlmc}(\mathcal{R}) &:= r^*(\mathcal{S}_1, \ldots, \mathcal{S}_n)\,;r(\varphi_1, \ldots, \varphi_n)\,;\mathcal{T} \end{aligned}$$
  Case 2B: Otherwise, by maximality of C, some of the $\mathcal{S}_i$ must touch the redex pattern of φ. Let $\mathcal{S}_k$ be the leftmost such reduction. Now we define:
  $$\begin{aligned} \mathit{Lmc}(\mathcal{R}) &:= C[s_1, \ldots, s_{k-1}, \mathit{Lmc}(\mathcal{S}_k), s_{k+1}, \ldots, s_n] \\ \mathit{Nlmc}(\mathcal{R}) &:= C^*[\mathcal{S}_1, \ldots, \mathcal{S}_{k-1}, \mathit{Nlmc}(\mathcal{S}_k), \mathcal{S}_{k+1}, \ldots, \mathcal{S}_n]\,;\varphi\,;\mathcal{T} \end{aligned}$$

(ii) The n-bounded standard form of a reduction $\mathcal{R}$, written $\mathit{Std}^n_{\mathrm{S}}(\mathcal{R})$, is defined as follows: $\mathit{Std}^n_{\mathrm{S}}(\mathcal{R}) = \varepsilon$ if $\mathcal{R}$ consists of empty steps only, and otherwise:
$$\begin{aligned} \mathit{Std}^0_{\mathrm{S}}(\mathcal{R}) &:= \mathcal{R} \\ \mathit{Std}^{n+1}_{\mathrm{S}}(\mathcal{R}) &:= \mathit{Lmc}(\mathcal{R})\,;\mathit{Std}^n_{\mathrm{S}}(\mathit{Nlmc}(\mathcal{R})) \end{aligned}$$

(iii) The selection standard form of a reduction $\mathcal{R}$, written $Std_{\mathrm{S}}(\mathcal{R})$, is defined as the shortest reduction of which $Std^n_{\mathrm{S}}(\mathcal{R})[n]$ is a prefix for all n.

The algorithm for finding the left-most, outermost redex of a reduction intuitively works as follows:

- If the reduction does not contain a head step, then all the proof terms in the reduction have the same function symbol as head. Cases 1A and 1B 'zoom in' to the left-most non-empty reduction taking place in one of the arguments, by calling the algorithm recursively on it, and return the left-most, outermost step (and the tail) of this reduction.
- If the reduction does contain a head step, then there are to subcases. If none of the steps before the head step are involved in creating the redex pattern of the head step, then the head step contracts the left-most, outermost redex (Case 2A). On the other hand, if there is a step before the head step then the algorithm is recursively applied to the part of the reduction before the head step.

Before proving that the algorithm actually works, let's see some examples:

Example 5.3.2. Consider the one-rule HRS

$$\mu : \mathsf{mu}(\lambda x.z(x)) \to z(\mathsf{mu}(\lambda x.z(x)))$$

of Ex. 2.4.15 and let the multistep $\varphi = \mu(\lambda x.\rho(x))$ be given. We apply the standardization procedure to this multistep:

- First, $Lmc(\rho(x)) = \rho(x)$ and $Nlmc(\rho(x)) = \varepsilon$ by Case 2A.
- Then, $Lmc(\lambda x.\rho(x)) = \lambda x.\rho(x)$ and $Lmc(\rho(x)) = \varepsilon$ by Case 1B.
- Finally, $Lmc(\varphi) = \mu(\lambda x.\mathsf{f}(x))$ and $Nlmc(\varphi) = \rho(\mathsf{mu}(\lambda x.\rho(x)))$ by Case 2A.

So: $Std_{\mathrm{S}}(\varphi) = \mu(\lambda x.\mathsf{f}(x)), Std_{\mathrm{S}}(\rho(\mathsf{mu}(\lambda x.\rho(x))))$. By a similar derivation, we obtain

$$Std_{\mathrm{S}}(\varphi) = \mu(\lambda x.\mathsf{f}(x)), \rho(\mathsf{mu}(\lambda x.\mathsf{f}(x))), \mathsf{f}(\mathsf{mu}(\lambda x.\rho(x))).$$

Example 5.3.3. Let the following HRS be given:

$$\begin{aligned} \zeta :&\quad \mathsf{h}(\lambda x_1 x_2.z(x_1, x_2), y) \to z(y, y) \\ \eta :&\quad \mathsf{f}(x, \mathsf{b}) \to x \\ \theta :&\quad \mathsf{a} \to \mathsf{b} \end{aligned}$$

Consider the non-standard reduction

$$\mathcal{R} = \mathsf{h}(\lambda xy.\mathsf{f}(x, y), \theta), \zeta(\lambda xy.\mathsf{f}(x, y), \mathsf{b}), \eta(\mathsf{b}).$$

We execute the algorithm:

- First, by Case 2A:

$$\begin{aligned} Lmc(\mathcal{R}) &= \zeta(\lambda xy.\mathsf{f}(x,y),\mathsf{a}) \\ Nlmc(\mathcal{R}) &= \mathsf{f}(\theta,\theta),\eta(\mathsf{b}) \end{aligned}$$

- Let $\mathcal{R}' = Nlmc(\mathcal{R})$. Then, by Case 2B:

$$\begin{aligned} Lmc(\mathcal{R}') &= \mathsf{f}(\mathsf{a},\theta) \\ Nlmc(\mathcal{R}') &= \mathsf{f}(\theta,\mathsf{b}),\eta(\mathsf{b}) \end{aligned}$$

- Let $\mathcal{R}'' = Nlmc(\mathcal{R}')$. Then, by Case 2A:

$$\begin{aligned} Lmc(\mathcal{R}'') &= \eta(\mathsf{a}) \\ Nlmc(\mathcal{R}'') &= \theta \end{aligned}$$

- Finally, $Lmc(\theta) = \theta$ and $Nlmc(\theta) = \varepsilon$.

So: $Std_{\mathrm{S}}(\mathcal{R}) = \zeta(\lambda xy.\mathsf{f}(x,y),\mathsf{a}),\mathsf{f}(\mathsf{a},\theta),\eta(\mathsf{a}),\theta$.

In order to prove properties about the selection standardization algorithm, we need a measure on the reductions such that the arguments of each recursive call to the *Lmc* and *Nlmc* functions are strictly decreasing. For this, we define the 'depth' of the reduction as follows:

(i) The depth of a step φ is defined by:

$$\begin{aligned} \mathsf{dpt}(\lambda x.\varphi) &:= \mathsf{dpt}(\varphi) + 1 \\ \mathsf{dpt}(f(\varphi_1,\ldots,\varphi_n)) &:= \max_{1\leq i\leq n} \mathsf{dpt}(\varphi_i) + 1 \\ \mathsf{dpt}(\rho(\varphi_1,\ldots,\varphi_n)) &:= \max_{1\leq i\leq n} \mathsf{dpt}(\varphi_i) + 1 \end{aligned}$$

where $\max \varnothing = 0$.

(ii) The depth of a reduction $\mathcal{R}$ is defined by the following cases:

- If $\mathcal{R}$ is an empty reduction, then $\mathsf{dpt}(\mathcal{S}) = 0$.
- If $\mathcal{R}$ contains a head step, then $\mathcal{R} = \mathcal{S}\,;\varphi\,;\mathcal{T}$, where φ is the first head step of $\mathcal{R}$, and

$$\mathsf{dpt}(\mathcal{R}) := \mathsf{dpt}(\mathcal{S}).$$

- If $\mathcal{R} = f^*[\mathcal{R}_1,\ldots,\mathcal{R}_n]$, for some n-ary function symbol f, then

$$\mathsf{dpt}(\mathcal{R}) := \max_{1\leq i\leq n} \mathsf{dpt}(\mathcal{R}_i) + 1$$

- If $\mathcal{R} = \lambda x^*.\mathcal{R}_0$, then

$$\mathsf{dpt}(\mathcal{R}) := \mathsf{dpt}(\mathcal{R}_0) + 1$$

It is now easy to observe that, in the definition of *Lmc* and *Nlmc*, the depth of the argument of each recursive call is strictly decreasing. This makes it possible to use induction on the depth of a reduction as a proof method when proving properties of *Lmc* and *Nlmc*.

Lemma 5.3.4. *Let $\mathcal{R}, \mathcal{S}$ be (finite or infinite) reductions. If $\mathcal{R} =_{/1} \mathcal{S}$ then:*

(i) $Lmc(\mathcal{R}) = Lmc(\mathcal{S})$ *and* $Nlmc(\mathcal{R}) =_{/1} Nlmc(\mathcal{S})$*;*
(ii) for all $n \in \mathbb{N}$, $Std^n_{\mathrm{S}}(\mathcal{R}) = Std^n_{\mathrm{S}}(\mathcal{S})$*;*
(iii) $Std_{\mathrm{S}}(\mathcal{R}) = Std_{\mathrm{S}}(\mathcal{S})$.

Proof. (i) By induction on the depth of $\mathcal{R}$ and (ii) by induction on n using (i). Item (iii) follows directly from the definition and (ii). □

We need to prove that Std_{S} is well-defined. This claim boils down to a number of subclaims, which we prove in the following three lemmas.

The first subclaim may seem a bit counter-intuitive at first sight: it establishes that the length of the tail of the standard reduction is equal to the length of the original reduction. However, it is less counter-intuitive once we realize that the notion of length counts empty steps as well. The lemma proves that the star notation in case 1A of the definition may indeed be used (it is only defined if all subreductions have the same length).

Lemma 5.3.5. *Let $\mathcal{R}$ be a (finite or infinite) reduction.* $|Nlmc(\mathcal{R})| = |\mathcal{R}|$.

Proof. By induction on the depth of $\mathcal{R}$. □

The next lemma establishes that $Std_{\mathrm{S}}(\mathcal{R})$ is a well-formed reduction, for all $\mathcal{R}$, that is, that the target of each step is equal to the source of the step immediately following it.

Lemma 5.3.6. *Let $\mathcal{R}$ be a (finite or infinite) reduction.*

(i) $\mathrm{tgt}(Lmc(\mathcal{R})) = \mathrm{src}(Nlmc(\mathcal{R}))$*;*
(ii) $Nlmc(\mathcal{R})$ *is a well-defined reduction.*

Proof. Both items follow by induction on the depth of $\mathcal{R}$ □

Since $Std_{\mathrm{S}}(\mathcal{R})$ is defined as the limit of a sequence of reductions, we must show that such a limit actually exists. This is established in the third lemma:

Lemma 5.3.7. $Std^n_{\mathrm{S}}(\mathcal{R})[n]$ *is a prefix of* $Std^{n+1}_{\mathrm{S}}(\mathcal{R})[n+1]$.

Proof. By induction on n. □

Proposition 5.3.8. *Std is well-defined.*

Proof. By Lemmata 5.3.5, 5.3.6 and 5.3.7. □

5.3.2 Existence of standard reductions

In this subsection we show that for each relatively finite reduction there exists at least one equivalent standard reduction. This is proven by showing that we can use the standardization procedure of the previous subsection to find such a reduction. This boils down to three subclaims: first, that the Selection Standardization actually produces a reduction, that is, that it terminates when given a finite reduction as input (Theorem 5.3.13); second, that the reduction it returns is equivalent to the input reduction (Prop. 5.3.14); and third that the reduction it produces is actually a standard reduction (Theorem 5.3.15).

Lemma 5.3.9. *Let $\mathcal{R} = C^*[\mathcal{R}_1, \ldots, \mathcal{R}_n]$ be a (finite or infinite) reduction, with $s_i = \mathrm{src}(\mathcal{R}_i)$. Suppose $\mathcal{R}_k \approx \varphi_k \,;\, \mathcal{R}'_k$, for some $1 \leq k \leq n$, such that $|\mathcal{R}'_k| = |\mathcal{R}_k|$. Then:*

$$\mathcal{R} \approx C[s_1, \ldots, \varphi_k, \ldots, s_n] \,;\, C^*[\mathcal{R}_1, \ldots, \mathcal{R}'_k, \ldots, \mathcal{R}_n].$$

Proof. By Lemma 3.2.7, which can be applied due to the assumptions that $\mathcal{R}_k \approx \varphi_k;\mathcal{R}'_k$ and $|\mathcal{R}'_k| = |\mathcal{R}_k|$ and the easy fact that, by definition, $\mathcal{R}_i \approx s_i;\mathcal{R}_i$, we have:

$$\begin{aligned}
&C^*[\mathcal{R}_1, \ldots, \mathcal{R}_n] \\
&\approx C^*[s_1 \,;\, \mathcal{R}_1, \ldots, \varphi_k \,;\, \mathcal{R}'_k, \ldots, s_n \,;\, \mathcal{R}_n] \\
&\approx C[s_1, \ldots, \varphi_k, \ldots, s_n] \,;\, C^*[\mathcal{R}_1, \ldots, \mathcal{R}'_k, \ldots, \mathcal{R}_n]
\end{aligned}$$

which proves the result. □

Lemma 5.3.10. *Let $\rho\colon \,\rightarrow r$ be a rule, and let $\mathcal{R} = l^*(\mathcal{R}_1, \ldots, \mathcal{R}_n)$ be a finite reduction, with $\mathcal{R}_i\colon s_i \multimap\!\!\!\!\rightarrow\!\!\!\!\rightarrow t_i$. Then:*

$$l^*(\mathcal{R}_1, \ldots, \mathcal{R}_n) \,;\, \rho(t_1, \ldots, t_n) \approx \rho(s_1, \ldots, s_n) \,;\, r^*(\mathcal{R}_1, \ldots, \mathcal{R}_n).$$

Proof. By induction on the length of $\mathcal{R}$. If $\mathcal{R}$ is empty, then $s_i = t_i$ and the result follows immediately. Otherwise, let φ be the last step of $\mathcal{R}$, that is: $\mathcal{R} = \mathcal{R}' \,;\, \varphi$, where $\mathcal{R}' = l^*(\mathcal{R}'_1, \ldots, \mathcal{R}'_n)$ and $\varphi = l(\varphi_1, \ldots, \varphi_n)$, with $\mathcal{R}'_i\colon s_i \multimap\!\!\!\!\rightarrow\!\!\!\!\rightarrow t'_i$ and $\varphi_i\colon t'_i \multimap\!\!\!\!\rightarrow t_i$. We perform one permutation:

$$\mathcal{R}' \,;\, l(\varphi_1, \ldots, \varphi_n) \,;\, \rho(t_1, \ldots, t_n) \approx \mathcal{R}' \,;\, \rho(t'_1, \ldots, t'_n) \,;\, r(\varphi_1, \ldots, \varphi_n)$$

By the induction hypothesis:

$$l^*(\mathcal{R}'_1, \ldots, \mathcal{R}'_n) \,;\, \rho(t'_1, \ldots, t'_n) \approx \rho(s_1, \ldots, s_n) \,;\, r^*(\mathcal{R}'_1, \ldots, \mathcal{R}'_n)$$

which proves the desired result. □

Lemma 5.3.11. *Let $\mathcal{R}$ be a (finite or infinite) reduction.*

(i) for non-empty $\mathcal{R}$: $Lmc(\mathcal{R}) \,;\, Nlmc(\mathcal{R}) \approx \mathcal{R}$;

(ii) for all n, $Std^n_{\mathrm{S}}(\mathcal{R}) \approx \mathcal{R}$.

Proof. (i) By induction on the depth of $\mathcal{R}$. Since $\mathcal{R}$ is non-empty, Case 2A functions as base case, here. We distinguish the cases of Def. 5.3.1 ($\mathcal{R}_k$, $\mathcal{S}$ and $\mathcal{S}_k$ are defined in the text of the definition):

- *Cases 1A and 1B.* By the induction hypothesis,

  $$\mathcal{R}_k \approx \mathit{Lmc}(\mathcal{R}_k)\,;\,\mathit{Nlmc}(\mathcal{R}_k).$$

  By Lemma 5.3.5 it is the case that $|\mathit{Nlmc}(\mathcal{R}_k)| = |\mathcal{R}_k|$ and therefore we can apply Lemma 5.3.9, yielding the desired result.
- *Case 2A.* Directly from Lemma 5.3.10, which can be applied because, in the definition, $\mathcal{S}$ is finite by construction.
- *Case 2B.* By the induction hypothesis,

  $$\mathcal{R}_k \approx \mathit{Lmc}(\mathcal{R}_k)\,;\,\mathit{Nlmc}(\mathcal{R}_k).$$

  By Lemma 5.3.5 it is the case that $|\mathit{Nlmc}(\mathcal{R}_k)| = |\mathcal{R}_k|$ and therefore we can apply Lemma 5.3.9, easily yielding the required result.

(ii) By induction on n. If $n = 0$, $\mathit{Std}^n_{\mathrm{S}}(\mathcal{R}) = \mathcal{R}$, so the desired result follows trivially. Otherwise:

$$\mathit{Std}^n_{\mathrm{S}}(\mathcal{R}) = \mathit{Lmc}(\mathcal{R})\,;\,\mathit{Std}^{n-1}_{\mathrm{S}}(\mathit{Nlmc}(\mathcal{R})).$$

By the induction hypothesis, $\mathit{Std}^{n-1}_{\mathrm{S}}(\mathit{Nlmc}(\mathcal{R})) \approx \mathit{Nlmc}(\mathcal{R})$. Then the desired result follows from (i). □

Lemma 5.3.12. *Let $\mathcal{R}$ be a (finite or infinite) reduction. For each $n \in \mathbb{N}$:*

(i) $\mathcal{R}\mathit{ules}(\mathit{Std}^{n+1}_{\mathrm{S}}(\mathcal{R})) \subseteq \mathcal{R}\mathit{ules}(\mathit{Std}^n_{\mathrm{S}}(\mathcal{R}))$*;*
(ii) $\mathsf{D}(\mathit{Std}^{n+1}_{\mathrm{S}}(\mathcal{R}^\omega)) \leq \mathsf{D}(\mathit{Std}^n_{\mathrm{S}}(\mathcal{R}^\omega))$*;*

Proof. Both items follow by induction on on the depth of $\mathcal{R}$, using the fact that for each rule $l = \lambda\overline{x}.l_0 \to \lambda\overline{x}.r_0 = r$, l_0 must be a pattern containing all of $\overline{x}$. By this fact all rule symbols and labels in $r(\overline{\varphi})$ also appear in $l(\overline{\varphi})$, so that no 'new' rule symbols and labels are created in Case 2A of Def. 5.3.1. □

We prove that the Selection Standardization procedure terminates when given relatively finite reductions as input. The proof of this theorem given here relies on the property of Finite Family Developments proved in the previous chapter. We conjecture that an alternative proof can be found based on the strictly weaker property of Finite Developments.

Theorem 5.3.13. *Let $\mathcal{R}$ be a reduction. Then $\mathit{Std}_{\mathrm{S}}(\mathcal{R})$ is finite if and only if $\mathcal{R}$ is relatively finite. In fact, if $\mathcal{R}$ is relatively finite, then $\mathit{Std}_{\mathrm{S}}(\mathcal{R}) = \mathit{Std}^n_{\mathrm{S}}(\mathcal{R})$ for some $n \in \mathbb{N}$.*

Proof. ($\Rightarrow$): Follows from the fact that for a relatively infinite $\mathcal{R}$, $Lmc(\mathcal{R})$ is non-empty and $Nlmc(\mathcal{R})$ is relatively infinite. This is the case, because at each step of the procedure, only a finite prefix of the reduction is considered, and the rest is left as-is.

($\Leftarrow$): Assume that $\mathcal{R}$ is relatively finite. Let $\mathcal{R}_0$ contain only the non-empty steps of $\mathcal{R}$. Thus, $\mathcal{R}_0$ is finite. Also, since $\mathcal{R}_0 =_{/1} \mathcal{R}$, it follows from Lemma 5.3.4, $Std_{\mathrm{S}}(\mathcal{R}_0) = Std_{\mathrm{S}}(\mathcal{R})$. So, it remains to show that $Std_{\mathrm{S}}(\mathcal{R}_0)$ is finite.

We construct the reduction graph G which has as edges all non-empty steps that occur in some reduction $Std^n_{\mathrm{S}}(\mathcal{R}_0)$ for arbitrary n, and as vertices the sources and targets of these steps. Because $\mathcal{R}_0$ is finite, only finitely many rules are applied in it. So, by Lemma 5.3.12 (i), only finitely many rules occur in G. Since all terms are finite by definition, this means that G is finitely branching.

$\mathcal{R}_0$ is finite, and therefore there is a bound on its labels. In fact, by Lemma 5.3.12, there is a bound $\ell \in \mathbb{N}$ on the labels of each step in G, and thus a bound on the labels of each path of G. Therefore, by Finite Family Developments (Theorem 4.5.8) all paths of G are finite. Since $Std_{\mathrm{S}}(\mathcal{R}_0)$ is one of these paths, it, too, is finite.

By König's Lemma (Lemma 2.2.4) G is finite, and therefore there must be a longest path. Let n be the length of this path. Since $Lmc(\mathcal{R})$ is non-empty for non-empty R, it must be the case that $Std^n_{\mathrm{S}}(\mathcal{R}_0)[n] = Std_{\mathrm{S}}(\mathcal{R}_0)$. □

Proposition 5.3.14. *Let $\mathcal{R}$ be a relatively finite reduction. $Std_{\mathrm{S}}(\mathcal{R}) \approx \mathcal{R}$*

Proof. Follows from Lemma 5.3.11 and Theorem 5.3.13. □

Theorem 5.3.15. *Let $\mathcal{R}$ be a (finite or infinite) reduction:*

(i) for non-empty $\mathcal{R}$: $Nlmc(\mathcal{R})$ is standard for $Lmc(\mathcal{R})$;

(ii) $Std^n_{\mathrm{S}}(\mathcal{R})[n]$ is a standard reduction, for all $n \in \mathbb{N}$;

(iii) $Std(\mathcal{R})$ is a standard reduction.

Proof. (i) By induction on the depth of $\mathcal{R}$. Because $\mathcal{R}$ is non-empty by assumption, Case 2A must at some point occur, and function as base cases. We distinguish the following cases from the definition ($\mathcal{S}_k$ is defined in the text of the definition):

- *Cases 1A and 1B.* These follow easily from the induction hypothesis.
- *Case 2A.* In this case, $Lmc(\mathcal{R})$ is a head step, so $Nlmc(\mathcal{R})$ is standard for it.
- *Case 2B.* The required result follows from the following two facts: first, by the induction hypothesis, $Nlmc(\mathcal{S}_k)$ is standard for $Lmc(\mathcal{S}_k)$; and second, $i \neq k$, the $\mathcal{S}_i$ occur either below $Lmc(\mathcal{R})$, or $Lmc(\mathcal{R})$ contributes to them.

(ii) By induction on n we prove that for each step of $Std^n_{\mathrm{S}}(\mathcal{R})[n]$ it holds that the rest of the reduction is standard for it. If $n = 0$ or $\mathcal{R}$ is empty, then the claim follows trivially, because ε is standard by definition.

Otherwise, suppose that $n = n' + 1$. Let $\mathcal{R}' = Std^{n'}_{\mathrm{S}}(Nlmc(\mathcal{R}))$. Then $Std^n_{\mathrm{S}}(\mathcal{R}) = Lmc(\mathcal{R}) ; \mathcal{R}'$. By Lemma 5.3.11 $\mathcal{R}' \approx Nlmc(\mathcal{R})$, and thus it follows from (i) and Lemma 5.2.4 that $\mathcal{R}'$ is standard for $Lmc(\mathcal{R})$ and thus, in particular, $\mathcal{R}'[n']$ is standard for it. Furthermore, by the induction hypothesis each step of $\mathcal{R}[k']'$ is standard for the rest of the reduction, which concludes the proof.

(iii) Let $\mathcal{S} = Std(\mathcal{R})$. Suppose $\mathcal{S} = \varphi_1, \varphi_2, \ldots$ is not standard. Then there is some anti-standard pair φ_i, φ_j, where $i < j$. But now $Std^j_{\mathrm{S}}(\mathcal{R})[j] = \varphi_1, \ldots, \varphi_j$ is not a standard reduction, which contradicts (ii).[1] □

The fact that each relatively finite reduction has a permutation equivalent standard reduction is now a trivial consequence of the previous results:

Corollary 5.3.16. *For every relatively finite reduction there exists a permutation equivalent standard reduction.*

Proof. Let $\mathcal{R}$ be a relatively finite reduction. By Theorem 5.3.13 and Theorem 5.3.15, $Std_{\mathrm{S}}(\mathcal{S})$ is a standard reduction, and by Prop. 5.3.14, it is permutation equivalent to $\mathcal{R}$. □

5.3.3 Uniqueness of standard reductions

In this section we prove that for permutation equivalent finite reductions the Selection Standardization procedure yields the same standard reduction.

Lemma 5.3.17. *Let $\mathcal{R}, \mathcal{S}$ be finite, non-empty reductions. If $\mathcal{R} \approx \mathcal{S}$, then $Lmc(\mathcal{R}) = Lmc(\mathcal{S})$ and $Nlmc(\mathcal{R}) \approx Nlmc(\mathcal{S})$.*

Proof. By definition, it holds that $\mathcal{R} \approx \mathcal{S}$ if and only if $\mathcal{R} \Leftrightarrow^*_{\mathrm{P}} \mathcal{S}$. We prove the lemma by induction on the inference of $\mathcal{R} \Leftrightarrow^*_{\mathrm{P}} \mathcal{S}$. If $\mathcal{R} \Leftrightarrow^*_{\mathrm{P}} \mathcal{S}$ is derived by reflexivity, symmetry or transitivity, the result follows easily. Otherwise, one of the equivalences of the definition has been applied, that is, $\mathcal{R} = \mathcal{T} ; \mathcal{G} ; \mathcal{U}$ and $\mathcal{S} = \mathcal{T} ; \mathcal{D} ; \mathcal{U}$, for some equivalence $\mathcal{G} \Leftrightarrow \mathcal{D}$.

We prove the lemma by induction on the depth of $\mathcal{R}$. Suppose $\mathcal{R} = C^*[\mathcal{R}_1, \ldots, \mathcal{R}_n]$, for some non-empty base context C containing no rule symbols. In this case it also holds that $\mathcal{S} = C^*[\mathcal{S}_1, \ldots, \mathcal{S}_n]$ and, whichever equivalence rule was applied, $\mathcal{R}_i \Leftrightarrow^*_{\mathrm{P}} \mathcal{S}_i$. So, by the (nested) induction hypothesis,

$$Lmc(\mathcal{R}_i) = Lmc(\mathcal{S}_i) \quad \text{and} \quad Nlmc(\mathcal{R}_i) \approx Nlmc(\mathcal{S}_i).$$

One of the Cases 1A and 1B of the definition must apply. In both cases it is easily seen that $Lmc(\mathcal{R}) = Lmc(\mathcal{S})$ and $Nlmc(\mathcal{R}) \approx Nlmc(\mathcal{S})$.

[1] Note that, if $\mathcal{S}$ is finite, a more direct proof is possible. By Theorem 5.3.13, $\mathcal{S} = Std^m_{\mathrm{S}}(\mathcal{R})$ for some m, and then the claim follows directly from Theorem 5.3.15(ii).

Otherwise $\mathcal{R}$ and $\mathcal{S}$ both contain at least one head step. If the first head step occurs before or after $\mathcal{G}$ (and $\mathcal{D}$), the result follows easily. If the first head step occurs within $\mathcal{G}$, then one of the (flat-l) or (flat-r) equivalences has been applied, and $\mathcal{G} = \rho(\overline{\varphi})$, where $\varphi_i : s_i \multimap t_i$. Suppose the (flat-l) equivalence was applied, so $\mathcal{D} = \rho(\overline{s}), r(\overline{\varphi})$. According to the definition there are two possible cases:

- If $\mathcal{T} = l^*(\mathcal{T}_1, \ldots, \mathcal{T}_n)$, for reductions $\mathcal{T}_i : u_i \multimap\!\!\!\!\rightarrow s_i$, then Case 2A of the definition applies and we have:

$$\begin{aligned} \mathit{Lmc}(\mathcal{R}) &= \rho(u_1, \ldots, u_n) \\ \mathit{Nlmc}(\mathcal{R}) &= r^*(\mathcal{T}_1, \ldots, \mathcal{T}_n)\,;\, r(\varphi_1, \ldots, \varphi_n)\,;\, \mathcal{U} \\ \mathit{Lmc}(\mathcal{S}) &= \rho(u_1, \ldots, u_n) \\ \mathit{Nlmc}(\mathcal{S}) &= r^*(\mathcal{T}_1, \ldots, \mathcal{T}_n)\,;\, r(s_1, \ldots, s_n)\,;\, r(\varphi_1, \ldots, \varphi_n)\,;\, \mathcal{U} \end{aligned}$$

  It is easy to see that this satisfies the requirements of the lemma, because $r(s_1, \ldots, s_n) = 1$.

- Otherwise, Case 2B applies. In this case, the result follows from the induction hypothesis and the definition in much the same way as in Cases 1A and 1B.

The case that the (flat-r) equivalence rule was applied, is similar to the above. □

Proposition 5.3.18. *Let $\mathcal{R}, \mathcal{S}$ be finite reductions. If $\mathcal{R} \approx \mathcal{S}$, then it holds that $\mathit{Std}_{\mathrm{S}}(\mathcal{R}) = \mathit{Std}_{\mathrm{S}}(\mathcal{S})$.*

Proof. By induction on k we prove that $\mathit{Std}^k_{\mathrm{S}}(\mathcal{R})[k] = \mathit{Std}^k_{\mathrm{S}}(\mathcal{S})[k]$. In the base case, $k = 0$, and $\mathit{Std}^k_{\mathrm{S}}(\mathcal{R})[k] = \varepsilon = \mathit{Std}^k_{\mathrm{S}}(\mathcal{S})[k]$. If $k = k' + 1$, then, by definition, $\mathit{Std}^k_{\mathrm{S}}(\mathcal{R})[k] = \mathit{Lmc}(\mathcal{R})\,;\, \mathit{Std}^{k'}_{\mathrm{S}}(\mathit{Nlmc}(\mathcal{R}))[k']$. By Lemma 5.3.18:

$$\mathit{Lmc}(\mathcal{R}) = \mathit{Lmc}(\mathcal{S}) \qquad (\star)$$
$$\mathit{Nlmc}(\mathcal{R}) \approx \mathit{Nlmc}(\mathcal{S}) \qquad (\star\star)$$

By $(\star\star)$ we can apply the induction hypothesis, yielding:

$$\mathit{Std}^{k'}_{\mathrm{S}}(\mathit{Nlmc}(\mathcal{R}))[k'] = \mathit{Std}^{k'}_{\mathrm{S}}(\mathit{Nlmc}(\mathcal{S}))[k'] \qquad (\star\star\star)$$

By $(\star)$ and $(\star\star\star)$ we have:

$$\begin{aligned} &\mathit{Std}^k_{\mathrm{S}}(\mathcal{R})[k] \\ &= \mathit{Lmc}(\mathcal{R})\,;\, \mathit{Std}^{k'}_{\mathrm{S}}(\mathit{Nlmc}(\mathcal{R}))[k'] \\ &\approx \mathit{Lmc}(\mathcal{S})\,;\, \mathit{Std}^{k'}_{\mathrm{S}}(\mathit{Nlmc}(\mathcal{S}))[k'] \\ &= \mathit{Std}^k_{\mathrm{S}}(\mathcal{S})[k] \end{aligned}$$

The lemma follows from this and Theorem 5.3.13, which ensures that, for an arbitrary reduction $\mathcal{R}$, $\mathit{Std}(\mathcal{R}) = \mathit{Std}^k_{\mathrm{S}}(\mathcal{R})$, for some k. □

Note that this is not a proof that each permutation equivalence class of reductions contains at most one standard reduction: although it does show that the Selection Standardization procedure yields the same standard reduction for permutation equivalent reductions, it does not exclude the possibility that some permutation equivalence class of reductions does contain a second standard reduction which will never be obtained by the Selection Standardization procedure. The stronger claim is true, however, and will be proved in the next section, where an alternative standardization procedure is investigated.

5.4 Inversion standardization

In this section we explore a different method of finding standard reductions. This standardization procedure corresponds to Klop's strong standardization [25], but is called *Inversion Standardization* here, after Van Oostrom & De Vrijer [43], because of its resemblance to the inversion sort algorithm. In inversion sort, sometimes also called exchange sort, a list is sorted by exchanging adjacent list elements which are in the wrong order. Bubble sort, cocktail-shaker sort and gnome sort are specific implementations of inversion sort. Similarly, the Inversion Standardization procedure transforms a given reduction into a permutation equivalent standard one by non-deterministically exchanging adjacent (modulo inversion of steps which are parallel to each other) non-standard steps. However, because, in the standardization process (residuals of) redex patterns may be duplicated and nested, auxiliary work must be done to 'serialize' and 'un-nest' the steps of the reductions. Selection Standardization can be seen as a specific strategy for Inversion Standardization, viz. the strategy that iteratively permutes the leftmost-outermost step of the reduction to the front.

5.4.1 The standardization procedure

The Inversion Standardization algorithm is defined by a meta-rewrite system operating on reductions, the rules of which are carried out modulo the (par) equation of Def. 3.2.1. The normal forms of this system are then normalized with respect to a rewrite system which orders the parallel steps from left to right. All rules are instances of rules of permutation equivalence, or derived rules thereof.

Again, the major difficulty of the transformation of the procedure from the first-order to the higher-order case, is proving its termination. The problem is again that in first-order term rewriting, the residuals of two parallel, un-nested redexes are still un-nested, a fact which does not hold in higher-order rewriting. This makes it more difficult to find a reduction order in which each step of the meta-rewrite system strictly decreases the reduction. See also pag. 82.

Recall that a base context is a context in which the arguments of holes do not contain holes themselves. This means that such contexts can be used to distinguish nested and parallel subterms: if C is 2-ary base context, the

subterms M, N of $C[M, N]$ do not occur nested. This fact is useful for defining the meta-rewrite system:

Definition 5.4.1 (Inversion Standardization)**.**

(i) Let the rewrite systems (on reductions) $\Rightarrow_{\mathrm{Std}}$ contain the following rules:

$$C[l(\overline{\varphi})]\,, C[\rho(\overline{t})] \Rightarrow C[\rho(\overline{s})]\,, C[r(\overline{\varphi})] \quad \text{(std)}$$
$$C[\rho(\overline{\varphi})] \Rightarrow C[\rho(\overline{s})]\,, C[r(\overline{\varphi})] \quad \text{(flat)}$$
$$C[\varphi_1, \ldots, \varphi_n] \Rightarrow C[\varphi_1, s_2, \ldots, s_n]\,, C[t_1, \varphi_1, \ldots, \varphi_n] \quad \text{(ser)}$$
$$1 \Rightarrow \varepsilon \quad \text{(unit)}$$

where C is a base context *containing no rule symbols*, $\rho : l \to r$ and $\varphi_{(i)} : s_{(i)} \multimap t_{(i)}$. In the (std) and (flat) rules at least one of the φ_i is required to be non-empty, while in the (ser) rule all φ_i are required to be non-empty and $n \geq 2$.

The rewrite rules are applied *modulo* the equivalence relation $\approx_{\mathrm{par}}$ generated by the equation:

$$C[s, \psi]\,, C[\varphi, v] \approx C[\varphi, u]\,, C[t, \psi] \quad \text{(par)}$$

(ii) The rule of the system $\Rightarrow_{\mathrm{LR}}$ is the (par) equation oriented from left to right:

$$C[s, \psi]\,, C[\varphi, v] \Rightarrow C[\varphi, u]\,, C[t, \psi] \quad \text{(par)}$$

where C is a base context containing no rule symbols and $\varphi : s \geq t$ and $\psi : u \geq v$ are non-empty proof terms.

(iii) We use $\Rightarrow_{\mathrm{Std}}$ and $\Rightarrow_{\mathrm{LR}}$ to find an equivalent standard reduction of a reduction as follows:

$$\mathit{Std}_{\mathrm{I}}(\mathcal{R}) = \mathcal{S} \quad \text{if } \mathcal{R} \Rightarrow^{!}_{\mathrm{Std}} \mathcal{R}_0 \text{ and } \mathcal{R}_0 \Rightarrow^{!}_{\mathrm{LR}} \mathcal{S}$$

where $\Rightarrow^{!}$ denotes normalization w.r.t. $\Rightarrow$.

The $\mathit{Std}_{\mathrm{I}}(\mathcal{R})$ function of Def. 5.4.1 is not trivially well-defined: for that we need that both $\Rightarrow_{\mathrm{Std}}$ and $\Rightarrow_{\mathrm{LR}}$ are terminating and confluent. This will be proved in Sect. 5.4.2.

Note that the requirement that some (or all) of the $\varphi_{(i)}, \psi_{(i)}$ in the rules of $\Rightarrow_{\mathrm{Std}}$ and $\Rightarrow_{\mathrm{LR}}$ are non-empty is required. Without this restriction, both systems are trivially non-terminating. Consider for example the following simple infinite reductions:

$$C[\rho(\overline{s})] \Rightarrow_{\mathrm{Std}} C[\rho(\overline{s})]\,, C[r(\overline{s})] \Rightarrow_{\mathrm{Std}} \cdots$$
$$C[s, u]\,, C[s, u] \Rightarrow_{\mathrm{LR}} C[s, u]\,, C[s, u] \Rightarrow_{\mathrm{LR}} \cdots$$

Before proving well-definedness and correctness of the algorithm, let's review some examples of how the algorithm works:

Example 5.4.2. Consider one-rule HRS:

$$\mu : \mathsf{mu}(\lambda x.z(x)) \to z(\mathsf{mu}(\lambda x.z(x)))$$

of Ex. 2.4.15 and let the multistep $\mu(\lambda x.\rho(x))$ be given. We apply the standardization procedure to this multistep:

$$\begin{aligned}
&\mu(\lambda x.\rho(x))\\
&\Rightarrow_{(\mathrm{flat})}\ \mu(\lambda x.\mathsf{f}(x))\,,\,\rho(\mathsf{mu}(\lambda x.\rho(x)))\\
&\Rightarrow_{(\mathrm{flat})}\ \mu(\lambda x.\mathsf{f}(x)),\rho(\lambda x.\mathsf{mu}(\lambda x.\mathsf{f}(x))),\mathsf{g}(\lambda x.\mathsf{mu}(\lambda x.\rho(x)))
\end{aligned}$$

The result is in normal form w.r.t. $\Rightarrow_{\mathrm{Std}}$ and $\Rightarrow_{\mathrm{LR}}$.

Example 5.4.3. Let the following HRS be given:

$$\begin{aligned}
\rho :&\quad \mathsf{let}(\lambda x.z(x),y) \to z(y)\\
\theta :&\quad \mathsf{f}(x) \to \mathsf{g}(x)\\
\eta :&\quad \mathsf{a} \to \mathsf{b}
\end{aligned}$$

and consider the reduction $\mathsf{let}(\lambda x.\mathsf{f}(x),\eta)\,;\,\mathsf{let}(\lambda x.\theta(x),\mathsf{b})\,;\,\rho(\lambda x.\mathsf{g}(x),\mathsf{b})$ from $\mathsf{let}(\lambda x.\mathsf{f}(x),\mathsf{a})$ to $\mathsf{g}(\mathsf{b})$. Then a $\Rightarrow_{\mathrm{Std}}$-reduction of this reduction looks like:

$$\begin{aligned}
&\mathsf{let}(\lambda x.\mathsf{f}(x),\eta)\,,\,\mathsf{let}(\lambda x.\theta(x),\mathsf{b})\,,\,\rho(\lambda x.\mathsf{g}(x),\mathsf{b})\\
&\Rightarrow_{(\mathrm{std})}\ \mathsf{let}(\lambda x.\mathsf{f}(x),\eta)\,,\,\rho(\lambda x.\mathsf{f}(x),\mathsf{b})\,,\,\theta(\mathsf{b})\\
&\Rightarrow_{(\mathrm{std})}\ \rho(\lambda x.\mathsf{f}(x),\mathsf{a})\,,\,\mathsf{f}(\eta)\,,\,\theta(\mathsf{b})\\
&\Rightarrow_{(\mathrm{std})}\ \rho(\lambda x.\mathsf{f}(x),\mathsf{a})\,,\,\theta(\mathsf{a})\,,\,\mathsf{g}(\eta)
\end{aligned}$$

The result is in normal form w.r.t. $\Rightarrow_{\mathrm{Std}}$ and $\Rightarrow_{\mathrm{LR}}$.

Example 5.4.4. Let the following HRS be given:

$$\begin{aligned}
\zeta :&\quad \mathsf{h}(\lambda x_1x_2.z(x_1,x_2),y) \to z(y,y)\\
\eta :&\quad \mathsf{f}(x,\mathsf{b}) \to x\\
\theta :&\quad \mathsf{a} \to \mathsf{b}
\end{aligned}$$

Consider the non-standard reduction

$$\mathsf{h}(\lambda xy.\mathsf{f}(x,y),\theta)\,,\,\zeta(\lambda xy.\mathsf{f}(x,y),\mathsf{b})\,,\,\eta(\mathsf{b}).$$

We obtain the following meta-reduction starting from here:

$$\begin{aligned}
&\mathsf{h}(\lambda xy.\mathsf{f}(x,y),\theta)\,,\,\zeta(\lambda xy.\mathsf{f}(x,y),\mathsf{b})\,,\,\eta(\mathsf{b})\\
&\Rightarrow_{(\mathrm{std})}\ \zeta(\lambda xy.\mathsf{f}(x,y),\mathsf{a})\,,\,\mathsf{f}(\theta,\theta)\,,\,\eta(\mathsf{b})\\
&\Rightarrow_{(\mathrm{ser})}\ \zeta(\lambda xy.\mathsf{f}(x,y),\mathsf{a})\,,\,\mathsf{f}(\theta,\mathsf{a})\,,\,\mathsf{f}(\mathsf{b},\theta)\,;\,\eta(\mathsf{b})\\
&\approx_{\mathrm{par}}\quad \zeta(\lambda xy.\mathsf{f}(x,y),\mathsf{a})\,,\,\mathsf{f}(\mathsf{a},\theta)\,,\,\mathsf{f}(\theta,\mathsf{b})\,,\,\eta(\mathsf{b})\\
&\Rightarrow_{(\mathrm{std})}\ \zeta(\lambda xy.\mathsf{f}(x,y),\mathsf{a})\,,\,\mathsf{f}(\mathsf{a},\theta)\,,\,\eta(\mathsf{a})\,,\,\theta
\end{aligned}$$

The result is in normal form w.r.t. $\Rightarrow_{\mathrm{Std}}$ and $\Rightarrow_{\mathrm{LR}}$.

In the following example it is shown that the definition does not have any difficulties with non-orthogonal, even non-confluent HRSs. The reason is that we standardize reductions which are found beforehand. In other words, in cases that to redexes overlap, it is already decided which of the two will be contracted.

Example 5.4.5. Consider the following HRS:

$$\begin{aligned}\rho:&\quad \mathsf{h}(\lambda x.\mathsf{f}(x), y) \to y\\ \theta:&\quad \mathsf{h}(\lambda x.\mathsf{g}(x), y) \to \mathsf{b}\\ \eta:&\quad \mathsf{f}(x) \to \mathsf{g}(x)\end{aligned}$$

We standardize the following two reductions from $\mathsf{h}(\lambda x.\mathsf{f}(x), \mathsf{f}(\mathsf{a}))$ to normal form:

$$\begin{aligned}&\mathsf{h}(\lambda x.\mathsf{f}(x), \eta(\mathsf{a}))\,, \rho(\mathsf{g}(\mathsf{a}))\\ \Rightarrow_{(\mathrm{std})}\ &\rho(\mathsf{f}(\mathsf{a}))\,, \eta(\mathsf{a})\end{aligned}$$

$$\begin{aligned}&\mathsf{h}(\lambda x.\mathsf{f}(x), \eta(\mathsf{a}))\,, \mathsf{h}(\lambda x.\eta(x), \mathsf{g}(\mathsf{a}))\,, \theta(\mathsf{g}(\mathsf{a}))\\ \approx_{\mathrm{par}}\ &\mathsf{h}(\lambda x.\eta(x), \mathsf{f}(\mathsf{a}))\,, \mathsf{h}(\lambda x.\mathsf{g}(x), \eta(\mathsf{a}))\,, \theta(\mathsf{g}(\mathsf{a}))\\ \Rightarrow_{(\mathrm{std})}\ &\mathsf{h}(\lambda x.\eta(x), \mathsf{f}(\mathsf{a}))\,, \theta(\mathsf{f}(\mathsf{a}))\end{aligned}$$

5.4.2 Existence and uniqueness of standard reductions

In this section we prove that each reduction has an equivalent standard reduction, by showing that the procedure given above is well-defined and yields standard reductions. Contrary to the section about selection standardization, here we do not merely prove that the procedure yields the same standard reductions for permutation equivalent reductions; because we show both termination and confluence of the procedure, and because of the fact that the normal forms of the meta-rewrite system exactly correspond to the standard reductions, we also prove that each permutation equivalence class contains precisely one standard reduction.

Showing well-definedness amounts to proving termination and confluence of both the $\Rightarrow_{\mathrm{Std}}$ and the $\Rightarrow_{\mathrm{LR}}$ rewrite systems. First we show, that permutation equivalence is actually equal to convertibility in the inversion standardization meta-rewrite system:

Lemma 5.4.6. $(\Rightarrow_{\mathrm{Std}} \cup \Rightarrow_{\mathrm{Std}}^{-1} \cup \approx_{par})^* = \approx$.

Proof. ($\Rightarrow$): Simple, because all rules/equivalences of both $\Rightarrow_{\mathrm{Std}}$ and $\approx_{\mathrm{par}}$ are either instances of, or can be simulated by, the equations from Def. 3.2.1.

($\Leftarrow$): This is a bit more work, but it follows because each equation from Def. 3.2.1 can be simulated with a $(\Rightarrow_{\mathrm{Std}} \cup \approx_{\mathrm{par}})$-conversion. □

Lemma 5.4.7. *Let $\mathcal{R}, \mathcal{S}$ be finite reductions. If $\mathcal{R} \Rightarrow_{\mathrm{Std}} \mathcal{S}$ then:*

(i) $\mathcal{R}ules(\mathcal{R}) \subseteq \mathcal{R}ules(\mathcal{S})$;

(ii) $\mathsf{D}(\mathcal{R}) \leq \mathsf{D}(\mathcal{S})$.

Proof. Follows from the facts that in all the rules of Def. 5.4.1 the variables of the right-hand side are a subset of the variables of the left-hand side, and in the (par) equation the rule symbols on both sides of the equation are the same. Therefore, applying such a rule cannot 'make up' new labels or rule symbols. □

The restriction to finite reductions above is due to the fact that $\Rightarrow_{\mathrm{Std}}$ is only defined for finite reductions. Extending the meta-rewrite relation to infinite reductions is not difficult, and in this case the above lemma also holds: the proof does not depend on the finiteness of $\mathcal{R}$.

Remember that $\langle\!\langle \mathcal{R} \rangle\!\rangle_{\Rightarrow_{\mathrm{Std}}}$ means $\{\mathcal{S} \mid \mathcal{R} \Rightarrow^*_{\mathrm{Std}} \mathcal{S}\}$, that is the set of reductions $\Rightarrow_{\mathrm{Std}}$-reachable from $\mathcal{R}$. We can show, with the help of Finite Family Developments, that there is a bound on the length $|\cdot|$ of the reductions in $\langle\!\langle \mathcal{R} \rangle\!\rangle_{\Rightarrow_{\mathrm{Std}}}$.

Lemma 5.4.8. *Let $\mathcal{R}$ be a finite reduction. There is an $n \in \mathbb{N}$ such that for all $\mathcal{S} \in \langle\!\langle \mathcal{R} \rangle\!\rangle_{\Rightarrow_{\mathrm{Std}}}$, $|\mathcal{S}| \leq n$.*

Proof. Let G be the graph which has as edges the non-empty steps occurring in one of the $\mathcal{S} \in \langle\!\langle \mathcal{R} \rangle\!\rangle_{\Rightarrow_{\mathrm{Std}}}$, and as vertices the sources and targets of those steps. By Lemma 5.4.7 (ii) there is a bound on the labels of the steps of such $\mathcal{S}$, and therefore there is a bound on the labels of each path in G. Thus, by Finite Family Developments (Theorem 4.5.8), each path in G is finite. By Lemma 5.4.7 (i), G is finitely branching, and thus we can apply König's Lemma (Lemma 2.2.4) and obtain that G is finite. Therefore, there is a bound on the length of the paths of G and thus there is a bound on the number of non-empty steps in the reductions $\mathcal{S} \in \langle\!\langle \mathcal{R} \rangle\!\rangle_{\Rightarrow_{\mathrm{Std}}}$. The result of the lemma follows from that and the fact the number of empty steps in the reductions $\mathcal{S} \in \langle\!\langle \mathcal{R} \rangle\!\rangle_{\Rightarrow_{\mathrm{Std}}}$ are lower than or equal to the number of empty steps in $\mathcal{R}$. □

By the previous lemma, the following definition is well-defined:

Definition 5.4.9. Let $\mathcal{R}$ be a reduction. The *standard length* of $\mathcal{R}$, written $|\mathcal{R}|_{\mathrm{Std}}$, is defined as the smallest n such that $|\mathcal{S}| \leq n$ for all $\mathcal{S} \in \langle\!\langle \mathcal{R} \rangle\!\rangle_{\Rightarrow_{\mathrm{Std}}}$.

Lemma 5.4.10. *Let $\mathcal{R}, \mathcal{S}$ be finite reductions. Then:*

$$|\mathcal{R}\,;\mathcal{S}|_{\mathrm{Std}} \geq |\mathcal{R}|_{\mathrm{Std}} + |\mathcal{S}|_{\mathrm{Std}}.$$

Proof. Let $\mathcal{R}', \mathcal{S}'$ be maximal (w.r.t. length) reductions in $\langle\!\langle \mathcal{R} \rangle\!\rangle_{\Rightarrow_{\mathrm{Std}}}$ and $\langle\!\langle \mathcal{S} \rangle\!\rangle_{\Rightarrow_{\mathrm{Std}}}$, respectively. Then $\mathcal{R}\,;\mathcal{S} \Rightarrow^*_{\mathrm{Std}} \mathcal{R}'\,;\mathcal{S}'$ and thus $\mathcal{R}'\,;\mathcal{S}' \in \langle\!\langle \mathcal{R}\,;\mathcal{S} \rangle\!\rangle_{\Rightarrow_{\mathrm{Std}}}$. Therefore:

$$|\mathcal{R}\,;\mathcal{S}|_{\mathrm{Std}} \geq |\mathcal{R}'\,;\mathcal{S}'| = |\mathcal{R}'| + |\mathcal{S}'| = |\mathcal{R}|_{\mathrm{Std}} + |\mathcal{S}|_{\mathrm{Std}}.$$ □

The following corollary is a direct consequence of the above lemma:

Corollary 5.4.11. *Let $\mathcal{R}$ be a finite reduction, and φ a non-empty multistep. Then:*

(i) $|\mathcal{R}|_{\mathrm{Std}} < |\mathcal{R}\,;\varphi|_{\mathrm{Std}}$

(ii) $|\mathcal{R}|_{\mathrm{Std}} < |\varphi\,;\mathcal{R}|_{\mathrm{Std}}$

Proof. Directly from Lemma 5.4.10 and the fact that $|\varphi|_{\mathrm{Std}} \geq 1$ because none of the rules has a non-empty step in the left-hand side and an empty one in the right-hand side. □

With the above auxiliary results, we can prove termination of the $\Rightarrow_{\mathrm{Std}}$ meta-rewrite system. Note that the auxiliary results make use of the property of Finite Family Developments, and thus so does (indirectly) the following theorem.

Theorem 5.4.12. *The rewrite system $\Rightarrow_{\mathrm{Std}}$ (modulo $\approx_{par}$) is terminating.*

Proof. Before we begin the actual proof, we define the *rule length* of a step φ, written $\#_{\mathrm{R}}(\varphi)$, as the number of rule symbols in φ. Furthermore, recall that φ be called a *head multistep* if it is of the form $\rho(\varphi_1, \ldots, \varphi_n)$, where ρ is a rule symbol and the φ_i may contain other rule symbols.

Now, assume that $\Rightarrow_{\mathrm{Std}}$ is not terminating, and let

$$\mathcal{R} \Rightarrow_{\mathrm{Std}} \mathcal{R}_1 \Rightarrow_{\mathrm{Std}} \mathcal{R}_2 \Rightarrow_{\mathrm{Std}} \cdots$$

be a minimal (with respect to the length of the source term of the reduction $\mathcal{R}$) infinite meta-reduction from a finite reduction $\mathcal{R}$.

Since $\mathcal{R}$ is finite, all $\mathcal{R}' \in \langle\!\langle \mathcal{R} \rangle\!\rangle_{\Rightarrow_{\mathrm{Std}}}$ are finite and can be uniquely written as

$$\mathcal{R}' = \mathcal{S}_1\,;\varphi_1\,;\cdots;\mathcal{S}_n\,;\varphi_n\,;\mathcal{S}_{n+1}$$

for some $n \in \mathbb{N}$, where the φ_i are the head multisteps of $\mathcal{R}'$ and the $\mathcal{S}_i$ do not contain any head multisteps. We associate with each reduction $\mathcal{R}' \in \langle\!\langle \mathcal{R} \rangle\!\rangle_{\Rightarrow_{\mathrm{Std}}}$ the following value tuple:

$$\mathcal{V}(\mathcal{R}') = \langle |\mathcal{S}_1|_{\mathrm{Std}}, \#_{\mathrm{R}}(\varphi_1), \ldots, |\mathcal{S}_n|_{\mathrm{Std}}, \#_{\mathrm{R}}(\varphi_n), |\mathcal{S}_{n+1}|_{\mathrm{Std}} \rangle.$$

Since $<$ is well-founded on all its components and the number of head multisteps in the reductions is bounded (by Lemma 5.4.8 the length of the reductions reachable from $\mathcal{R}$ is bounded, and thus also the number of head multisteps in them), the lexicographic extension of $<$ is well-founded on the tuples, and we can order the reductions in $\langle\!\langle \mathcal{R} \rangle\!\rangle_{\Rightarrow_{\mathrm{Std}}}$ with this ordering.

Since a head step cannot be parallel to another step, all $\approx_{\mathrm{par}}$-steps occur within the $\mathcal{S}_i$. Therefore, it holds that if $\mathcal{R} \approx_{\mathrm{par}} \mathcal{S}$, then $\mathcal{V}(\mathcal{R}) = \mathcal{V}(\mathcal{S})$. Now we show, that a succession of $\Rightarrow_{\mathrm{Std}}$-steps will always eventually stricly decrease the value of the reduction.

Consider the $(i-1)$th step of the infinite meta-reduction: $\mathcal{R}_i \Rightarrow_{\mathrm{Std}} \mathcal{R}_{i+1}$, and let $\mathcal{R}_i$ and $\mathcal{R}_{i+1}$ be partitioned in the above way:

$$\mathcal{R}_i = \mathcal{S}_1 \,;\, \varphi_1 \,;\, \cdots \,;\, \mathcal{S}_n \,;\, \varphi_n \,;\, \mathcal{S}_{n+1}$$
$$\mathcal{R}_{i+1} = \mathcal{T}_1 \,;\, \psi_1 \,;\, \cdots \,;\, \mathcal{T}_n \,;\, \psi_n \,;\, \mathcal{T}_{n+1}$$

where the φ_i and ψ_i are the head multisteps of $\mathcal{R}_i$ and $\mathcal{R}_{i+1}$, respectively. There are two possibilities.

- If the $\Rightarrow_{\mathrm{Std}}$-step does not involve a head step, it is the case that, for some $i \in \{1, \ldots, n+1\}$, $\mathcal{S}_i \Rightarrow_{\mathrm{Std}} \mathcal{T}_i$, and for all $j \neq i$, $\mathcal{S}_j = \mathcal{T}_j$. Additionally, for all $i \in \{1 \ldots n\}$, $\varphi_i = \psi_i$. Obviously, $|\mathcal{S}_i|_{\mathrm{Std}} \geq |\mathcal{T}_i|_{\mathrm{Std}}$, and thus $\mathcal{V}(\mathcal{R}_i) \geq_{\mathrm{lex}} \mathcal{V}(\mathcal{R}_{i+1})$.

  However, by the minimality assumption, such $\Rightarrow_{\mathrm{Std}}$-steps cannot occur infinitely many times after each other, because then an infinite meta-reduction exists in which the object reductions have a smaller source. So, at some point, the second case must occur.

- Suppose the $\Rightarrow_{\mathrm{Std}}$-step involves a head step φ_k; this means that for all $i < k$, $\varphi_i = \psi_k$ and $\mathcal{S}_i = \mathcal{T}_i$. We distinguish cases based on the $\Rightarrow_{\mathrm{Std}}$-rule applied:

  - (std): In this case we know that:

    $$\mathcal{S}_k \,;\, \varphi_k = \mathcal{S}'_k \,;\, l(\overline{\chi}), \rho(\overline{t}) \Rightarrow_{\mathrm{Std}} \mathcal{S}'_k \,;\, \rho(\overline{s}), r(\overline{\chi}) = \mathcal{T}_k \,;\, \rho(\overline{s}), r(\overline{\chi})$$

    for some rule $\rho : l \to r$ multisteps $\chi_i : s_i \multimap t_i$.
    It follows from Lemma 5.4.10, together with the assumption from Def. 5.4.1 that at least one of the χ_i is non-empty, that:

    $$|\mathcal{S}_k|_{\mathrm{Std}} = |\mathcal{T}_k \,;\, l(\overline{\chi})|_{\mathrm{Std}} > |\mathcal{T}_k|_{\mathrm{Std}}$$

    and therefore $\mathcal{V}(\mathcal{R}_i) >_{\mathrm{lex}} \mathcal{V}(\mathcal{R}_{i+1})$.
  - (flat): In this case we know that $\mathcal{S}_k = \mathcal{T}_k$ and

    $$\varphi_k = \rho(\overline{\chi}) \Rightarrow_{\mathrm{Std}} \rho(\overline{s}), r(\overline{\chi}) = \psi_k, r(\overline{\chi})$$

    for some rule $\rho : l \to r$ multisteps $\chi_i : s_i \multimap t_i$.
    Since one of the χ_j must be non-empty by definition, it holds that:

    $$\#_{\mathrm{R}}(\varphi) = \#_{\mathrm{R}}(\rho(\overline{\chi})) > \#_{\mathrm{R}}(\rho(\overline{s}))\#_{\mathrm{R}}(\psi_k)$$

    and therefore $\mathcal{V}(\mathcal{R}_i) >_{\mathrm{lex}} \mathcal{V}(\mathcal{R}_{i+1})$.
  - (ser): Because of the requirement that $n \geq 2$ in the application of the (ser)-rule, the context cannot be empty and so the left-hand side cannot be a head multistep.
  - (unit): Since head multisteps cannot be empty, this case cannot occur.

The above facts contradict the assumption that an infinite $\Rightarrow_{\mathrm{Std}}$-reduction exists, and therefore we must reject that assumption. □

We are now going to prove that $\Rightarrow_{\mathrm{Std}}$ is also confluent. We could do this by a critical pair analysis, but this is difficult because we are working modulo $\approx_{\mathrm{par}}$-equivalence. We chose the easy way out, and prove the property in a somewhat weaker sense: if $\mathcal{S}_1 \Leftarrow_{\mathrm{Std}} \mathcal{R} \Rightarrow_{\mathrm{Std}} \mathcal{R}_2$ we select $Std_{\mathrm{S}}(\mathcal{R})$ as the common reduct of $\mathcal{S}_1$ and $\mathcal{S}_2$. For this, we need the following auxiliary lemma:

Lemma 5.4.13. *Let $\mathcal{R}, \mathcal{S}$ be reductions.*

(i) If $\mathcal{R} \Rightarrow_{\mathrm{Std}} \mathcal{S}$, then $Std_{\mathrm{S}}(\mathcal{R}) = Std_{\mathrm{S}}(\mathcal{S})$.

(ii) $\mathcal{R} \Rightarrow^*_{\mathrm{Std}} Std_{\mathrm{S}}(\mathcal{R})$.

Proof. (i) This follows immediately from Lemma 5.4.6 and Prop. 5.3.18.

(ii) The claim follows from the facts that, if $\mathcal{R}$ is empty, $\mathcal{R} = Std_{\mathrm{S}}(\mathcal{R})$, and if $\mathcal{R}$ is non-empty, then

$$\mathcal{R} \Rightarrow^*_{\mathrm{Std}} Lmc(\mathcal{R}) \,;\, Nlmc(\mathcal{R}). \qquad (\star)$$

We prove $(\star)$ by induction on the depth of $\mathcal{R}$. We distinguish the cases from the definition (since $\mathcal{R}$ is non-empty by assumption, at some point Case 2A must apply, which functions as the base case):

- *Case 1A and 1B.* In these cases, $\mathcal{R} = C^*[\mathcal{R}_1, \ldots, \mathcal{R}_n]$. Let k be the index of the first of the $\mathcal{R}_i$ which is non-empty. If the left-most contracted redex of $\mathcal{R}_k$ is part of a multistep containing at least two redexes, then we perform as many (flat)- and (ser)-steps to isolate this redex, yielding a reduction $\mathcal{R}' = C^*[\mathcal{R}'_1, \ldots, \mathcal{R}'_n]$ such that $\mathcal{R} \Rightarrow^*_{\mathrm{Std}} \mathcal{R}'$. By the induction hypothesis, $\mathcal{R}'_k \Rightarrow^*_{\mathrm{Std}} Lmc(\mathcal{R}'_k) \,;\, Nlmc(\mathcal{R}'_k)$. By construction, $Lmc(\mathcal{R}'_k) = Lmc(\mathcal{R}_k)$ and $Nlmc(\mathcal{R}'_k) = Nlmc(\mathcal{R}'_k)$, from which proving the desired result is easy.
- *Case 2A.* In this case, $\mathcal{R} = l^*(\mathcal{S}_1, \ldots, \mathcal{S}_n) \,;\, \rho(\varphi_1, \ldots, \varphi_n) \,;\, \mathcal{T}$. It is easy to see that $\mathcal{R} \Rightarrow^*_{\mathrm{Std}} \rho(s_1, \ldots, s_n) \,;\, r^*(\mathcal{S}_1, \ldots, \mathcal{S}_n) \,;\, \mathcal{T}$ by a number of applications of the (std)-rule.
- *Case 2B.* This case is analogous to cases 1A and 1B. □

Theorem 5.4.14. *The rewrite system $\Rightarrow_{\mathrm{Std}}$ (modulo $\approx_{par}$) is confluent.*

Proof. Since we have proved termination of $\Rightarrow_{\mathrm{Std}}$ in Theorem 5.4.12, it suffices to show local confluence here. Suppose

$$\mathcal{S}_1 \Leftarrow_{\mathrm{Std}} \mathcal{R} \Rightarrow_{\mathrm{Std}} \mathcal{R}_2.$$

By Lemma 5.4.13(i),

$$Std_{\mathrm{S}}(\mathcal{S}_1) = Std_{\mathrm{S}}(\mathcal{R}) = Std_{\mathrm{S}}(\mathcal{S}_2)$$

and by Lemma 5.4.13 (ii),

$$\mathcal{S}_1 \Rightarrow^*_{\text{Std}} \mathit{Std}_{\text{S}}(\mathcal{R}) \Leftarrow^*_{\text{(Std)}} \mathcal{S}_2,$$

as required. □

Proposition 5.4.15. *The rewrite system* $\Rightarrow_{\text{LR}}$ *is complete.*

Proof. In order to show completeness, we must show termination and confluence. The $\Rightarrow_{\text{LR}}$ rewrite system sorts steps that are parallel to each other in left-to-right order; the proof of completeness therefore closely follows the proof of completeness of the inversion sort algorithm.

We begin by proving termination. Let $\mathcal{R}$ be a finite reduction. It is clear from the definition of the $\Rightarrow_{\text{LR}}$-rewrite system that it does not alter the number of steps in the reduction, that is, if $\mathcal{R} \Rightarrow^*_{\text{LR}} \mathcal{S}$, then $|\mathcal{R}| = |\mathcal{S}|$. Given this fact, the following ordering on reductions is well-founded on $\mathcal{S} \in \langle\!\langle \mathcal{R} \rangle\!\rangle_{\Rightarrow_{\text{LR}}}$:

$$\begin{aligned} C[s, \psi] \,;\, \mathcal{R} &>_{\text{LR}} C[\varphi, u] \,;\, \mathcal{S} \\ \varphi \,;\, \mathcal{R} &>_{\text{LR}} \varphi \,;\, \mathcal{S} \quad \text{if } \mathcal{R} >_{\text{LR}} \mathcal{S} \end{aligned}$$

where φ is a non-empty step in the first equation and a possibly empty step in the second. Now, termination follows from the fact that if $\mathcal{R} \Rightarrow_{\text{LR}} \mathcal{S}$, then $\mathcal{R} >_{\text{LR}} \mathcal{S}$.

Given termination confluence follows from local confluence, which in turn follows from the fact that there is one critical pair, which can be joined:

$$\begin{aligned} & C[s_1, s_2, \varphi_3] \,;\, C[s_1, \varphi_2, t_3] \,;\, C[\varphi_1, t_2, t_3] \\ & \Rightarrow_{\text{LR}} C[s_1, \varphi_2, s_3] \,;\, C[s_1, t_2, \varphi_3] \,;\, C[\varphi_1, t_2, t_3] \\ & \Rightarrow_{\text{LR}} C[s_1, \varphi_2, s_3] \,;\, C[\varphi_1, t_2, s_3] \,;\, C[t_1, t_2, \varphi_3] \\ & \Rightarrow_{\text{LR}} C[\varphi_1, s_2, s_3] \,;\, C[t_1, \varphi_2, s_3] \,;\, C[t_1, t_2, \varphi_3] \end{aligned}$$

$$\begin{aligned} & C[s_1, s_2, \varphi_3] \,;\, C[s_1, \varphi_2, t_3] \,;\, C[\varphi_1, t_2, t_3] \\ & \Rightarrow_{\text{LR}} C[s_1, s_2, \varphi_3] \,;\, C[\varphi_1, s_2, t_3] \,;\, C[t_1, \varphi_2, t_3] \\ & \Rightarrow_{\text{LR}} C[\varphi_1, s_2, s_3] \,;\, C[t_1, s_2, \varphi_3] \,;\, C[t_1, \varphi_2, t_3] \\ & \Rightarrow_{\text{LR}} C[\varphi_1, s_2, s_3] \,;\, C[t_1, \varphi_2, s_3] \,;\, C[t_1, t_2, \varphi_3] \end{aligned}$$

□

By the previous results, $\mathit{Std}_{\text{I}}(\cdot)$ is actually a function, so each result is actually unique. From this it follows that Def. 5.4.1 (iii) is well-defined. It remains to show that the normal forms of the meta-rewrite system correspond exactly to the standard reductions:

Proposition 5.4.16.

(i) If $\mathcal{S}$ *is a standard reduction, it is a normal form w.r.t.* $\Rightarrow_{\text{Std}}$ *and* $\Rightarrow_{\text{LR}}$.

(ii) Normal forms w.r.t. $\Rightarrow_{\text{Std}}$ *and* $\Rightarrow_{\text{LR}}$ *are standard reductions.*

Proof. (i) Suppose $\mathcal{R}$ is a reduction which is not in normal form w.r.t. either $\Rightarrow_{\mathrm{Std}}$ or $\Rightarrow_{\mathrm{LR}}$.

- If a (flat), (ser) or (unit) rule can be applied, then $\mathcal{R}$ has at least one step with either more than one or zero rule symbols in it, and so is not a proper reduction and thus not standard.
- If a (std) rule can be applied, then there are adjacent (modulo $\approx_{\mathrm{par}}$) steps $C[l(\overline{\varphi})]$ and $\rho(\overline{t})$, which form an anti-standard pair.
- If a (*par*) rule can be applied, then there are adjacent steps $C[s, \psi]$ and $C[\varphi, v]$, which form an anti-standard pair.

(ii) Let $\mathcal{R}$ be an arbitrary reduction. By Lemma 5.4.13 $\mathcal{R} \Rightarrow^*_{\mathrm{Std}} \mathit{Std}_{\mathrm{S}}(\mathcal{R})$. By Theorem 5.3.15, $\mathit{Std}_{\mathrm{S}}(\mathcal{R})$ is a standard reduction, and therefore by item (i) it is the normal form w.r.t. $\Rightarrow_{\mathrm{Std}}$ and $\Rightarrow_{\mathrm{LR}}$. □

5.5 The Standardization Theorem and standardization equivalence

The main result of this chapter is the Standardization Theorem. Here, we give its proof.

Theorem 5.5.1 (Standardization Theorem)**.** *For every finite reduction there exists a unique, permutation equivalent, standard reduction. This standard reduction is the same for permutation equivalent reductions.*

Proof. Existence of standard reductions follows from Theorem 5.4.12, Theorem 5.4.14 and Prop. 5.4.15 (well-definedness of $\mathit{Std}_{\mathrm{I}}$), Lemma 5.4.6 (the fact that $\mathit{Std}_{\mathrm{I}}$ produces equivalent reductions) and Prop. 5.4.16 (the fact that $\mathit{Std}_{\mathrm{I}}$ produces standard reductions). The claim of uniqueness in first sentence of the theorem and the entire second sentence follow from the fact that confluence in rewriting implies the Church–Rosser property. □

Note that, Lemma 5.4.13, Selection Standardization and Inversion Standardization produce the same results. In the following, we will therefore omit the subscript and just write $\mathit{Std}(\mathcal{R})$ for the equivalent standard reduction of $\mathcal{R}$.

The Standardization Theorem provides an elegant way to formalize the notion of equivalence of reductions: we call two reductions equivalent if they have their unique equivalent standard reductions are the same.

Definition 5.5.2. Reductions $\mathcal{R}, \mathcal{S}$ are *standardization equivalent*, written $\mathcal{R} \equiv \mathcal{S}$, if $\mathit{Std}(\mathcal{R}) = \mathit{Std}(\mathcal{S})$.

Note that the $=$ in the definition above denotes literal equality. The fact that standardization and permutation equivalence are equivalent is an easy corollary of the Standardization Theorem:

Corollary 5.5.3. *Let $\mathcal{R}, \mathcal{S}$ be finite reductions. $\mathcal{R} \equiv \mathcal{S}$ if and only if $\mathcal{R} \approx \mathcal{S}$.*

Proof. Follows directly from the definition and Theorem 5.5.1. □

5.6 Non-local HRSs

In the previous sections we restricted our attention to local (that is linear and fully extended) HRSs. This is not without reason. In this section we show by means of two counter examples that standardization of reductions of a non-local HRS is problematic.

Example 5.6.1 (Non-fully-extended). Consider the following, non fully extended, HRS:

$$\begin{aligned} \rho: &\quad \mathsf{f}(\lambda x.y) \to \mathsf{g}(y) \\ \theta: &\quad \mathsf{h}(x) \to \mathsf{a} \end{aligned}$$

and the reduction:

$$\mathcal{R} : \mathsf{f}(\lambda x.\mathsf{h}(x)) \to \mathsf{f}(\lambda x.\mathsf{a}) \to \mathsf{g}(\mathsf{a})$$

So, $\mathcal{R} = \mathsf{f}(\lambda x.\theta(x)) \,;\, \rho(a)$. This reduction is not standard, according to our definition, because the redex contracted in the first step is not to the left of the redex contracted in the second step. However, the two steps cannot be swapped: the first step removes the bound variable x, which is a prerequisite for applying the second step. $\mathcal{R}$ does not have an equivalent standard reduction.

Example 5.6.2 (Non-linear). Consider the non-linear HRS:

$$\begin{aligned} \rho: &\quad \mathsf{f}(x,x) \to x \\ \theta: &\quad \mathsf{a} \to \mathsf{b} \end{aligned}$$

and the reduction:

$$\mathcal{R} : \mathsf{f}(\mathsf{a},\mathsf{b}) \to \mathsf{f}(\mathsf{b},\mathsf{b}) \to \mathsf{b}$$

This reduction is not standard according to our definition. However, it is not possible to swap the two steps because the first step makes sure that the arguments of the function symbol f become equal, which is a prerequisite for the second step to be applied. So, $\mathcal{R}$ does not have an equivalent standard reduction.

5.7 Standardization of infinite reductions

The notion of standard reduction (Def. 5.2.1) can also be applied to infinite reductions. However, the Standardization Theorem (Theorem 5.5.1) is limited to finite reductions only. In this section we briefly investigate how much of the Standardization Theorem can be retained for infinite reductions.

Inversion Standardization is defined by a rewrite system (which operates on finite objects), and can therefore not be used with infinite reductions. In the definition of Selection Standardization, on the other hand, we have made

sure that everything is well-defined for infinite reductions also (although, of course, it will in practice not terminate). Still, it cannot be used to extend the Standardization Theorem to infinite reductions in full, because Prop. 5.3.14 is limited to finite ones only. In fact, the Standardization Theorem does not hold for infinite reductions, not even in the first-order case, as the following counter example shows:

Example 5.7.1. Consider the following TRS:

$$\begin{aligned} \mathsf{f}(x) &\to \mathsf{g}(\mathsf{f}(x)) \\ \mathsf{a} &\to \mathsf{b} \end{aligned}$$

and the following non-terminating reduction sequence:

$$\mathcal{R} : \mathsf{f}(\mathsf{a}) \to \mathsf{f}(\mathsf{b}) \to \mathsf{g}(\mathsf{f}(\mathsf{b})) \to \mathsf{g}(\mathsf{g}(\mathsf{f}(\mathsf{b}))) \to \cdots$$

However, we have:

$$\mathit{Std}_{\mathrm{S}}(\mathcal{R}) : \mathsf{f}(\mathsf{a}) \to \mathsf{g}(\mathsf{f}(\mathsf{a})) \to \mathsf{g}(\mathsf{g}(\mathsf{f}(\mathsf{a}))) \to \cdots$$

Observe that the residuals of the redex contracted in the first step of $\mathcal{R}$ are never contracted in any step of $\mathit{Std}_{\mathrm{S}}(\mathcal{R})$ (the first step is, as it were, postponed indefinitely). This is undesired in a reduction which is supposed to be equivalent.

A different counter-example to the same claim, which is more complex but not even head-normalizing, is presented as Ex. 4.1 in [22].

In this section we will prove a weaker version of the Standardization Theorem which does hold for infinite reductions. First, we need to extend the result of Lemma 5.3.11(ii) to the infinite notion of permutation equivalence.

Lemma 5.7.2. *Let $\mathcal{R}$ be a (finite or infinite) reduction. For all n it holds that $\mathit{Std}^n_{\mathrm{S}}(\mathcal{R}) \approx_\infty \mathcal{R}$.*

Proof. Follows from the fact that the definition of $\mathit{Std}^n_{\mathrm{S}}(\mathcal{R})$ depends only on a finite prefix of $\mathcal{R}$. We can take this finite prefix and then apply Lemma5.3.11(ii). □

In order to weaken the Standardization Theorem, we can do one of three things: weaken the notion of equivalence, weaken the notion of standard, or drop the requirement that each reduction is actually equivalent to its standard form. Here, we choose the last option.

Definition 5.7.3.

(i) A standard reduction $\mathcal{S}$ is a *standard part* of a reduction $\mathcal{R}$, if for all finite $\mathcal{S}'$ such that $\mathcal{S}' \sqsubseteq \mathcal{S}$, there exists some $\mathcal{S}''$ such that $\mathcal{S}' ; \mathcal{S}'' \approx_\infty \mathcal{R}$.

(ii) A *standard approximation* of $\mathcal{R}$ is a $\sqsubseteq$-maximal element of the set of standard parts of $\mathcal{R}$.

Theorem 5.7.4. *Every reduction has a standard approximation.*

Proof. What we prove is that, in fact, $Std_{\mathrm{S}}(\mathcal{R})$ is a standard approximation of $\mathcal{R}$. First, we have to prove that $Std_{\mathrm{S}}(\mathcal{R})$ is a standard part of $\mathcal{R}$. We already know from Theorem 5.3.15 that $Std_{\mathrm{S}}(\mathcal{R})$ is standard. Furthermore, by definition, the finite prefixes of $Std_{\mathrm{S}}(\mathcal{R})$ are of the form $Std^n_{\mathrm{S}}(\mathcal{R})[n]$. Thus they are prefixes of $Std^n_{\mathrm{S}}(\mathcal{R})$, and by Lem. 5.7.2, $Std^n_{\mathrm{S}}(\mathcal{R}) \approx_\infty \mathcal{R}$.

Now we need to show that $Std_{\mathrm{S}}(\mathcal{R})$ is a $\sqsubseteq$-maximal element of the set of standard parts of $\mathcal{R}$. Suppose $\mathcal{R}$ has a standard approximation $\mathcal{S}$ such that $Std_{\mathrm{S}}(\mathcal{R}) \sqsubseteq \mathcal{S}$. If $\mathcal{R}$ is infinite, then so is $Std_{\mathrm{S}}(\mathcal{R})$, and thus it must be the case that $\mathcal{S} = Std(\mathcal{R})$. If $\mathcal{R}$ is finite, then the result follows from the Standardization Theorem (Theorem 5.5.1). □

Although we have established the *existence* of a standard approximation for each reduction, we have not shown *uniqueness*. In fact, uniqueness does not hold, as is witnessed by the following counter example (again, a first-order example suffices):

Example 5.7.5. Consider the following TRS:

$$\mathsf{f}(x) \to \mathsf{f}(x)$$
$$\mathsf{a} \to \mathsf{a}$$

and the following non-terminating reduction (contracted redexes are underlined):

$$\mathcal{R} : \underline{\mathsf{f}}(\mathsf{a}) \to \mathsf{f}(\underline{\mathsf{a}}) \to \underline{\mathsf{f}}(\mathsf{a}) \to \mathsf{f}(\underline{\mathsf{a}}) \to \cdots$$

Both of the following reductions are standard approximations of $\mathcal{R}$:

$$\mathcal{R}_1 : \mathsf{f}(\underline{\mathsf{a}}) \to \mathsf{f}(\underline{\mathsf{a}}) \to \mathsf{f}(\underline{\mathsf{a}}) \to \mathsf{f}(\underline{\mathsf{a}}) \to \cdots$$
$$\mathcal{R}_2 : \underline{\mathsf{f}}(\mathsf{a}) \to \underline{\mathsf{f}}(\mathsf{a}) \to \underline{\mathsf{f}}(\mathsf{a}) \to \underline{\mathsf{f}}(\mathsf{a}) \to \cdots$$

Although the previous negative result invalidates standardization for formalizing equivalence of reductions, it has other applications. For example, in [24], an infinite standardization result is used to prove that the $\lambda\mathrm{x}^-$-calculus [7] (a λ-calculus similar to the λx-calculus defined in Sect. 4.3.1) preserves strong normalization of the λ-calculus.

5.8 Related work

A standardization procedure for HRSs was described previously by Van Oostrom [40], however only sketchy. The inversion standardization procedure of the present chapter fills in a lot of details for Van Oostrom's procedure, among which detailed proofs. The selection standardization procedure was, to my knowledge, not previously described for HRSs. Standardization is a well-known property, and therefore standardization results have been obtained in many guises for many different rewriting paradigms. In this section, we compare our result to some of the related work. Comparable standardization results can

be divided in three categories: standardization results in other higher-order rewriting paradigms, abstract standardization results and standardization results which use similar methodology.

5.8.1 Standardization in other higher-order rewriting paradigms

HRSs are not the only higher-order rewriting paradigm. First, higher-order functions can be formulated in the λ-calculus. In fact, the first standardization results were obtained for the λ-calculus [10, 44]. Proving standardization for the λ-calculus is relatively easy, because it is non-overlapping and left-normal. Second, standardization theorems have also been proved for Combinatory Reduction Systems (CRSs) [25, 26], a class of rewrite systems which is very close to HRSs; they can be seen as 'second-order' HRSs, that is every variable represents either an object or a function operating on objects. The current work improves on those results in important aspects; see below.

1980: Klop. In his PhD dissertation, Klop [25] proves the Standardization Theorem for the λ-calculus. He devises two methods to produce a standard reduction for a given reduction: *weak standardization*, which works by finding the step contracting the left-most redex, moving it to the front of the reduction and then recursively applying the procedure to the rest of the reduction; and *strong standardization*, which works by permuting steps which are in the wrong order until a normal form is reached. The idea of defining the second standardization procedure by a meta-rewrite system which operates on reductions is also due to Klop. Klop also extends his standardization result to left-normal, orthogonal CRSs. In contrast, the present result applies to all (local) HRSs: left-normality and orthogonality are not assumed.

2000–: Wells and Muller. In their working paper [55], Wells & Muller prove a standardization result for CRSs. They use a variant of the strong standardization procedure of Klop. Their notion of "standard reduction", however, does not take into account the fact that anti-standard steps do not need to be directly adjacent to each other. As a result, it is not the case that each class of equivalent reductions contains at most one standard reduction. In contrast, standard reductions in this chapter are unique in their equivalence class.

5.8.2 Abstract standardization results

Investigating standardization from an abstract, axiomatic point of view is an interesting area of research because it uncovers general, syntax-independent principles of standardization. Theoretically, abstract standardization results apply to any rewriting paradigm for which it can been shown that the axioms hold. In practice, however, proving that the axioms hold is not much easier than proving the standardization theorem directly. Also, abstract standardization

results typically provide standardization algorithms on a more abstract level, making them much harder to implement for specific rewriting paradigms such as HRSs.

1992: Gonthier, Lévy and Melliès. In [16] an abstract standardization theorem is proved from 5 axioms. The authors restrict their attention to orthogonal rewrite systems, and prove that a unique (up to a "square equivalence", which corresponds to our $\approx_{\mathrm{par}}$) standard reduction exists in each permutation equivalence class of reductions, by giving a standardization procedure which corresponds to Klop's weak standardization. The axioms in the paper are true for orthogonal HRSs.

2005: Melliès. Melliès [33] extends the result of [16] to potentially non-orthogonal rewrite systems. Additionally, he gives an axiomatization of "2-dimensional transition systems", which are comparable to what we call "meta-rewrite systems", and a method to obtain such a 2-dimensional transition system when given an (abstract) rewrite system.

Melliès' axioms for 1-dimensional transition systems do hold for HRSs, but his axioms for 2-dimensional transition systems do not apply directly to the meta-rewrite system for inversion standardization. Already the first axiom, which puts conditions on the shape of the meta-rewrite rules, does not hold! The 'problem' seems to be that we use multisteps, and Melliès' axiomatization presupposes proper steps.

5.8.3 Standardization results with similar methodology

2002/2003: Van Oostrom and De Vrijer. Van Oostrom & De Vrijer [42, 43] derive standardization results for first-order TRSs. In [42] only a standardization procedure based on Klop's strong standardization is investigated, up to permutation of parallel steps. In [43] both weak and strong standardization procedures are defined.

In both works, the strong standardization procedure is defined by a meta-rewrite system on proof terms. This is comparable to the approach followed in this chapter. There are some notable differences, however. First, Van Oostrom & De Vrijer use proof terms which include a composition operator, which may be nested within a function symbol. Reductions can be represented by a single proof term. Second, as a consequence, their meta-rewrite system is a rewrite system on proof terms, rather than a meta-rewrite system on sequences of proof terms, like $\Rightarrow_{\mathrm{Std}}$ in this chapter.

It is shown in Sect. 2.4.2 that this approach is not viable for higher-order rewriting, because the possibility of nesting composition operators conflicts with the presence of bound variables. To recuperate, in a sense, Van Oostrom & De Vrijer's proof terms are a bit more complex than ours, while their meta-rewrite system is easier to state.

5.9 Discussion

In this chapter we defined the notion of standard reduction and gave two standardization procedures which find, given an arbitrary (finite) reduction find a permutation equivalent standard one. Both procedures produce the same output for finite reductions. Still, there are some notable differences:

- Inversion standardization is only defined on finite reductions. Selection Standardization, on the other hand, is well-defined on infinite reductions. The notion of permutation equivalence, however, is not trivially extended to infinite reductions, and therefore the Standardization Theorem as such does not hold. We spent some time discussing a possible solution in Sect. 5.7.
- Selection Standardization is, in some sense, a particular strategy for Inversion Standardization, viz. the strategy that iteratively swaps the left-most step of the reduction to the start of the reduction. It is interesting to further investigate possible other strategies and the differences between them and Selection Standardization with respect to, for example, efficiency.

Six

Residuals

6.1 Introduction

In this chapter we study residuals in HRSs, with the ultimate goal of developing a third notion of equivalence of reductions besides permutation and standardization equivalence (but other applications of residual theory will also be briefly considered).

In general, two approaches to studying residuals can be identified. The first focusses on specific redex occurrences and traces them along a reduction, much in the same way as that positions were traced in Chapter 2. The second approach takes into account entire reductions, and tries to give an answer the following question: "What remains of a reduction after another reduction starting from the same object has been performed?" Since the second approach does not require the notion of "redex occurrence" to be formalized, it is more suitable for building up the theory of residuals from an abstract, axiomatic point of view towards a more applied one. For this reason, in this chapter we take the second approach to investigating residuals.

The idea of the approach of this chapter is the following. Let $\mathcal{R}$ and $\mathcal{S}$ be reductions. Intuitively, the residual of $\mathcal{R}$ after $\mathcal{S}$, written $\mathcal{R} \,/\, \mathcal{S}$, should consist of exactly those steps of $\mathcal{R}$ which are not in $\mathcal{S}$. As a first example, consider the (first-order) TRS consisting of the rules:

$$\begin{aligned} \mathsf{d}(x) &\to \mathsf{f}(x, x) \\ \mathsf{a} &\to \mathsf{b} \end{aligned}$$

and consider the following reductions:

$$\begin{aligned} \mathcal{R}\colon \mathsf{d}(\mathsf{a}) &\to \mathsf{f}(\mathsf{a}, \mathsf{a}) \to \mathsf{f}(\mathsf{b}, \mathsf{a}) \to \mathsf{f}(\mathsf{b}, \mathsf{b}) \\ \mathcal{S}\colon \mathsf{d}(\mathsf{a}) &\to \mathsf{f}(\mathsf{a}, \mathsf{a}) \to \mathsf{f}(\mathsf{a}, \mathsf{b}) \end{aligned}$$

The first step of both reductions is the same. Intuitively, the second step of $\mathcal{S}$ contracts the redex in the second argument of f, and thus corresponds to the third step of $\mathcal{R}$ (but note the steps are *not* equal, because their source is not

the same). The second step of $\mathcal{R}$ does not correspond to any step in $\mathcal{S}$, and so $\mathcal{R} / \mathcal{S}$ should consist of a step which corresponds to the second step of $\mathcal{R}$: $\mathcal{R} / \mathcal{S} : \mathsf{f}(\mathsf{b}, \mathsf{a}) \to \mathsf{f}(\mathsf{b}, \mathsf{b})$.

Even in first-order term rewriting, calculating residuals is a non-trivial task, for two reasons. First, it is conceptually non-trivial to define what residuals are. Even in a relatively simple rewriting paradigm as string rewriting, contracted redexes may be replaced by strings which are strictly longer, causing other redexes to be 'pushed away'.

Second, performing a step may duplicate and erase the redexes of other steps. For example, suppose we add the following rule to the rewrite system above:

$$\mathsf{b} \to \mathsf{c}.$$

Consider the following two reductions:

$$\begin{aligned} &\mathcal{T} \colon \mathsf{d}(\mathsf{a}) \to \mathsf{d}(\mathsf{b}) \to \mathsf{d}(\mathsf{c}) \to \mathsf{f}(\mathsf{b}, \mathsf{b}) \\ &\mathcal{U} \colon \mathsf{d}(\mathsf{a}) \to \mathsf{f}(\mathsf{a}, \mathsf{a}) \end{aligned}$$

Intuitively, we have

$$\mathcal{T} / \mathcal{U} : \mathsf{f}(\mathsf{a}, \mathsf{a}) \to \mathsf{f}(\mathsf{b}, \mathsf{a}) \to \mathsf{f}(\mathsf{b}, \mathsf{b}) \to \mathsf{f}(\mathsf{c}, \mathsf{b}) \to \mathsf{f}(\mathsf{c}, \mathsf{c}).$$

Although, intuitively, $\mathcal{T} / \mathcal{U}$ 'does less work' than $\mathcal{T}$, the reduction $\mathcal{T} / \mathcal{U}$ contains more steps than $\mathcal{T}$.

In higher-order rewriting, the problems caused by duplication are more severe: now, copies of the same redex may get nested. Consider the orthogonal HRS which consists of the following two rules:

$$\begin{aligned} \mu : &\quad \mathsf{mu}(\lambda x.z(x)) \to z(\mathsf{mu}(\lambda x.z(x))) \\ \delta : &\quad \mathsf{d}(x) \to \mathsf{f}(x, x) \end{aligned}$$

Consider the term $s = \mathsf{mu}(\lambda x.\mathsf{d}(x))$. The rule μ can be applied to the whole term and the rule ρ can be applied to the subterm $\mathsf{d}(x)$, so the following steps exist from s:

$$\begin{aligned} \varphi : &\quad \mathsf{mu}(\lambda x.\mathsf{d}(x)) \to \mathsf{d}(\mathsf{mu}(\lambda x.\mathsf{d}(x))) \\ \psi : &\quad \mathsf{mu}(\lambda x.\mathsf{d}(x)) \to \mathsf{mu}(\lambda x.\mathsf{f}(x, x)) \end{aligned}$$

The residual of ψ after φ is the reduction:

$$\begin{aligned} &\mathsf{d}(\mathsf{mu}(\lambda x.\mathsf{d}(x))) \\ \to &\quad \mathsf{f}(\mathsf{mu}(\lambda x.\mathsf{d}(x)), \mathsf{mu}(\lambda x.\mathsf{d}(x)) \\ \to^* &\quad \mathsf{f}(\mathsf{mu}(\lambda x.\mathsf{f}(x, x)), \mathsf{mu}(\lambda x.\mathsf{f}(x, x))) \end{aligned}$$

in which we see that one copy of the ρ-redex duplicates another (nested) copy of the ρ-redex.

This chapter is concerned with defining a residual operation for higher-order multisteps. The outline is as follows. First, we introduce residual

theory in an abstract, axiomatic way. Then we develop a residual operation for higher-order reductions which satisfies the axioms of the abstract theory. Finally, we present two important applications of the results of this chapter: we give an alternative proof of confluence of orthogonal HRSs and we define a notion of equivalence of reductions.

Remark. The material in this chapter is based on [8].

6.2 Abstract residual theory

Residual theory is studied in, among others, [18, 19, 23, 29, 32]. In this section, we present residuals in an abstract, axiomatic setting, partly following [53, 42], which is, in turn, based on [51]. The plan for this section is as follows: first, in Sect. 6.2.1, we present axioms, which must be satisfied by a residual operator on *steps* of a certain rewrite system; then, in Sect. 6.2.2, we give a general way of extending residual operators for steps to residual operators for reductions.

6.2.1 Residual systems

A residual system is a rewrite system with an additional residual operator and a function mapping each term to an empty step from this term.

Definition 6.2.1. A *residual system* is specified by a triple $\langle \mathfrak{R}, 1, / \rangle$ where:

- $\mathfrak{R}$ is an (abstract) rewriting system;
- 1 is a function from objects (of $\mathfrak{R}$) to steps, such that $\mathrm{src}(1_a) = a = \mathrm{tgt}(1_a)$; and
- /, the projection function, is a (total) function from pairs of coinitial steps to steps, with $\mathrm{src}(\varphi/\psi) = \mathrm{tgt}(\psi)$ and $\mathrm{tgt}(\varphi/\psi) = \mathrm{tgt}(\psi/\varphi)$;

such that the following residual laws hold:

$$\begin{aligned} 1_a \,/\, \varphi &= 1_b \\ \varphi \,/\, 1_a &= \varphi \\ \varphi \,/\, \varphi &= 1_b \\ (\varphi \,/\, \psi) \,/ (\chi \,/\, \psi) &= (\varphi \,/\, \chi) \,/ (\psi \,/\, \chi) \end{aligned}$$

where, in all cases above, $a = \mathrm{src}(\varphi)$ and $b = \mathrm{tgt}(\varphi)$.

The result of projecting φ over ψ (that is φ/ψ) is called the *residual* of φ after ψ. Steps in the range of 1 are called *empty steps*. We will, in practice, omit the argument of 1, and just write 1 for any empty step; usually, the intended one is determined by the context.

In Fig. 6.1 we can see visually depicted the conditions to the source and target of residuals: if φ, ψ are two coinitial[1] steps, say from a to b and from a to c, respectively, then there exist steps $\varphi \,/\, \psi$ and $\psi \,/\, \psi$ from c to d and from b to d, respectively. This basic diagram can be used to build more complex diagrams: by 'iteratively' drawing it, diagrams can be formed which give intuitive insight into the meaning of the axioms. The fourth identity in the definition above, for example, which is called the *cube identity* and is depicted in Fig. 6.2, expresses that the order in which the projections are resolved, does not matter.

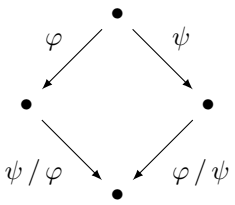


Figure 6.1: Conditions to sources and targets of residuals.

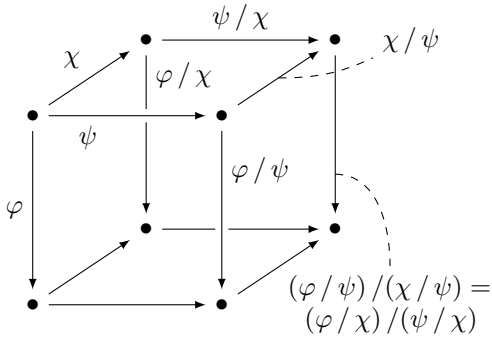


Figure 6.2: Cube axiom.

From Fig. 6.1, we easily obtain the following important theorem:

Theorem 6.2.2. *If $\langle \mathfrak{R}, 1, / \rangle$ is a residual system, then $\mathfrak{R}$ has the diamond property.*

Proof. By Fig. 6.1: let φ be a step from a to b and ψ a step from a to c. Then $\psi \,/\, \varphi$ is a step from b to some d and φ/ψ a step from c to the same object d. □

Note, that the diamond property does not hold for every rewrite system. So, not every rewrite system can be used as the basis of a residual system. However, the diamond property does hold for orthogonal HRSs (and TRSs), if we allow multisteps, as will be shown later.

[1] In later sections we will restrict this even further, to coinitial and *compatible* steps. Intuitively the notion of compatibility expresses that two steps can be performed independently of each other. Later, it will be proved that it corresponds exactly to the notion of orthogonality.

6.2.2 Residuals of reductions

Above, residuals of steps are defined. We also want to define residuals for reductions. For this, we use the notion of Abstract Rewriting System with Composition (as defined in Def. 2.2.5).

Definition 6.2.3. A *residual system with composition* (RSC) is a triple $\langle \mathfrak{R}, 1, / \rangle$, such that:

- $\mathfrak{R}$ is an ARS with composition,
- $\langle \mathfrak{R}, 1, / \rangle$ is a residual system, and
- for each step χ and composable steps φ, ψ with $\mathrm{src}(\varphi) = \mathrm{src}(\chi)$, it holds that:

$$\chi \,/\, (\varphi \,;\, \psi) = (\chi \,/\, \varphi) \,/\, \psi$$
$$(\varphi \,;\, \psi) \,/\, \chi = (\varphi \,/\, \chi) \,;\, (\psi \,/\, (\chi \,;\, \varphi)))$$

Again by using the diagram of Fig. 6.1 as the basic building block, we can construct the diagram of Fig. 6.3, which depicts the identities of the definition above.

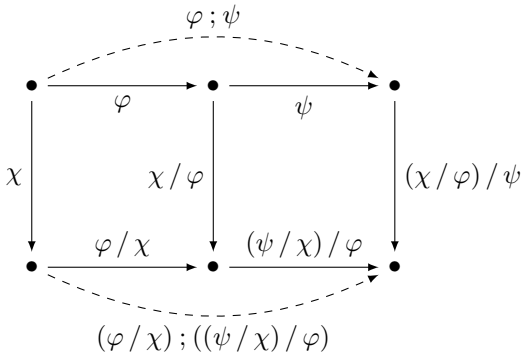


Figure 6.3: Axioms of residual systems with composition

Now we de define a canonical way to transform a residual system to a residual system with compositions. Let an (abstract) rewriting system $\mathfrak{R} = \langle A, \Phi, \mathrm{src}, \mathrm{tgt} \rangle$ and a residual system $\mathbf{R} = \langle \mathfrak{R}, 1, / \rangle$ be given. We define the *reflexive, transitive closure* of $\mathbf{R}$ by: $\mathbf{R}^* = \langle \mathfrak{R}^*, 1^*, /^*, ; \rangle$, where:

- $\mathfrak{R}^*$ is the reflexive, transitive closure of $\mathfrak{R}$, as defined in Def. 2.2.5;
- 1^* is the function which maps each object a to the empty reduction from a to a; and

- $/^*$ is defined in terms $/$ by the following (meta-)rewriting system $\Rightarrow_{\mathrm{R}}$:

$$1 \Rightarrow \varepsilon \qquad \text{(unit)}$$
$$\varphi \,/^*\, \psi \Rightarrow \varphi \,/\, \psi \qquad \text{(step)}$$
$$\mathcal{R} \,/^*\, (\mathcal{S}_1 \,;\, \mathcal{S}_2) \Rightarrow (\mathcal{R} \,/^*\, \mathcal{S}_1) \,/^*\, \mathcal{S}_2 \qquad \text{(comp-r)}$$
$$(\mathcal{R}_1 \,;\, \mathcal{R}_2) \,/^*\, \mathcal{S} \Rightarrow (\mathcal{R}_1 \,/^*\, \mathcal{S}) \,;\, (\mathcal{R}_2 \,/^*\, (\mathcal{S} \,/^*\, \mathcal{R}_1)) \qquad \text{(comp-l)}$$

  where $\mathcal{R}, \mathcal{S}$ (possibly with subscript) denote arbitrary reductions (sequences of steps), while φ, ψ denote arbitrary steps (reductions of length 1).

Note that the symbol 1 in the first meta-rewrite rule, by convention, denotes any object in the range of the function 1. In this case, it is a single, empty *step*. Also observe that the $/$ symbol in the second rewrite rule of the meta-rewrite system is already defined in the residual system **R**.

Example 6.2.4. Let $\mathcal{R}$ be the two-step reduction φ_1, φ_2 and $\mathcal{S}$ the one-step reduction ψ. The residual of $\mathcal{S}$ after $\mathcal{R}$ is:

$$\begin{aligned} &(\varphi_1 \,;\, \varphi_2) \,/^*\, \psi \\ &\Rightarrow_{(\text{comp-l})} \; (\varphi_1 \,/^*\, \psi) \,;\, (\varphi_2 \,/^*\, (\psi \,/^*\, \varphi_1) \\ &\Rightarrow^*_{(\text{step})} \quad (\varphi_1 \,/\, \psi) \,;\, (\varphi_2 \,/\, (\psi \,/\, \varphi_1) \end{aligned}$$

The final term is defined by the original residual system.

In Ex. 6.3.3 an example of the use of the above rewrite system together with a previously defined residual operation on steps is given.

Proposition 6.2.5. *The operator $/^*$ is well-defined, that is the rewrite relation $\Rightarrow_{\mathrm{R}}$ is confluent and terminating.*

Proof. We use standard first-order rewriting theory in the course of this proof. We begin by proving termination. This can be proved by semantic labelling [57], cf. [53, Ex. 6.5.43]. As quasi-model for the rewrite system we take the strictly positive integers with

$$|\varphi| = 1 \quad |x \,;\, y| = x + y \quad \text{and} \quad |x \,/^*\, y| = x.$$

We only label the symbol $/^*$, by the sum of the values of its arguments. The resulting labelling is proved terminating with the recursive path ordering, using the following precedence on the function symbols:

$$\underset{i}{/^*} > \underset{j}{/^*} > \,; > \varphi$$

for all i, j such that $i > j$.

Given termination, by Newman's Lemma, confluence follows from local confluence, which in turn follows from the fact that there is one critical pair,

which can be be joined:

$$
\begin{aligned}
&(\mathcal{R}_1 \,;\, \mathcal{R}_2) \,/^*\, (\mathcal{S}_2 \,;\, \mathcal{S}_2) \\
&\Rightarrow_{\mathrm{R}} ((\mathcal{R}_1 \,;\, \mathcal{R}_2) \,/^*\, \mathcal{S}_1) \,/^*\, \mathcal{S}_2 \\
&\Rightarrow_{\mathrm{R}} ((\mathcal{R}_1 \,/^*\, \mathcal{S}_1) \,;\, (\mathcal{R}_2 \,/^*\, (\mathcal{S}_1 \,/^*\, \mathcal{R}_1))) \,/^*\, \mathcal{S}_2 \\
&\Rightarrow_{\mathrm{R}} ((\mathcal{R}_1 \,/^*\, \mathcal{S}_1) \,/^*\, \mathcal{S}_2) \,;\, ((\mathcal{R}_2 \,/^*\, (\mathcal{S}_1 \,/^*\, \mathcal{R}_1)) \,/^*\, (\mathcal{S}_2 \,/^*\, (\mathcal{R}_1 \,/^*\, \mathcal{S}_1)))
\end{aligned}
$$

$$
\begin{aligned}
&(\mathcal{R}_1 \,;\, \mathcal{R}_2) \,/^*\, (\mathcal{S}_2 \,;\, \mathcal{S}_2) \\
&\Rightarrow_{\mathrm{R}} (\mathcal{R}_1 \,/^*\, (\mathcal{S}_1 \,;\, \mathcal{S}_2)) \,;\, (\mathcal{R}_2 \,/^*\, ((\mathcal{S}_1 \,;\, \mathcal{S}_2) \,/^*\, \mathcal{R}_1)) \\
&\Rightarrow_{\mathrm{R}} ((\mathcal{R}_1 \,/^*\, \mathcal{S}_1) \,/^*\, \mathcal{S}_2) \,;\, (\mathcal{R}_2 \,/^*\, ((\mathcal{S}_1 \,;\, \mathcal{S}_2) \,/^*\, \mathcal{R}_1)) \\
&\Rightarrow_{\mathrm{R}} ((\mathcal{R}_1 \,/^*\, \mathcal{S}_1) \,/^*\, \mathcal{S}_2) \,;\, ((\mathcal{R}_2 \,/^*\, (\mathcal{S}_1 \,/^*\, \mathcal{R}_1)) \,/^*\, (\mathcal{S}_2 \,/^*\, (\mathcal{R}_1 \,/^*\, \mathcal{S}_1)))
\end{aligned}
$$

This proves confluence. □

Proposition 6.2.6. *For any residual system* $\mathbf{R}$*, its reflexive, transitive closure* $\mathbf{R}^*$ *is a residual system with composition.*

Proof. The proof boils down to showing that the identities of Def. 6.2.1 and Def. 6.2.3 hold, that is, for all reductions $\mathcal{R}, \mathcal{S}, \mathcal{T}$:

$$
\begin{aligned}
1 \,/^*\, \mathcal{R} &\Leftrightarrow^* 1 \\
\mathcal{R} \,/^*\, 1 &\Leftrightarrow^* \mathcal{R} \\
\mathcal{R} \,/^*\, \mathcal{R} &\Leftrightarrow^* 1 \\
(\mathcal{R} \,/^*\, \mathcal{S}) \,/^*\, (\mathcal{T} \,/^*\, \mathcal{S}) &\Leftrightarrow^* (\mathcal{R} \,/^*\, \mathcal{T}) \,/^*\, (\mathcal{S} \,/^*\, \mathcal{T})
\end{aligned}
$$

Let $\mathcal{R}$ be a reduction. The three unit identities of Def. 6.2.1 are easily proved by induction on the length of $\mathcal{R}$. If the length of $\mathcal{R}$ is 1, then the result follows easily because then $/^*$ is defined solely in terms of $/$. Otherwise, $\mathcal{R} = \mathcal{R}_1 \,;\, \mathcal{R}_2$. Then:

(i) $1 \,/^*\, (\mathcal{R}_1 \,;\, \mathcal{R}_2) \Rightarrow (1 \,/^*\, \mathcal{R}_1) \,/^*\, \mathcal{R}_2 \Rightarrow^*_{\mathrm{IH}} 1 \,/^*\, \mathcal{R}_2 \Rightarrow^*_{\mathrm{IH}} 1$

(ii)
$$
\begin{aligned}
&(\mathcal{R}_1 \,;\, \mathcal{R}_2) \,/^*\, 1 \\
\Rightarrow\quad &(\mathcal{R}_1 \,/^*\, 1) \,;\, (\mathcal{R}_2 \,;\, (1 \,/^*\, \mathcal{R}_2)) \\
\Rightarrow^*_{\mathrm{IH}}\quad &\mathcal{R}_1 \,;\, (\mathcal{R}_2 \,;\, (1 \,/^*\, \mathcal{R}_2)) \\
\Rightarrow^*_{(\mathrm{i})}\quad &\mathcal{R}_1 \,;\, \mathcal{R}_2
\end{aligned}
$$

(iii)
$$
\begin{aligned}
&(\mathcal{R}_1 \,;\, \mathcal{R}_2) \,/^*\, (\mathcal{R}_1 \,;\, \mathcal{R}_2) \\
\Rightarrow\quad &((\mathcal{R}_1 \,;\, \mathcal{R}_2) \,/^*\, \mathcal{R}_1) \,/^*\, \mathcal{R}_2 \\
\Rightarrow\quad &((\mathcal{R}_1 \,/^*\, \mathcal{R}_1) \,;\, (\mathcal{R}_2 \,/^*\, (\mathcal{R}_1 \,/^*\, \mathcal{R}_1))) \,/^*\, \mathcal{R}_2 \\
\Rightarrow^*_{\mathrm{IH}}\quad &(1 \,;\, (\mathcal{R}_2 \,/^*\, 1)) \,/^*\, \mathcal{R}_2 \\
\Rightarrow^*_{(\mathrm{ii})}\quad &\mathcal{R}_2 \,/^*\, \mathcal{R}_2 \\
\Rightarrow_{\mathrm{IH}}\quad &1
\end{aligned}
$$

as required. In order to show that the fourth identity also holds, we use the *layered size* $|\cdot|$, as defined in [53]:

$$\begin{aligned} |\varphi| &= 1 \\ |\mathcal{R}\,;\mathcal{S}| &= |\mathcal{R}| + 1 + |\mathcal{S}| \\ |\mathcal{R}\,/^*\,\mathcal{S}| &= |\mathcal{R}| \end{aligned}$$

where $\mathcal{R}, \mathcal{S}$ are reductions and φ is a step (reduction of length 1). Now we can prove the fourth identity by induction on the sum of the layered sizes of $\mathcal{R}, \mathcal{S}, \mathcal{T}$. Because $\Rightarrow_{\mathrm{R}}$ is confluent and terminating, we may assume that all three are reductions and contain no $/^*$-symbols. If all of $\mathcal{R}, \mathcal{S}, \mathcal{T}$ are steps, the result follows trivially because then $/^*$ equals $/$ by definition. Otherwise, one of $\mathcal{R}, \mathcal{S}, \mathcal{T}$ must contain at least two steps.

- If $\mathcal{R} = \mathcal{R}_1\,;\mathcal{R}_2$, then we do:

$$\begin{aligned} & \underline{((\mathcal{R}_1\,;\mathcal{R}_2)\,/^*\,\mathcal{S})}\,/^*(\mathcal{T}\,/^*\,\mathcal{S}) \\ \Rightarrow\quad & \underline{((\mathcal{R}_1/^*\mathcal{S})\,;(\mathcal{R}_2/^*(\mathcal{S}/^*\mathcal{R}_1)))\,/^*(\mathcal{T}\,/^*\,\mathcal{S})} \\ \Rightarrow\quad & \underline{(\mathcal{R}_1/^*\mathcal{S})\,/^*(\mathcal{T}\,/^*\,\mathcal{S})}\,;((\mathcal{R}_2/^*(\mathcal{S}/^*\mathcal{R}_1))\,/^*\,\underline{((\mathcal{T}\,/^*\,\mathcal{S})\,/^*(\mathcal{R}_1/^*\mathcal{S}))}) \\ \Leftrightarrow^*_{\mathrm{IH}}\ & (\mathcal{R}_1/^*\mathcal{T})\,/^*(\mathcal{S}/^*\mathcal{T})\,;\underline{((\mathcal{R}_2/^*(\mathcal{S}/^*\mathcal{R}_1))\,/^*((\mathcal{T}/^*\mathcal{R}_1)/^*(\mathcal{S}/^*\mathcal{R}_1)))} \\ \Leftrightarrow^*_{\mathrm{IH}}\ & \underline{(\mathcal{R}_1/^*\mathcal{T})\,/^*(\mathcal{S}/^*\mathcal{T})\,;((\mathcal{R}_2/^*(\mathcal{T}/^*\mathcal{R}_1))\,/^*((\mathcal{S}/^*\mathcal{R}_1)/^*(\mathcal{T}/^*\mathcal{R}_1)))} \\ \Leftarrow\quad & \underline{((\mathcal{R}_1/^*\mathcal{T})\,;(\mathcal{R}_2/^*(\mathcal{T}/^*\mathcal{R}_1)))}\,/^*(\mathcal{S}/^*\mathcal{T}) \\ \Leftarrow\quad & ((\mathcal{R}_1\,;\mathcal{R}_2)/^*\mathcal{T})\,/^*(\mathcal{S}/^*\mathcal{T}) \end{aligned}$$

- If $\mathcal{S} = \mathcal{S}_1\,;\mathcal{S}_2$, then we do:

$$\begin{aligned} & \underline{(\mathcal{R}/^*(\mathcal{S}_1\,;\mathcal{S}_2))}\,/^*\,\underline{(\mathcal{T}\,/^*(\mathcal{S}_1\,;\mathcal{S}_2))} \\ \Rightarrow^*\ & \underline{((\mathcal{R}/^*\mathcal{S}_1)/^*\mathcal{S}_2)\,/^*((\mathcal{T}/^*\mathcal{S}_1)/^*\mathcal{S}_2)} \\ \Leftrightarrow^*_{\mathrm{IH}}\ & \underline{((\mathcal{R}/^*\mathcal{S}_1)\,/^*(\mathcal{T}/^*\mathcal{S}_1))}\,/^*(\mathcal{S}_2/^*(\mathcal{T}/^*\mathcal{S}_1)) \\ \Leftrightarrow^*_{\mathrm{IH}}\ & \underline{((\mathcal{R}/^*\mathcal{T})/^*(\mathcal{S}_1/^*\mathcal{T}))\,/^*(\mathcal{S}_2/^*(\mathcal{T}/^*\mathcal{S}_1))} \\ \Leftarrow\quad & (\mathcal{R}/^*\mathcal{T})\,/^*\,\underline{(\mathcal{S}_1/^*\mathcal{T}\,;\mathcal{S}_2/^*(\mathcal{T}/^*\mathcal{S}_1))} \\ \Leftarrow\quad & (\mathcal{R}/^*\mathcal{T})\,/^*((\mathcal{S}_1\,;\mathcal{S}_2)/^*\mathcal{T}) \end{aligned}$$

- The case that $\mathcal{T} = \mathcal{T}_1\,;\mathcal{T}_2$ is the inverse of the previous case. □

Theorem 6.2.7. *If $\langle \mathfrak{R}, 1, / \rangle$ is a residual system, then $\mathfrak{R}$ is confluent.*

Proof. Let $\mathbf{R} = \langle \mathfrak{R}, 1, / \rangle$. By Prop. 6.2.6, $\mathbf{R}^*$ is a residual system, and thus $\mathfrak{R}^*$ has, by Theorem 6.2.2, the diamond property. Therefore, by Lemma 2.2.11, $\mathfrak{R}$ is confluent. □

In the following, we will often just write $/$ for $/^*$, except when the distinction is important. Usually, however, the distinction does not make a difference: the second rule of the rewrite system $\Rightarrow_{\mathrm{R}}$ (see pag. 114) makes sure that $/$ and $/^*$ behave the same for steps, and $/$ is not defined for reductions.

6.3 Residuals for higher-order multisteps

In this section we will develop a residual operation for higher-order multisteps. In the first subsection we will define projection terms as terms which contain so-called projection operators, and in the rest of the section we will associate to each projection term a unique term, in such a way that the projection operator calculates a projection operation which satisfies all the axioms of the previous section.

6.3.1 Projection terms

We want to define the projection operation by syntactic means. To do so, we add, for each type α, a new function symbol to the language of proof terms, the projection operator:

$$/_\alpha : \alpha \to \alpha \to \alpha.$$

Terms over this extended signature will be called *projection terms*.[2] Note that the arity of $/_\alpha$ depends on the type α: $\mathrm{ar}(/_\alpha) = 2 + \mathrm{ar}(\alpha)$. We will write

$$(t_1 /_\alpha t_2)(t_3, \ldots, t_n) \quad \text{for} \quad /_\alpha(t_1, t_2, \ldots, t_n).$$

Usually, $t_3, \ldots, t_n$ are just bound variables. If this is the case, we will often refrain from writing the term in $\beta\overline{\eta}$-normal form; instead, we write, for example, $(\lambda x.\varphi) /_\alpha (\lambda x.\psi)$, rather than its η-expansion $\lambda z./_\alpha(\lambda x.\varphi, \lambda x.\psi, z)$. This means the function symbol $/_\alpha$ will usually be written with only two arguments. We make use of this fact and write it as a binary, infix symbol. In the following, the subscript α of $/_\alpha$ will usually be omitted.

The goal of the next subsections is to associate to each projection term a unique proof term (that is a projection term containing no projection operators) in such a way that the residual laws are satisfied.

6.3.2 A first attempt

In [53, Sect. 8.7], such unique projection terms are calculated (for first-order rewriting) by means of a TRS on projection terms: given a projection term φ, typically of the form φ_1 / φ_2, the normal form ψ represents the proof term for the residual of φ_1 after φ_2. Using a TRS has the advantage that standard rewriting theory can be easily applied. We have used meta-rewriting systems to transform reductions in earlier chapters for the same reason.

However, in higher-order rewriting it is problematic to use an HRS on projection terms to calculate residuals, due to nesting. The naive extension of the system in [53] to the higher-order case is the following HRS on projection

[2] Projection terms are called *slash-dot terms* in [8].

terms:

$$\begin{aligned}
x(\varphi_1,\ldots,\varphi_n) \,/\, x(\psi_1,\ldots,\psi_n) &\Rightarrow x(\varphi_1 \,/\, \psi_1,\ldots,\varphi_n \,/\, \psi_n)\\
f(\varphi_1,\ldots,\varphi_n) \,/\, f(\psi_1,\ldots,\psi_n) &\Rightarrow f(\varphi_1 \,/\, \psi_1,\ldots,\varphi_n \,/\, \psi_n)\\
\rho(\varphi_1,\ldots,\varphi_n) \,/\, \rho(\psi_1,\ldots,\psi_n) &\Rightarrow r(\varphi_1 \,/\, \psi_1,\ldots,\varphi_n \,/\, \psi_n)\\
\rho(\varphi_1,\ldots,\varphi_n) \,/\, l(\psi_1,\ldots,\psi_n) &\Rightarrow \rho(\varphi_1 \,/\, \psi_1,\ldots,\varphi_n \,/\, \psi_n)\\
l(\varphi_1,\ldots,\varphi_n) \,/\, \rho(\psi_1,\ldots,\psi_n) &\Rightarrow r(\varphi_1 \,/\, \psi_1,\ldots,\varphi_n \,/\, \psi_n)\\
\lambda x.\varphi \,/\, \lambda x.\psi &\Rightarrow \lambda x.(\varphi(x) \,/\, \psi(x))
\end{aligned}$$

The first five rules are directly from [53]. The last rule moves the projection operator inside an abstraction.[3] The proposed HRS, however, does not correctly calculate residuals. Consider the HRS $\mathfrak{M}u$:

$$\begin{aligned}
\mu:&\quad \lambda z.\mathsf{mu}(\lambda x.z(x)) \to \lambda z.z(\mathsf{mu}(\lambda x.z(x)))\\
\rho:&\quad \lambda x.\mathsf{f}(x) \to \lambda x.\mathsf{g}(x)
\end{aligned}$$

and the two $\mathfrak{M}u$-steps:

$$\begin{aligned}
\mathsf{mu}(\lambda x.\rho(x)) &: \mathsf{mu}(\lambda x.\mathsf{f}(x)) \to \mathsf{mu}(\lambda x.\mathsf{g}(x))\\
\mu(\lambda x.\mathsf{f}(x)) &: \mathsf{mu}(\lambda x.\mathsf{f}(x)) \to \mathsf{f}(\mathsf{mu}(\lambda x.\mathsf{f}(x)))
\end{aligned}$$

Note that here mu is a function symbol, while μ is the name of the first rule of the HRS. Now we see that:

$$\begin{aligned}
&\mathsf{mu}(\lambda x.\rho(x)) \,/\, \mu(\lambda x.\mathsf{f}(x))\\
\Rightarrow\ &(\lambda x.\rho(x) \,/\, \lambda x.\mathsf{f}(x))(\mathsf{mu}(\lambda x.\rho(x) \,/\, \lambda x.\mathsf{f}(x)))\\
\Rightarrow\ &\rho(\mathsf{mu}(\lambda x.\rho(x) \,/\, \lambda x.\mathsf{f}(x))) \,/\, \mathsf{f}(\mathsf{mu}(\lambda x.\rho(x) \,/\, \lambda x.\mathsf{f}(x)))\\
\Rightarrow^*\ &\rho(\mathsf{mu}(\lambda x.\mathsf{f}(x)))
\end{aligned}$$

The final proof term is incorrect because it witnesses a (multi)step from $\mathsf{f}(\mathsf{mu}(\lambda x.(\mathsf{f}(x))))$ to $\mathsf{g}(\mathsf{mu}(\lambda x.(\mathsf{f}(x))))$, and thus this HRS on projection terms does not satisfy the residual laws. Inspection of the above $\Rightarrow$-reduction reveals that the problem is that instances of the same /-operator get nested, and then 'cancel each other out' because $\varphi \,/\, \varphi \Rightarrow^* 1$, while, in this case, we need $\varphi \,/\, \varphi \Rightarrow^* \varphi$. At first sight, the problem can be solved by introducing a new function symbol $\bot$, changing the sixth rule of the HRS to:

$$\lambda x.\varphi(x) \,/\, \lambda x.\psi(x) \Rightarrow \lambda x.(\varphi(\bot(x)) \,/\, \psi(\bot(x))).$$

and adding rules to make sure that $\bot(\varphi) \,/\, \bot(\varphi) \Rightarrow^* \varphi$, but this solution has an ad-hoc flavor to it, and may have other nesting-related problems. We will leave it to future research to investigate whether this approach offers an elegant solution to the problem posed above. Here, we will define the projection operation by another means.

[3] This rule is given in [8] as $(\lambda x.\varphi \,/\, \lambda x.\psi)z \Rightarrow \varphi(z) \,/\, \psi(z)$. This is the same rule, but transformed to $\beta\overline{\eta}$-normal form (and, following the convention, removing leading abstractions).

Residual rules:

$$\frac{\varphi_1 \,/\, \psi_1 \succcurlyeq \chi_1 \cdots \varphi_n \,/\, \psi_n \succcurlyeq \chi_n}{x(\varphi_1,\ldots,\varphi_n) \,/\, x(\psi_1,\ldots,\psi_n) \succcurlyeq x(\chi_1,\ldots,\chi_n)}\,\mathsf{R}_x$$

$$\frac{\varphi_1 \,/\, \psi_1 \succcurlyeq \chi_1 \cdots \varphi_n \,/\, \psi_n \succcurlyeq \chi_n}{f(\varphi_1,\ldots,\varphi_n) \,/\, f(\psi_1,\ldots,\psi_n) \succcurlyeq f(\chi_1,\ldots,\chi_n)}\,\mathsf{R}_f$$

$$\frac{\varphi_1 \,/\, \psi_1 \succcurlyeq \chi_1 \cdots \varphi_n \,/\, \psi_n \succcurlyeq \chi_n}{\rho(\varphi_1,\ldots,\varphi_n) \,/\, \rho(\psi_1,\ldots,\psi_n) \succcurlyeq r(\chi_1,\ldots,\chi_n)}\,\mathsf{R}_\rho$$

$$\frac{\varphi_1 \,/\, \psi_1 \succcurlyeq \chi_1 \cdots \varphi_n \,/\, \psi_n \succcurlyeq \chi_n}{\rho(\varphi_1,\ldots,\varphi_n) \,/\, l(\psi_1,\ldots,\psi_n) \succcurlyeq \rho(\chi_1,\ldots,\chi_n)}\,\mathsf{R}_\mathrm{R}$$

$$\frac{\varphi_1 \,/\, \psi_1 \succcurlyeq \chi_1 \cdots \varphi_n \,/\, \psi_n \succcurlyeq \chi_n}{l(\varphi_1,\ldots,\varphi_n) \,/\, \rho(\psi_1,\ldots,\psi_n) \succcurlyeq r(\chi_1,\ldots,\chi_n)}\,\mathsf{R}_\mathrm{L}$$

$$\frac{\varphi \,/\, \psi \succcurlyeq \chi}{\lambda x.\varphi \,/\, \lambda x.\psi}\,\mathsf{R}_\lambda \qquad \frac{\varphi \succcurlyeq \varphi' \quad \psi \succcurlyeq \psi' \quad \varphi' \,/\, \psi' \succcurlyeq \chi}{\varphi \,/\, \psi \succcurlyeq \chi}\,\mathsf{r}+\mathsf{t}_/$$

Replacement rules:

$$\frac{\varphi_1 \succcurlyeq \psi_1 \cdots \varphi_n \succcurlyeq \psi_n}{x(\varphi_1,\ldots,\varphi_n) \succcurlyeq x(\psi_1,\ldots,\psi_n)}\,\mathsf{repl}_x$$

$$\frac{\varphi_1 \succcurlyeq \psi_1 \cdots \varphi_n \succcurlyeq \psi_n}{f(\varphi_1,\ldots,\varphi_n) \succcurlyeq f(\psi_1,\ldots,\psi_n)}\,\mathsf{repl}_f$$

$$\frac{\varphi \succcurlyeq \psi}{\lambda x.\varphi \succcurlyeq \lambda x.\psi}\,\mathsf{repl}_\lambda \qquad \frac{\varphi_1 \succcurlyeq \psi_1 \cdots \varphi_n \succcurlyeq \psi_n}{\rho(\varphi_1,\ldots,\varphi_n) \succcurlyeq \rho(\psi_1,\ldots,\psi_n)}\,\mathsf{repl}_\rho$$

Table 6.1: Residual operator for multisteps: the inference system $\mathcal{R}es$.

6.3.3 Definition of the residual operator

We define the residual operation by defining a "simplification relation", by means of the inference system $\mathcal{R}es$, which maps each projection term to a proof term. Using an inferences system gives us more control of the 'rewriting' strategy and allows us to calculate subpropblems before they are duplicated and nested. The inference rules of $\mathcal{R}es$ are listed in Table 6.1 on page 119. The system proves judgements of the form $\varphi \succcurlyeq \psi$, where φ is a projection term and ψ a proof term. The most important rules of the system are the R_x, R_f, R_ρ, R_R, R_L and R_λ rules; they describe how the residual operator works on

projection terms of different forms; the replacement rules, repl_x, repl_f, repl_λ and repl_ρ, 'zoom in' on the interesting parts of a projection term; and finally, the $\mathsf{r} + \mathsf{t}_/$ rule combines replacement of the /-symbol and transitivity.

We write $\mathcal{R}es \vdash^{\mathcal{K}} \varphi \succcurlyeq \chi$, or just $\vdash^{\mathcal{K}} \varphi \succcurlyeq \chi$, to denote that $\mathcal{R}es$-inference $\mathcal{K}$ has $\varphi \succcurlyeq \chi$ as its final conclusion. We write $\vdash \varphi \succcurlyeq \chi$ or just $\varphi \succcurlyeq \chi$ if the inference is not important. If $\varphi \succcurlyeq \chi$, we say that projection term φ *simplifies* to χ.

Example 6.3.1. Consider the HRS $\mathfrak{Mu}$ and the steps

$$\mathsf{mu}(\lambda x.\rho(x)) \quad \text{and} \quad \mu(\lambda x.\mathsf{f}(x))$$

from page 118. With the inference system we obtain the correct residual of $\mathsf{mu}(\lambda x.\rho(x))$ after $\mu(\lambda x.\mathsf{f}(x))$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x \,/\, x \succcurlyeq x}\,\mathsf{R}_x}{\rho(x)/f(x) \succcurlyeq \rho(x)}\,\mathsf{R}_f}{\lambda x.\rho(x) \,/\, \lambda x.f(x) \succcurlyeq \lambda x.\rho(x)}\,\mathsf{R}_\lambda}{\mathsf{mu}(\lambda x.\rho(x)) \,/\, \mu(\lambda x.\mathsf{f}(x)) \succcurlyeq \rho(\mathsf{mu}(\lambda x.\rho(x)))}\,\mathsf{R}_\mathrm{L}$$

The result is indeed the desired proof term:

$$\rho(\mathsf{mu}(\lambda x.\rho(x))) : \mathsf{f}(\mathsf{mu}(\lambda x.\mathsf{f}(x))) \to \mathsf{g}(\mathsf{mu}(\lambda x.\mathsf{g}(x))).$$

The next example shows the raison d'être of the replacement rules and the $\mathsf{r} + \mathsf{t}_/$ rule: they allow for projection symbols nested inside a context and other projection symbols, respectively.

Example 6.3.2. Consider the HRS $\mathfrak{Mu}$ from page 118. Let the following projection term be given: $\mathsf{f}(\mu(\lambda x.\mathsf{f}(x)) \,/\, \mathsf{mu}(\lambda x.(\rho(x) \,/\, \mathsf{f}(x))))$. The following inference is used to calculate the proof term that the above projection term simplifies to:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x \succcurlyeq x}\,\mathsf{repl}_x}{\mathsf{f}(x) \succcurlyeq \mathsf{f}(x)}\,\mathsf{repl}_f \qquad \dfrac{}{x \,/\, x \succcurlyeq x}\,\mathsf{R}_x}{\rho(x) \,/\, \mathsf{f}(x) \succcurlyeq \rho(x)}\,\mathsf{R}_\mathrm{R} \qquad \dfrac{}{x \,/\, x \succcurlyeq x}\,\mathsf{R}_x}{\mathsf{f}(x) \,/\, \rho(x) \succcurlyeq \mathsf{f}(x)}\,\mathsf{R}_\mathrm{L}}{\mathsf{f}(x) \,/\, (\rho(x) \,/\, \mathsf{f}(x)) \succcurlyeq \mathsf{f}(x)}\,\mathsf{r} + \mathsf{t}_/}{\lambda x.\mathsf{f}(x) \,/\, \lambda x.(\rho(x) \,/\, \mathsf{f}(x)) \succcurlyeq \lambda x.\mathsf{f}(x)}\,\mathsf{R}_\lambda}{\mu(\lambda x.\mathsf{f}(x)) \,/\, \mathsf{mu}(\lambda x.(\rho(x) \,/\, \mathsf{f}(x))) \succcurlyeq \mu(\lambda x.\mathsf{f}(x))}\,\mathsf{R}_\mathrm{R}}{\mathsf{f}(\mu(\lambda x.\mathsf{f}(x)) \,/\, \mathsf{mu}(\lambda x.(\rho(x) \,/\, \mathsf{f}(x)))) \succcurlyeq \mathsf{f}(\mu(\lambda x.\mathsf{f}(x)))}\,\mathsf{repl}_f$$

In the next example we calculate the residual of one higher-order reduction after another. For this, we combine the rewrite system $\Rightarrow_\mathrm{R}$ and the inference system $\mathcal{R}es$.

Example 6.3.3. Consider the HRS $\mathfrak{Mu}$ from page 118. Let the following reductions be given:

$$\mathcal{R} : \mathsf{mu}(\lambda x.\mathsf{f}(x)) \to \mathsf{mu}(\lambda x.\mathsf{g}(x)) \to \mathsf{g}(\mathsf{mu}(\lambda x.\mathsf{g}(x)))$$
$$\mathcal{S} : \mathsf{mu}(\lambda x.\mathsf{f}(x)) \to \mathsf{f}(\mathsf{mu}(\lambda x.\mathsf{f}(x))) \to \mathsf{g}(\mathsf{mu}(\lambda x.\mathsf{f}(x)))$$

We construct the sequences of proof terms that witness both reductions:

$$\mathcal{R} = \mathsf{mu}(\lambda x.\rho(x)) \,;\, \mu(\lambda x.\mathsf{f}(x))$$
$$\mathcal{S} = \mu(\lambda x.\mathsf{f}(x)) \,;\, \rho(\mathsf{mu}(\lambda x.\mathsf{f}(x)))$$

We normalize $\mathcal{R} \,/\, \mathcal{S}$ and $\mathcal{S} \,/\, \mathcal{R}$ with the rewrite system $\Rightarrow_{\mathrm{R}}$ (given on page 114).

$$\begin{aligned}
\mathcal{R} \,/^*\, \mathcal{S} &= \mathsf{mu}(\lambda x.\rho(x)) \,;\, \mu(\lambda x.\mathsf{f}(x)) \,/^*\, \mu(\lambda x.\mathsf{f}(x)) \,;\, \rho(\mathsf{mu}(\lambda x.\mathsf{f}(x))) \\
&\Rightarrow_{\mathrm{R}} ((\mathsf{mu}(\lambda x.\rho(x)) \,;\, \mu(\lambda x.\mathsf{f}(x))) \,/^*\, \mu(\lambda x.\mathsf{f}(x))) \,/^*\, \rho(\mathsf{mu}(\lambda x.\mathsf{f}(x))) \\
&\Rightarrow^*_{\mathrm{R}} ((\mathsf{mu}(\lambda x.\rho(x)) \,/\, \mu(\lambda x.\mathsf{f}(x))) \\
&\qquad\qquad ;\, (\mu(\lambda x.\mathsf{f}(x)) \,/^*\, (\mu(\lambda x.\mathsf{f}(x)) \,/\, \mathsf{mu}(\lambda x.\rho(x))))) \\
&\qquad /^*\, \rho(\mathsf{mu}(\lambda x.\mathsf{f}(x))) \\
&\Rightarrow^*_{\mathrm{R}} (\rho(\mathsf{mu}(\lambda x.\rho(x))) \,;\, (\mu(\lambda x.\mathsf{f}(x)) \,/\, \mu(\lambda x.\mathsf{f}(x)))) \\
&\qquad /^*\, \rho(\mathsf{mu}(\lambda x.\mathsf{f}(x))) \\
&\Rightarrow^*_{\mathrm{R}} (\rho(\mathsf{mu}(\lambda x.\rho(x))) \,;\, 1) \,/\, \rho(\mathsf{mu}(\lambda x.\mathsf{f}(x))) \\
&\Rightarrow_{\mathrm{R}} \rho(\mathsf{mu}(\lambda x.\rho(x))) \,/\, \rho(\mathsf{mu}(\lambda x.\mathsf{f}(x))) \\
&\Rightarrow_{\mathrm{R}} \mathsf{f}(\mathsf{mu}(\lambda x.\rho(x)))
\end{aligned}$$

and:

$$\begin{aligned}
\mathcal{S} \,/^*\, \mathcal{R} &= (\mu(\lambda x.\mathsf{f}(x)) \,;\, \rho(\mathsf{mu}(\lambda x.\mathsf{f}(x)))) \,/^*\, (\mathsf{mu}(\lambda x.\rho(x)) \,;\, \mu(\lambda x.\mathsf{f}(x))) \\
&\Rightarrow_{\mathrm{R}} ((\mu(\lambda x.\mathsf{f}(x)) \,;\, \rho(\mathsf{mu}(\lambda x.\mathsf{f}(x)))) \,/^*\, \mathsf{mu}(\lambda x.\rho(x))) \,/^*\, \mu(\lambda x.\mathsf{f}(x)) \\
&\Rightarrow^*_{\mathrm{R}} ((\mu(\lambda x.\mathsf{f}(x)) \,/\, \mathsf{mu}(\lambda x.\rho(x))) \\
&\qquad\qquad ;\, (\rho(\mathsf{mu}(\lambda x.\mathsf{f}(x))) \,/^*\, (\mathsf{mu}(\lambda x.\rho(x)) \,/\, \mu(\lambda x.\mathsf{f}(x))))) \\
&\qquad /^*\, \mu(\lambda x.\mathsf{f}(x)) \\
&\Rightarrow^*_{\mathrm{R}} (\mu(\lambda x.\mathsf{f}(x)) \,;\, (\rho(\mathsf{mu}(\lambda x.\mathsf{f}(x))) \,/\, \rho(\mathsf{mu}(\lambda x.\rho(x))))) \\
&\qquad /^*\, \mu(\lambda x.\mathsf{f}(x)) \\
&\Rightarrow^*_{\mathrm{R}} (\mu(\lambda x.\mathsf{f}(x)) \,;\, 1) \,/^*\, \mu(\lambda x.\mathsf{f}(x)) \\
&\Rightarrow^*_{\mathrm{R}} 1
\end{aligned}$$

The reduction steps of the form $\varphi \,/\, \psi \Rightarrow_{\mathrm{R}} \chi$ are due to simple inferences in the $\mathcal{R}es$ inference system, for example the first one:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x \,/\, x \succcurlyeq x}\,\mathsf{R}_x}{\rho(x) \,/\, \mathsf{f}(x) \succcurlyeq \rho(x)}\,\mathsf{R}_{\mathrm{R}}}{\lambda x.\rho(x) \,/\, \lambda x.\mathsf{f}(x) \succcurlyeq \lambda x.\rho(x)}\,\mathsf{R}_\lambda}{\mathsf{mu}(\lambda x.\rho(x)) \,/\, \mu(\lambda x.\mathsf{f}(x)) \succcurlyeq \rho(\mathsf{mu}(\lambda x.\rho(x)))}\,\mathsf{R}_{\mathrm{L}}$$

With $\mathsf{pr}(\mathcal{K})$ we denote the *principal rule* of an inference $\mathcal{K}$, that is the rule which applied in the last step of the inference. Finally, the function $\mathsf{dpt}(\mathcal{K})$ returns the depth of $\mathcal{K}$, which, if we define that $\max \varnothing = 0$, can be recursively defined as

$$\mathsf{dpt}(\mathcal{K}) = \max\{\mathsf{dpt}(\mathcal{K}_i) \mid 1 \leq i \leq n\} + 1.$$

where $\mathcal{K}_1, \ldots, \mathcal{K}_n$ are the immediate subinferences of $\mathcal{K}$ (if any). For example, let $\mathcal{K}$ be the inference of Ex. 6.3.1. Then $\mathsf{pr}(\mathcal{K}) = \mathsf{R}_{\mathrm{L}}$, $\mathsf{dpt}(\mathcal{K}) = 4$ and $\vdash^{\mathcal{K}} \mathsf{mu}(\lambda x.\rho(x)) \,/\, \mu(\lambda x.\mathsf{f}(x)) \succcurlyeq \rho(\mathsf{mu}(\lambda x.\rho(x)))$.

It is easy to show that projection terms simplify to proof terms:

Lemma 6.3.4. *If $\varphi \succcurlyeq \chi$, then χ is a proof term.*

Proof. Suppose $\vDash^{\mathcal{K}} \varphi \succcurlyeq \chi$. We use induction on $\mathsf{dpt}(\mathcal{K})$. The conclusion follows because none of the inference rules of $\mathcal{R}es$ have projection symbols on the right of the $\succcurlyeq$-symbol, and thus χ must be a proof term. □

First, we prove a few standardization-like properties of the inference system. The next lemma proves the completeness of some sort of 'inference strategy': given a projection term φ, in each case that more than one inference rule can be applied to a desired goal of the form $\varphi \succcurlyeq \chi$, it is possible to deterministically choose one of those rules such that a solution can be found. This property is useful for two purposes:

- it allows us to prove (in Lemma 6.3.8) that each projection term simplifies to a unique proof term;
- second, as a direct consequence of the property, it is, given a projection term, decidable to which proof term it simplifies (we give an algorithm to do this in Sect. 6.3.5).

Lemma 6.3.5. *Suppose $\vdash^{\mathcal{K}} \varphi \,/\, \psi \succcurlyeq \chi$.*

(i) If $\varphi = x(\varphi_1, \ldots, \varphi_n)$ and $\psi = x(\psi_1, \ldots, \psi_n)$, then there exists an inference $\mathcal{K}'$ with $\mathsf{dpt}(\mathcal{K}') \leq \mathsf{dpt}(\mathcal{K})$ such that $\vdash^{\mathcal{K}'} \varphi \,/\, \psi \succcurlyeq \chi$ and $\mathsf{pr}(\mathcal{K}') = \mathsf{R}_x$.

(ii) If $\varphi = f(\varphi_1, \ldots, \varphi_n)$ and $\psi = f(\psi_1, \ldots, \psi_n)$, then there exists an inference $\mathcal{K}'$ with $\mathsf{dpt}(\mathcal{K}') \leq \mathsf{dpt}(\mathcal{K})$ such that $\vdash^{\mathcal{K}'} \varphi \,/\, \psi \succcurlyeq \chi$ and $\mathsf{pr}(\mathcal{K}') = \mathsf{R}_f$.

(iii) If $\varphi = \rho(\varphi_1, \ldots, \varphi_n)$ and $\psi = \rho(\psi_1, \ldots, \psi_n)$, then there exists an inference $\mathcal{K}'$ with $\mathsf{dpt}(\mathcal{K}') \leq \mathsf{dpt}(\mathcal{K})$ such that $\vdash^{\mathcal{K}'} \varphi \,/\, \psi \succcurlyeq \chi$ and $\mathsf{pr}(\mathcal{K}') = \mathsf{R}_\rho$.

(iv) If $\varphi = \rho(\varphi_1, \ldots, \varphi_n)$ and $\psi = l(\psi_1, \ldots, \psi_n)$, then there exists an inference $\mathcal{K}'$ with $\mathsf{dpt}(\mathcal{K}') \leq \mathsf{dpt}(\mathcal{K})$ such that $\vdash^{\mathcal{K}'} \varphi \,/\, \psi \succcurlyeq \chi$ and $\mathsf{pr}(\mathcal{K}') = \mathsf{R}_{\mathrm{R}}$.

(v) If $\varphi = l(\varphi_1, \ldots, \varphi_n)$ and $\psi = \rho(\psi_1, \ldots, \psi_n)$, then there exists an inference $\mathcal{K}'$ with $\mathsf{dpt}(\mathcal{K}') \leq \mathsf{dpt}(\mathcal{K})$ such that $\vdash^{\mathcal{K}'} \varphi \,/\, \psi \succcurlyeq \chi$ and $\mathsf{pr}(\mathcal{K}') = \mathsf{R}_{\mathrm{L}}$.

Proof. We only prove item (iii); the other cases can be proved analogously. We use induction on the depth of the inference. There are only two inference rules which match the conclusion, R_ρ and $\mathsf{r} + \mathsf{t}_{/}$, so we distinguish the following two cases:

- If $\mathsf{pr}(\mathcal{K}) = \mathsf{R}_\rho$, then we simply take $\mathcal{K}' = \mathcal{K}$.
- If $\mathsf{pr}(\mathcal{K}) = \mathsf{r} + \mathsf{t}_/$, then we know, by the induction hypothesis and the observation that only the repl_ρ-rule matches conclusions of the form $\rho(\varphi_1, \ldots, \varphi_n) \succcurlyeq \chi$, that the following inference $\mathcal{L}$ exists:

$$\dfrac{\dfrac{\cdots \varphi_i \succcurlyeq \varphi_i' \cdots}{\rho(\overline{\varphi}) \succcurlyeq \rho(\overline{\varphi'})}\,\mathsf{repl}_\rho \quad \dfrac{\cdots \psi_i \succcurlyeq \psi_i' \cdots}{\rho(\overline{\psi}) \succcurlyeq \rho(\overline{\psi'})}\,\mathsf{repl}_\rho \quad \dfrac{\cdots \varphi_i' \,/\, \psi_i' \succcurlyeq \chi_i \cdots}{\rho(\overline{\varphi'}) \,/\, \rho(\overline{\psi'}) \succcurlyeq r(\overline{\chi})}\,\mathsf{R}_\rho}{\rho(\overline{\varphi}) \,/\, \rho(\overline{\psi}) \succcurlyeq r(\overline{\chi})}\,\mathsf{r} + \mathsf{t}_/$$

Now, we construct the desired inference $\mathcal{K}'$ as follows:

$$\dfrac{\cdots \quad \dfrac{\varphi_i \succcurlyeq \varphi_i' \quad \psi_i \succcurlyeq \psi_i' \quad \varphi_i' \,/\, \psi_i' \succcurlyeq \chi_i}{\varphi_i \,/\, \psi_i \succcurlyeq \chi_i}\,\mathsf{r} + \mathsf{t}_/ \quad \cdots}{\rho(\overline{\varphi}) \,/\, \rho(\overline{\psi}) \succcurlyeq r(\overline{\chi})}\,\mathsf{R}_\rho$$

□

The relation $\succcurlyeq$ formalized by the inference system $\mathcal{R}es$ is not total, in the sense that there are projection terms φ such that $\varphi \succcurlyeq \chi$ is not defined for any χ. First, of course, $\varphi \,/\, \psi$ does not simplify to any proof term if φ and ψ are not coinitial. For example, none of the rules have a conclusion of the form $f(\varphi') \,/\, g(\psi') \succcurlyeq \chi$, where f and g are different function symbols and χ is an arbitrary proof term. But, in fact, the relation is also not total if we restrict our attention to projection terms of the form $\varphi \,/\, \psi$, where φ and ψ are coinitial multisteps. Consider an HRS consisting of two rules $\rho : \mathsf{f}(x) \to \mathsf{g}(x)$ and $\theta : \mathsf{f}(x) \to \mathsf{h}(x)$. The reader can easily verify that there is no proof term χ such that $\rho(\mathsf{a}) \,/\, \theta(\mathsf{a}) \succcurlyeq \chi$. The system above is not confluent. But even for confluent HRSs, the simplification relation is not total. Let the HRS with the single rule $\rho : \mathsf{f}(\mathsf{f}(x)) \to \mathsf{a}$ be given. Then both $\mathsf{f}(\rho(\mathsf{a}))$ and $\rho(\mathsf{f}(\mathsf{a}))$ are proof terms with source $\mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{a})))$. Still, it is easily seen that no inference exists with a conclusion of the form $\mathsf{f}(\rho(\mathsf{a})) \,/\, \rho(\mathsf{f}(\mathsf{a})) \succcurlyeq \chi$ From now on, we restrict our attention to projection terms for which the $\succcurlyeq$ relation is defined, that is we restrict our attention to *compatible* projection terms:

Definition 6.3.6 (Compatible)**.**

(i) A projection term φ is *compatible* if, for some proof term χ, an inference exists with a conclusion of the form $\varphi \succcurlyeq \chi$.

(ii) A proof term φ is *compatible with* a proof term ψ, if the projection term $\varphi \,/\, \psi$ is compatible.

A reduction $\mathcal{R}$ is compatible with a reduction $\mathcal{S}$ if, for all $\mathcal{T}$ such that $\mathcal{R} \,/^* \mathcal{S} \Rightarrow^*_{\mathrm{R}} \mathcal{T}$, all projection terms in $\mathcal{T}$ are compatible.

(iii) An HRS $\mathfrak{H}$ is compatible if all (proof terms witnessing) coinitial $\mathfrak{H}$-steps are compatible.

By definition, the simplification relation is total on compatible projection terms. In Sect. 6.4 it is shown that the notion of compatibility naturally corresponds to the notion of orthogonality.

Lemma 6.3.7. *Let $\mathfrak{H}$ be an* HRS.

(i) $\mathfrak{H}$ has pairs of incompatible reductions if and only if it has pairs of incompatible multisteps.

(ii) $\mathfrak{H}$ has pairs of incompatible multisteps if and only if it has pairs of incompatible proper steps.

Proof. (i) Let $\mathcal{R}, \mathcal{S}$ be incompatible reductions. By definition, there is a $\mathcal{T}$ such that $\mathcal{R} /^* \mathcal{R}' \Rightarrow_{\mathrm{R}}^* \mathcal{T}$ and which has incompatible multisteps. So, $\mathfrak{H}$ has pairs of incompatible multisteps. The other direction is immediate, because a multistep is also a reduction (of length 1).

(ii) By the facts that each proper step is a multistep by definition, and that, given incompatible multisteps φ, ψ, it is easy to construct incompatible proper steps φ', ψ' using Lemma 6.3.5. □

In the following we will consider / as a binary function on proof terms, that is, we will sometimes write $\varphi / \psi = \chi$ for $\varphi / \psi \succcurlyeq \chi$. That this function is well-defined, and satisfies the residual laws, is proved in the next section.

6.3.4 Correctness of the residual operator

In this section we prove that the residual operator defined in the previous section is well-defined (that is, it is actually a total function on compatible projection terms) and satisfies the residual laws of Def. 6.2.1.

First, we prove that the operator is indeed a function. This is done by showing that each projection term simplifies to a unique proof term, that is, if $\varphi \succcurlyeq \chi$ and $\varphi \succcurlyeq \chi'$, then $\chi = \chi'$. For this we use the standardization property of Lemma 6.3.5.

Lemma 6.3.8. *If $\varphi \succcurlyeq \chi$ and $\varphi \succcurlyeq \chi'$, then $\chi = \chi'$.*

Proof. Let $\vdash^{\mathcal{K}} \varphi \succcurlyeq \chi$ and $\vdash^{\mathcal{L}} \varphi \succcurlyeq \chi'$. We prove the lemma by induction on $\mathsf{dpt}(\mathcal{K}) + \mathsf{dpt}(\mathcal{L})$. If $\varphi = \varphi_1 / \varphi_2$, then, by Lemma 6.3.5, there exist inferences $\mathcal{K}', \mathcal{L}'$ such that $\vdash^{\mathcal{K}'} \varphi \succcurlyeq \chi$, $\vdash^{\mathcal{L}'} \varphi \succcurlyeq \chi'$ and $\mathsf{pr}(\mathcal{K}') = \mathsf{pr}(\mathcal{L}')$. Otherwise, only one rule of inference matches the desired conclusion, and so it must be the case that $\mathsf{pr}(\mathcal{K}) = \mathsf{pr}(\mathcal{L})$. In this case, we take $\mathcal{K}' := \mathcal{K}$ and $\mathcal{L}' := \mathcal{L}$.

In both cases the desired result follows from the induction hypothesis, because $\mathsf{dpt}(\mathcal{K}') \leq \mathsf{dpt}(\mathcal{K})$ and $\mathsf{dpt}(\mathcal{L}') \leq \mathsf{dpt}(\mathcal{L})$, and thus the depth of the direct subinferences is strictly smaller. □

We define the relation $\sim$ to be the reflexive, transitive closure of $\succcurlyeq$. By Lemmas 6.3.4 and 6.3.8 it is the case that $\varphi \sim \psi$ if and only if there exists a proof term χ such that $\varphi \succcurlyeq \chi$ and $\psi \succcurlyeq \chi$. Proof terms can now be considered the unique representatives of $\sim$-equivalence classes.

Lemma 6.3.9. *$\sim$ is a congruence, that is, for projection terms φ, ψ it holds that if $\varphi \sim \psi$, we have:*

(i) $x(\ldots, \varphi, \ldots) \sim x(\ldots, \psi, \ldots)$ *and* $f(\ldots, \varphi, \ldots) \sim f(\ldots, \psi, \ldots)$*;*

(ii) $\rho(\ldots, \varphi, \ldots) \sim \rho(\ldots, \psi, \ldots)$*;*

(iii) $\lambda x.\varphi \sim \lambda x.\psi$*;*

(iv) $\varphi \,/\, \chi \sim \psi \,/\, \chi$ *and* $\chi \,/\, \varphi \sim \chi \,/\, \psi$*, for any* χ*.*

Proof. Since $\varphi \sim \psi$, there is a ξ such that $\varphi \succcurlyeq \xi$ and $\psi \succcurlyeq \xi$. (i)-(iii) are then easily proved by applying the appropriate replacement rule (resp. repl_x or repl_f, repl_ρ and repl_λ), using the facts that $\varphi \succcurlyeq \xi$ and $\psi \succcurlyeq \xi$. The first part of (iv) follows from the following two inferences:

$$\frac{\varphi \succcurlyeq \xi \quad \chi \succcurlyeq \chi' \quad \xi \,/\, \chi' \succcurlyeq \xi'}{\varphi \,/\, \chi \succcurlyeq \xi'} \qquad \frac{\psi \succcurlyeq \xi \quad \chi \succcurlyeq \chi' \quad \xi \,/\, \chi' \succcurlyeq \xi'}{\psi \succcurlyeq \chi \,/\, \xi'}$$

The second part of (iv) is similar. □

In the next two lemmas we prove that the conditions of Def. 6.2.1 hold, that is, that the requirements on sources and targets are satisfied, and the residual laws hold.

Lemma 6.3.10. *Sources and targets match, that is:*

(i) $\mathrm{src}(\varphi \,/\, \psi) = \mathrm{tgt}(\psi)$

(ii) $\mathrm{tgt}(\varphi \,/\, \psi) = \mathrm{tgt}(\psi \,/\, \varphi)$

Proof. By induction of the inferences of $\varphi \,/\, \psi \succcurlyeq \chi$ and $\psi \,/\, \varphi \succcurlyeq \xi$ we easily show that $\mathrm{src}(\chi) = \mathrm{tgt}(\psi)$ and $\mathrm{tgt}(\chi) = \mathrm{tgt}(\xi)$. □

Lemma 6.3.11. *The residual laws hold, that is, for proof terms φ, ψ, χ:*

(i) $1 \,/\, \varphi \sim 1$

(ii) $\varphi \,/\, 1 \sim \varphi$

(iii) $\varphi \,/\, \varphi \sim 1$

(iv) $(\varphi \,/\, \psi) \,/ (\chi \,/\, \psi) \sim (\varphi \,/\, \chi) \,/ (\psi \,/\, \chi)$

Proof. (i) By induction on the length of φ. If $\varphi = f(\varphi_1, \ldots, \varphi_n)$, then:

$$\frac{\begin{array}{cc} \vdots\,\mathrm{IH} & \vdots\,\mathrm{IH} \\ \varphi_1 \,/\, 1 \succcurlyeq \varphi_1 & \ldots \quad \varphi_n \,/\, 1 \succcurlyeq \varphi_n \end{array}}{f(\varphi_1, \ldots, \varphi_n) \,/\, f(1, \ldots, 1) \succcurlyeq f(1, \ldots, 1)}\,\mathsf{R}_f$$

where we note that $f(1, \ldots, 1) = 1$. The cases that $\varphi = x(\varphi_1, \ldots, \varphi_n)$, $\varphi = \rho(\varphi_1, \ldots, \varphi_n)$ and $\varphi = \lambda x.\varphi_0$ are handled in similar ways. (ii) and (iii) are proved analogously.

(iv) By induction on the total length of φ, ψ, χ. Suppose that we have $\varphi = f(\varphi_1, \ldots, \varphi_n)$, $\psi = f(\psi_1, \ldots, \psi_n)$ and $\chi = f(\chi_1, \ldots, \chi_n)$. By Lemma 6.3.5 the following inferences must exist:

$$\dfrac{\dfrac{\vdots\,\mathcal{K}_1}{\cdots\ \varphi_i \,/\, \psi_i \succcurlyeq \zeta_{1,i}\ \cdots} \quad \dfrac{\vdots\,\mathcal{K}_2}{\cdots\ \chi_i \,/\, \psi_i \succcurlyeq \zeta'_{1,i}\ \cdots} \quad \dfrac{\vdots\,\mathcal{K}_3}{\cdots\ \zeta_{1,i} \,/\, \zeta'_{1,i} \succcurlyeq \xi_{1,i}\ \cdots}}{(f(\overline{\varphi}) \,/\, f(\overline{\psi})) \,/ (f(\overline{\chi}) \,/\, f(\overline{\psi})) \succcurlyeq f(\overline{\xi_1})}$$

$$\dfrac{\dfrac{\vdots\,\mathcal{L}_1}{\cdots\ \varphi_i \,/\, \chi_i \succcurlyeq \zeta_{2,i}\ \cdots} \quad \dfrac{\vdots\,\mathcal{L}_2}{\cdots\ \psi_i \,/\, \chi_i \succcurlyeq \zeta'_{2,i}\ \cdots} \quad \dfrac{\vdots\,\mathcal{L}_3}{\cdots\ \zeta_{2,i} \,/\, \zeta'_{2,i} \succcurlyeq \xi_{1,i}\ \cdots}}{(f(\overline{\varphi}) \,/\, f(\overline{\chi})) \,/ (f(\overline{\psi}) \,/\, f(\overline{\chi})) \succcurlyeq f(\overline{\xi_2})}$$

Using the same subinferences, we prove $(\varphi_i \,/\, \psi_i) \,/ (\chi_i \,/\, \psi_i) \succcurlyeq \xi_1$:

$$\dfrac{\vdots\,\mathcal{K}_1 \quad \vdots\,\mathcal{K}_2 \quad \vdots\,\mathcal{K}_3}{\varphi_i \,/\, \psi_i \succcurlyeq \zeta_{1,i} \quad \chi_i \,/\, \psi_i \succcurlyeq \zeta'_{1,i} \quad \zeta_{1,i} \,/\, \zeta'_{1,i} \succcurlyeq \xi_{1,i}}{(\varphi_i \,/\, \psi_i) \,/ (\chi_i \,/\, \psi_i) \succcurlyeq \xi_1}$$

and similarly $(\varphi_i \,/\, \chi_i) \,/ (\psi_i \,/\, \chi_i) \succcurlyeq \xi_2$. By induction hypothesis,

$$(\varphi_i \,/\, \psi_i) \,/ (\chi_i \,/\, \psi_i) \sim (\varphi_i \,/\, \chi_i) \,/ (\psi_i \,/\, \chi_i)$$

and therefore $\xi_{1,i} = \xi_{2,i}$. Thus:

$$(f(\overline{\varphi}) \,/\, f(\overline{\psi})) \,/ (f(\overline{\chi}) \,/\, f(\overline{\psi})) \sim (f(\overline{\varphi}) \,/\, f(\overline{\chi})) \,/ (f(\overline{\psi}) \,/\, f(\overline{\chi}))$$

For the other triples of coinitial steps the proofs proceed in the same way. □

We can now derive the following result:

Theorem 6.3.12. *Let $\mathfrak{H}$ be an* HRS*. The triple $\mathbf{H} = \langle \mathfrak{H}, 1, / \rangle$ is a residual system.*

Proof. Directly by Lemma 6.3.10 and Lemma 6.3.11. □

Confluence of compatible HRSs follows directly from the previous theorem and the fact that with the help of the residual operator is easy to find a common reduct.

Corollary 6.3.13. *A compatible* HRS *is confluent.*

Proof. By Theorem 6.3.12 and Theorem 6.2.7. □

6.3.5 Computing the simplification relation

In Sect. 6.3.3 only a specification of the simplification relation was given. Here, we present an algorithm which, given a projection term φ, effectively computes the proof term φ simplifies to. If a compatible projection term is given as input, the proof term it simplifies to is printed; otherwise, the program prints "incompatible". The program is written in a (pseudo) functional programming language that implements an eager reduction strategy.

Definition 6.3.14. The (recursive) function $\mathrm{sim}(\pi)$ on projection terms π, is defined by the following program (in pseudo-code):

$\mathrm{sim}((\varphi_1 \,/\, \varphi_2) \,/\, \psi) = \mathrm{sim}(\varphi' \,/\, \psi)$
  where $\varphi' = \mathrm{sim}(\varphi_1 \,/\, \varphi_2)$
$\mathrm{sim}(\varphi \,/ (\psi_1 \,/\, \psi_2)) = \mathrm{sim}(\varphi \,/\, \psi')$
  where $\psi' = \mathrm{sim}(\psi_1 \,/\, \psi_2)$
$\mathrm{sim}(x(\varphi_1, \ldots, \varphi_n) \,/\, x(\psi_1, \ldots, \psi_n)) = x(\mathrm{sim}(\varphi_1 \,/\, \psi_1), \ldots, \mathrm{sim}(\varphi_n \,/\, \psi_n))$
$\mathrm{sim}(f(\varphi_1, \ldots, \varphi_n) \,/\, f(\psi_1, \ldots, \psi_n)) = f(\mathrm{sim}(\varphi_1 \,/\, \psi_1), \ldots, \mathrm{sim}(\varphi_n \,/\, \psi_n))$
$\mathrm{sim}(\rho(\varphi_1, \ldots, \varphi_n) \,/\, \rho(\psi_1, \ldots, \psi_n)) = r(\mathrm{sim}(\varphi_1 \,/\, \psi_1), \ldots, \mathrm{sim}(\varphi_n \,/\, \psi_n))$
$\mathrm{sim}(\rho(\varphi_1, \ldots, \varphi_n) \,/\, l(\psi_1, \ldots, \psi_n)) = \rho(\mathrm{sim}(\varphi_1 \,/\, \psi_1), \ldots, \mathrm{sim}(\varphi_n \,/\, \psi_n))$
$\mathrm{sim}(l(\varphi_1, \ldots, \varphi_n) \,/\, \rho(\psi_1, \ldots, \psi_n)) = r(\mathrm{sim}(\varphi_1 \,/\, \psi_1), \ldots, \mathrm{sim}(\varphi_n \,/\, \psi_n))$
$\mathrm{sim}(\lambda x.\varphi \,/\, \lambda x.\psi) = \lambda x.(\varphi \,/\, \psi)$
$\mathrm{sim}(f(\varphi_1, \ldots, \varphi_n)) = f(\mathrm{sim}(\varphi_1), \ldots, \mathrm{sim}(\varphi_n))$
$\mathrm{sim}(\rho(\varphi_1, \ldots, \varphi_n)) = \rho(\mathrm{sim}(\varphi_1), \ldots, \mathrm{sim}(\varphi_n))$
$\mathrm{sim}(\lambda x.\varphi) = \lambda x.\mathrm{sim}(\varphi)$
if *none of the above cases apply* **then**
  print "*incompatible*"

Basically, the algorithm does nothing more than apply the strategy that was already hinted at in Lemma 6.3.5. However, it is useful anyway, because it can also be applied to incompatible projection terms, thus providing a useful method to analyse those as well.

Example 6.3.15. Consider the HRS $\mathfrak{Mu}$ from Sect. 6.3.2. We apply the algorithm of Def. 6.3.14 to the projection term $\mathsf{mu}(\lambda x.\rho(x)) \,/\, \mu(\lambda x.\mathsf{f}(x))$. The following is the trace of the algorithm:

$$\begin{aligned}
&\mathrm{sim}(\mathsf{mu}(\lambda x.\rho(x)) \,/\, \mu(\lambda x.\mathsf{f}(x))) \\
&= \quad (\lambda z.z(\mathsf{mu}(\lambda x.z(x))))(\mathrm{sim}(\lambda x.\rho(x)/\lambda x.\mathsf{f}(x))) \\
&= \quad (\lambda z.z(\mathsf{mu}(\lambda x.z(x))))(\lambda x.\mathrm{sim}(\rho(x)/\mathsf{f}(x))) \\
&= \quad (\lambda z.z(\mathsf{mu}(\lambda x.z(x))))(\lambda x.\rho(\mathrm{sim}(x/x))) \\
&= \quad (\lambda z.z(\mathsf{mu}(\lambda x.z(x))))(\lambda x.\rho(x)) \\
&\to\!\!\!\!\to_\beta \ \rho(\mathsf{mu}(\lambda x.\rho(x)))
\end{aligned}$$

We see that the result of the previous example is the same as the result of Ex. 6.3.1. So why does the algorithm work, and doesn't the rewriting system presented in Sect. 6.3.2? The crucial difference is that the algorithm calculates

subexpressions first, and does β-reductions afterwards, while the rewriting system performs β-reductions first and only then calculates the subexpressions. And even if this was fixed in the rewriting system, it would have to be equipped with a depth-first strategy to make it correct.

Lemma 6.3.16. $\mathrm{sim}(\varphi)$ *terminates for all projection terms* φ.

Proof. Trivial, because the length of the arguments of recursive calls to sim are strictly smaller. □

Lemma 6.3.17.

(i) If φ *is a compatible projection term, then* $\mathrm{sim}(\varphi) = \chi$ *if and only if* $\varphi \succcurlyeq \chi$.

(ii) If φ *is an incompatible projection term, then* $\mathrm{sim}(\varphi) =$ *"incompatible".*

Proof. (i) The 'only if' side is easily proved by recursively constructing an inference $\mathcal{K}$ of $\varphi \succcurlyeq \chi$. The 'if' side follows easily by induction on the inference of $\varphi \succcurlyeq \chi$, using Lemma 6.3.5.

(ii) From the fact that the cases of the program correspond exactly to the conlusions of the inference rules. □

6.4 Compatibility is orthogonality

In the previous sections we restricted our attention to compatible steps and reductions. Compatibility, however, is not a part of the standard rewriting jargon found in the literature. In this section, we prove that compatibility coincides with the well-known property of orthogonality, which yields an alternative proof of confluence of orthogonal HRSs.

Proposition 6.4.1. *An* HRS *is orthogonal if and only if it is compatible.*

Proof. Let $\mathfrak{H} = \langle \Sigma, R \rangle$ be an HRS. Note that we restrict our attention to linear HRSs.

($\Rightarrow$): Assume that $\mathfrak{H}$ is not compatible. By Lemma 6.3.7 there are coinitial proper steps φ, ψ that are incompatible. By Lemma 6.3.17, $\mathrm{sim}(\varphi \,/\, \psi)$ returns "incompatible". Without loss of generality, we assume that the projection term $\varphi \,/\, \psi$ was passed to sim in the last step before it terminated. Analysis of the program reveals that either $\varphi = \rho(s_1, \ldots, s_n)$ and $\psi \neq l(\psi_1, \ldots, \psi_n)$, or $\psi = \rho(s_1, \ldots, s_n)$ and $\varphi \neq l(\varphi_1, \ldots, \varphi_n)$, where $\rho : l \to r$ is a rule. We consider only the first case; the second case is symmetrical.

This case can only occur if $l = C[l_0]$, and $\psi = l'(\psi_1, \ldots, \psi_n)$, where $l' = C[\chi]$, for some proof term χ with a rule symbol as head. By induction on l it is easily shown that φ and ψ are non-orthogonal.

($\Leftarrow$): Assume that all coinitial multisteps are compatible. This implies, by Lemma 6.3.7(ii), that all coinitial proper steps are compatible. Let φ, ψ be the proof terms for such steps. There exists an inference $\mathcal{K}$ such that $\vdash^{\mathcal{K}} \varphi \,/\, \psi \succcurlyeq \chi$. We easily prove, by induction on $\mathcal{K}$, that φ, ψ are orthogonal. □

Confluence of orthogonal HRSs was proved by Nipkow in [37]. Using the result of the previous theorem, we give an alternative proof here by using residual theory:

Theorem 6.4.2. *An orthogonal* HRS *is confluent.*

Proof. Directly by Prop. 6.4.1 and Cor. 6.3.13. □

6.5 The projection order and projection equivalence

The projection operator provides an elegant way to (partially) order steps and reductions, the projection order. In turn, the projection order gives rise to an equivalence relation on reductions, projection equivalence. In this section we define the projection order and projection equivalence and prove that both relations have the expected properties. In the next section we will prove that projection equivalence actually coincides with permutation and standardization equivalence.

Definition 6.5.1. Let $\mathbf{R} = \langle \mathfrak{R}, 1, / \rangle$ be a residual system. We define, for $\mathfrak{R}$-steps φ, ψ:

$$\varphi \lesssim \psi \text{ if } \varphi / \psi = 1$$
$$\varphi \simeq \psi \text{ if } \varphi \lesssim \psi \text{ and } \psi \lesssim \varphi$$

We also define: $\gtrsim = \lesssim^{-1}$. Note that finite reductions are the steps of ARSC's, and so we have also defined the relations $\lesssim$ and $\simeq$ on finite reductions.

Example 6.5.2. Consider the HRS $\mathfrak{M}u$ from page 118 and the reductions:

$$\mathcal{R} = \mu(\lambda x.\rho(x))$$
$$\mathcal{S} = \mu(\lambda x.\mathsf{f}(x)), \mathsf{f}(\mathsf{mu}(\lambda x.\rho(x))), \rho(\mathsf{mu}(\lambda x.\mathsf{g}(x)))$$

First, we calculate $\mathcal{R} / \mathcal{S}$:

$$\begin{aligned}
&\mu(\lambda x.\rho(x)) / (\mu(\lambda x.\mathsf{f}(x)), \mathsf{f}(\mathsf{mu}(\lambda x.\rho(x))), \rho(\mathsf{mu}(\lambda x.\mathsf{f}(x))))\\
&\Rightarrow^* ((\mu(\lambda x.\rho(x)) / \mu(\lambda x.\mathsf{f}(x))) / \mathsf{f}(\mathsf{mu}(\lambda x.\rho(x)))) / \rho(\mathsf{mu}(\lambda x.\mathsf{f}(x)))\\
&\Rightarrow \quad \rho(\mathsf{mu}(\lambda x.\rho(x))) / \mathsf{f}(\mathsf{mu}(\lambda x.\rho(x)))) / \rho(\mathsf{mu}(\lambda x.\mathsf{f}(x)))\\
&\Rightarrow \quad \rho(\mathsf{mu}(\lambda x.\mathsf{g}(x))) / \rho(\mathsf{mu}(\lambda x.\mathsf{g}(x)))\\
&\Rightarrow \quad 1
\end{aligned}$$

Then, $\mathcal{S} \,/\, \mathcal{R}$:

$$
\begin{aligned}
&(\mu(\lambda x.\mathsf{f}(x)), \mathsf{f}(\mathsf{mu}(\lambda x.\rho(x))), \rho(\mathsf{mu}(\lambda x.\mathsf{f}(x)))) \,/\, \mu(\lambda x.\rho(x)) \\
&\Rightarrow \quad (\mu(\lambda x.\mathsf{f}(x)) \,/\, \mu(\lambda x.\rho(x))) \,; \\
&\qquad\quad ((\mathsf{f}(\mathsf{mu}(\lambda x.\rho(x))), \rho(\mathsf{mu}(\lambda x.\mathsf{f}(x)))) \,/ (\mu(\lambda x.\rho(x)) \,/\, \mu(\lambda x.\mathsf{f}(x)))) \\
&\Rightarrow^* 1 \,;\, ((\mathsf{f}(\mathsf{mu}(\lambda x.\rho(x))), \rho(\mathsf{mu}(\lambda x.\mathsf{f}(x)))) \,/\, \rho(\mathsf{mu}(\lambda x.\rho(x)))) \\
&\Rightarrow \quad 1 \,;\, (\mathsf{f}(\mathsf{mu}(\lambda x.\rho(x))) \,/\, \rho(\mathsf{mu}(\lambda x.\rho(x)))) \,; \\
&\qquad\quad (\rho(\mathsf{mu}(\lambda x.\mathsf{f}(x))) \,/ (\rho(\mathsf{mu}(\lambda x.\rho(x))) \,/\, \mathsf{f}(\mathsf{mu}(\lambda x.\rho(x))))) \\
&\Rightarrow^* 1 \,;\, 1 \,;\, 1 \\
&\Rightarrow^* 1
\end{aligned}
$$

So, $\mathcal{R} \simeq \mathcal{S}$.

Example 6.5.3. Consider (again) the HRS $\mathfrak{Mu}$ from page 118. Let the following reductions be given:

$$
\begin{aligned}
\mathcal{R} &: \mathsf{mu}(\lambda x.\mathsf{f}(x)) \to \mathsf{mu}(\lambda x.\mathsf{g}(x)) \to \mathsf{g}(\mathsf{mu}(\lambda x.\mathsf{g}(x))) \\
\mathcal{S} &: \mathsf{mu}(\lambda x.\mathsf{f}(x)) \to \mathsf{f}(\mathsf{mu}(\lambda x.\mathsf{f}(x))) \to \mathsf{g}(\mathsf{mu}(\lambda x.\mathsf{f}(x))) \\
\mathcal{T} &: \mathsf{mu}(\lambda x.\mathsf{f}(x)) \to \mathsf{f}(\mathsf{mu}(\lambda x.\mathsf{f}(x))) \to \mathsf{f}(\mathsf{mu}(\lambda x.\mathsf{g}(x))) \to \mathsf{g}(\mathsf{mu}(\lambda x.\mathsf{g}(x)))
\end{aligned}
$$

The witnesses to these reductions are as follows:

$$
\begin{aligned}
\mathcal{R} &= \mathsf{mu}(\lambda x.\rho(x)) \,;\, \mu(\lambda x.\mathsf{f}(x)) \\
\mathcal{S} &= \mu(\lambda x.\mathsf{f}(x)) \,;\, \rho(\mathsf{mu}(\lambda x.\mathsf{f}(x))) \\
\mathcal{T} &= \mu(\lambda x.\mathsf{f}(x)) \,;\, \mathsf{f}(\mathsf{mu}(\lambda x.\rho(x))) \,;\, \rho(\mathsf{mu}(\lambda x.\mathsf{g}(x)))
\end{aligned}
$$

From Ex. 6.3.3 we have already learned that $\mathcal{R} \,/\, \mathcal{S} = \mathsf{f}(\mathsf{mu}(\lambda x.\rho(x)))$ and $\mathcal{S} \,/\, \mathcal{R} = 1$. Similar calculations show that $\mathcal{R} \,/\, \mathcal{T} = 1$ and $\mathcal{R} \,/\, \mathcal{R} = 1$. So, by definition

$$\mathcal{R} \lesssim \mathcal{S}, \quad \mathcal{R} \lesssim \mathcal{T}, \quad \mathcal{T} \lesssim \mathcal{R} \quad \text{and} \quad \mathcal{R} \simeq \mathcal{T}.$$

Proposition 6.5.4.

(i) The relation $\lesssim$ is a quasi-order, that is, it is reflexive and transitive.

(ii) The relation $\simeq$ is an equivalence relation, that is, it is reflexive, symmetric and transitive.

Proof. (i) Reflexivity follows directly from the residual law $\varphi \,/\, \varphi = 1$. For transitivity, assume $\varphi \,/\, \psi = 1$ and $\psi \,/\, \chi = 1$. Then

$$\varphi \,/\, \chi = (\varphi \,/\, \chi) \,/\, 1 = (\varphi \,/\, \chi) \,/ (\psi \,/\, \chi) = (\varphi \,/\, \psi) \,/ (\chi \,/\, \psi) = 1 \,/ (\chi \,/\, \psi) = 1.$$

(ii) Reflexivity follows directly from reflexivity of $\lesssim$. Symmetry and transitivity follow from the commutativity and associativity of the "and" operator in the definition of $\simeq$, respectively. □

Lemma 6.5.5. *Let $\mathcal{R}_1, \mathcal{R}_2, \mathcal{S}, \mathcal{T}$ be finite reductions. If $\mathcal{R}_1 \simeq \mathcal{R}_2$, then*

$$\mathcal{R}_1 \,;\, \mathcal{S} \,/\, \mathcal{R}_2 \,;\, \mathcal{T} = \mathcal{S} \,/\, \mathcal{T}.$$

Proof. By the following reduction:

$$\begin{aligned}
&\mathcal{R}_1 \,;\, \mathcal{S} \,/^* \mathcal{R}_2 \,;\, \mathcal{T} \\
&\Rightarrow_{\mathrm{R}} ((\mathcal{R}_1 \,;\, \mathcal{S}) \,/^* \mathcal{R}_2) \,/^* \mathcal{T} \\
&\Rightarrow_{\mathrm{R}} ((\mathcal{R}_1 \,/^* \mathcal{R}_2) \,;\, (\mathcal{S} \,/^* (\mathcal{R}_2 \,/^* \mathcal{R}_1))) \,/^* \mathcal{T} \\
&\Rightarrow_{\mathrm{R}}^* (1 \,;\, (\mathcal{S} \,/^* 1)) \,/^* \mathcal{T} \\
&\Rightarrow_{\mathrm{R}}^* \mathcal{S} \,/^* \mathcal{T}
\end{aligned}$$

and so $\mathcal{R}_1 \,;\, \mathcal{S} \,/\, \mathcal{R}_2 \,;\, \mathcal{T} = \mathcal{S} \,/\, \mathcal{T}$. □

Corollary 6.5.6. *Let $\mathcal{R}, \mathcal{S}$ be finite reductions and φ a step. If $\mathcal{R} \simeq \mathcal{S}$, then:*

(i) $\mathcal{R} \,;\, \varphi \simeq \mathcal{S} \,;\, \varphi$

(ii) $\varphi \,;\, \mathcal{R} \simeq \varphi \,;\, \mathcal{S}$

Proof. Both items follow easily from Lemma 6.5.5, the fact that $\varphi \,/\, \varphi = 1$, and the assumption that $\mathcal{R} \,/\, \mathcal{S} = 1$ and $\mathcal{S} \,/\, \mathcal{R} = 1$. □

Lemma 6.5.7. *If $(\varphi \,;\, \mathcal{R}) \lesssim \mathcal{S}$, then $\varphi \lesssim \mathcal{S}$.*

Proof. By definition, $(\varphi \,;\, \mathcal{R}) \,/\, \mathcal{S} = 1$. We must prove that $\varphi \,/\, \mathcal{S} = 1$. Suppose $\varphi \,/^* \mathcal{S} \Rightarrow_{\mathrm{R}}^* \mathcal{T}$. Then

$$(\varphi \,;\, \mathcal{R}) \,/^* \mathcal{S} \Rightarrow_{\mathrm{R}}^* (\varphi \,/^* \mathcal{S}) \,;\, (\mathcal{R} \,/^* (\mathcal{S} \,/^* \varphi) \Rightarrow_{\mathrm{R}}^* \mathcal{T} \,;\, (\mathcal{R} \,/^* (\mathcal{S} \,/^* \varphi).$$

By confluence of $\Rightarrow_{\mathrm{R}}$ and the assumption that $(\varphi \,;\, \mathcal{R}) \,/\, \mathcal{S} = 1$, it follows that

$$\mathcal{T} \,;\, (\mathcal{R} \,/^* (\mathcal{S} \,/^* \varphi) \Rightarrow_{\mathrm{R}}^* 1$$

which is only possible if $\mathcal{T} \Rightarrow_{\mathrm{R}}^* 1$, as desired. □

Proposition 6.5.8. *Let φ, ψ be proof terms.*

(i) If $\varphi \lesssim \psi$ then $f(\varphi) \lesssim f(\psi)$, $\lambda x.\varphi \lesssim \lambda x.\psi$ and $\rho(\varphi) \lesssim \rho(\psi)$.

(ii) If $\varphi \simeq \psi$ then $f(\varphi) \simeq f(\psi)$, $\lambda x.\varphi \simeq \lambda x.\psi$ and $\rho(\varphi) \simeq \rho(\psi)$.

Proof. (i) Assume $\varphi \,/\, \psi = 1$. Then $f(\varphi) \,/\, f(\psi) = 1$ by a single application of the R_f rule, $\lambda x.\varphi \,/\, \lambda x.\psi = 1$ by a single application of the R_λ rule and $\rho(\varphi) \,/\, \rho(\psi) = 1$ by a single application of the R_ρ rule.

(ii) Follows directly from (i) and the definition of $\simeq$. □

Proposition 6.5.9. *If $\varphi \simeq \psi$, then $\mathrm{src}(\varphi) = \mathrm{src}(\psi)$ and $\mathrm{tgt}(\varphi) = \mathrm{tgt}(\psi)$.*

Proof. The condition on the sources of the steps follows directly because we assume (implicitly) that $\varphi \simeq \psi$ is well-defined. The condition on the targets of the steps follows from the fact that, by definition,

$$\mathrm{tgt}(\varphi) = \mathrm{src}(\varphi \,/\, \psi) = \mathrm{src}(\psi \,/\, \varphi) = \mathrm{tgt}(\psi).$$

□

6.6 Equivalence of projection and permutation equivalence

In the previous chapter, the fact that standardization equivalence and permutation equivalence are the same was a trivial corollary of the Standardization Theorem. The equivalence of projection equivalence and permutation equivalence is more cumbersome to show. One direction is easy:

Proposition 6.6.1. *Let $\mathcal{R}, \mathcal{S}$ be reductions. If $\mathcal{R} \approx \mathcal{S}$, then $\mathcal{R} \simeq \mathcal{S}$.*

Proof. By induction on the derivation of $\mathcal{R} \approx \mathcal{S}$. The induction steps (reflexivity, symmetry, transitivity and context) follow from Prop. 6.5.4 and Cor. 6.5.6. The base case follows from the fact that $\mathcal{L} \,/\, \mathcal{R} = 1$ and $\mathcal{R} \,/\, \mathcal{L} = 1$ for all equations $\mathcal{L} \approx \mathcal{R}$ of Def. 3.2.1. □

For the other direction, we want to use standardization equivalence. First, we give some evidence that the claim is actually true.

Example 6.6.2. Consider the reductions $\mathcal{R}$ and $\mathcal{S}$ from Ex. 6.5.2. We have already shown that $\mathcal{R} \simeq \mathcal{S}$. We calculate $\mathit{Std}(\mathcal{R})$ and $\mathit{Std}(\mathcal{S})$:

$$\begin{aligned}
&\mathcal{R} = \mu(\lambda x.\rho(x)) \\
&\Rightarrow_{(\text{flat})} \mu(\lambda x.\mathsf{f}(x)), \rho(\mathsf{mu}(\lambda x.\mathsf{f}(x))) \\
&\Rightarrow_{(\text{flat})} \mu(\lambda x.\mathsf{f}(x)), \rho(\mathsf{mu}(\lambda x.\mathsf{f}(x))), \mathsf{g}(\mathsf{mu}(\lambda x.\rho(x)))
\end{aligned}$$

$$\begin{aligned}
&\mathcal{S} = \mu(\lambda x.\mathsf{f}(x)), \mathsf{f}(\mathsf{mu}(\lambda x.\rho(x))), \rho(\mathsf{mu}(\lambda x.\mathsf{g}(x))) \\
&\Rightarrow_{(\text{std})} \mu(\lambda x.\mathsf{f}(x)), \rho(\mathsf{mu}(\lambda x.\mathsf{f}(x))), \mathsf{g}(\mathsf{mu}(\lambda x.\rho(x)))
\end{aligned}$$

Since $\mathit{Std}(\mathcal{R}) = \mathit{Std}(\mathcal{S})$, we conclude that $\mathcal{R} \equiv \mathcal{S}$.

We need to prove a few auxiliary properties on the interplay between projection and standardization.

Lemma 6.6.3. *Let φ, ψ be compatible coinitial proper steps which contract redexes at positions p, q, respectively, where $p <_{\text{lex}} q$. Then $\varphi \,/\, \psi$ is a proper step and $\mathcal{RP}os(\varphi \,/\, \psi) = \mathcal{RP}os(\varphi)$.*

Proof. By structural induction on the source of φ and ψ. We distinguish the following cases:

- Suppose $\varphi = \rho(s_1, \ldots, s_n)$ and $\psi = l(s_1, \ldots, \varphi_k, \ldots, s_n)$, for a rule $\rho : l \to r$ and multistep $\varphi_k : s_k \to t_k$. Then $\varphi \,/\, \psi = \rho(s_1, \ldots, t_k, \ldots, s_n)$, which satisfies the conditions of the lemma.
- Suppose $\varphi = f(s_1, \ldots, \varphi_{k_1}, \ldots, s_n)$ and $\psi = f(s_1, \ldots, \varphi_{k_2}, \ldots, s_n)$, where $k_1 < k_2$. Then:

  $$\varphi \,/\, \psi = f(s_1, \ldots, \varphi_{k_1}, \ldots, t_{k_2}, \ldots, s_n).$$

  which satisfies the conditions of the lemma.

- Suppose $\varphi = f(s_1, \ldots, \varphi_k, \ldots, s_n)$ and $\psi = f(s_1, \ldots, \psi_k, \ldots, s_n)$. Then the lemma follows directly from the induction hypothesis applied to φ_k and ψ_k.

Given that $p <_{\text{lex}} q$, there are no other possibilities. □

Lemma 6.6.4. *Let φ, ψ be proper steps contracting redexes at positions p, q, respectively, where $p <_{\text{lex}} q$, and let $\mathcal{R}$ be a proper reduction. If $\mathcal{R}$ is standard for φ, then $\mathcal{R} / \psi$ is standard for φ.*

Proof. If there is no step in $\mathcal{R}$ contracting a redex to the left of the redex of φ, then it follows from Lemma 6.6.3 that there is no such step in $\mathcal{R} / \psi$, either.

Otherwise, let χ be the first step of $\mathcal{R}$ which contracts a redex at position r such that $r <_{\text{lex}} p$. Let $\mathcal{R} = \mathcal{S} ; \varphi ; \mathcal{T}$. By transitivity it holds that $r <_{\text{lex}} q$, and thus it follows from Lemma 6.6.3 that

$$\mathcal{RP}os(\chi / (\psi / \mathcal{S})) = \mathcal{RP}os(\chi) \ni p$$

as required. □

Proposition 6.6.5. *Let $\mathcal{R}, \mathcal{S}$ be finite, standard reductions. If $\mathcal{R} \simeq \mathcal{S}$ then $\mathcal{R} = \mathcal{S}$.*

Proof. Let $\mathcal{R} = \varphi ; \mathcal{R}_0$ and $\mathcal{S} = \psi ; \mathcal{S}_0$, where φ contracts a redex at position p and ψ a redex at position q. Assume, without loss of generality, that $p \leq_{\text{lex}} q$. First, we prove, by induction on p and a nested induction on the length of $\mathcal{S}$, that $\varphi = \psi$.

Assume, to the contrary, that $\varphi \neq \psi$ (in other words, that $p <_{\text{lex}} q$). If the redex pattern of ψ overlaps the redex pattern of φ, then φ / ψ is not compatible, contradicting the implicit assumption that it $\mathcal{R}$ and $\mathcal{S}$ are. So, the redex patterns of φ and ψ do not overlap.

Let $\mathcal{S}_0 = \psi_1, \ldots, \psi_m$, where each step ψ_i contracts a redex at position q_i, respectively. Furthermore, let k be the first index such that $q_k <_{\text{lex}} q$. If there no such k, then, by transitivity $p <_{\text{lex}} q_i$ for all $1 \leq i \leq m$ and it follows from Lemma 6.6.3 that $\varphi / \mathcal{S}$ is a proper step, and thus not empty. However, by Lemma 6.5.7 and the fact that $\mathcal{R} \simeq \mathcal{S}$ by assumption, it must be the case $\varphi / \mathcal{S}$. This yields a contradiction.

So there must be such a k. By transitivity of $<_{\text{lex}}$ it follows that

$$p <_{\text{lex}} q_i \text{ for all } i < k. \tag{$\star$}$$

Also, because $\mathcal{S}$ is standard, by definition it must be the case that

$$q \in \mathcal{RP}os(\psi_k). \tag{$\star\star$}$$

Let $\mathcal{T} = \psi, \psi_1, \ldots, \psi_{k-1}$ and $\mathcal{U} = \psi_k, \ldots, \psi_{n-1}$ (that is $\mathcal{S} = \mathcal{T} ; \mathcal{U}$). Furthermore, let $\mathcal{R}' = \mathcal{R} / \mathcal{T}$. By ($\star$) and Lemma 6.6.3, the first step of $\mathcal{R}'$ is a proper step φ' such that $\mathcal{RP}os(\varphi') = \mathcal{RP}os(\varphi)$.

From Prop. 6.6.1 and Theorem 5.5.1 it follows that $Std(\mathcal{R}') \simeq \mathcal{R}'$. From Lemma 6.5.5 it follows that $\mathcal{R}' = \mathcal{R} \,/\, \mathcal{T} \simeq \mathcal{S} \,/\, \mathcal{T} = \mathcal{U}$. Together, this yields that $Std(\mathcal{R}') \simeq \mathcal{U}$.

We consider the following two cases:

- If $p <_{\text{lex}} q_k$, then the nested induction hypothesis applies to $Std(\mathcal{R}')$ and $\mathcal{U}$, because the first step of $Std(\mathcal{R}')$ contracts the same redex patterns as φ, and $\mathcal{U}$ is strictly shorter than $\mathcal{S}$.
- If $q_k <_{\text{lex}} p$, then the induction hypothesis applies to $\mathcal{U}$ and $Std(\mathcal{R}')$.

In both cases it follows, by the induction hypothesis, that the first steps of $Std(\mathcal{R}')$ and $\mathcal{U}$ are equal.

Since $\mathcal{R}$ is standard and all steps of $\mathcal{T}$ occur 'above' the first step φ of $\mathcal{R}$, it is a result of Lemma 6.6.4 that φ', the first step of $\mathcal{R}'$, is also the first step of $Std(\mathcal{R}')$. Since ψ_k is the first step of $\mathcal{U}$, this means that $\varphi' = \psi_k$. Together with $(\star)$ it follows that $q \in \mathcal{RP}os(\varphi)$, and thus that φ and ψ overlap, contradicting an earlier conclusion. So, we have to retract the assumption that $\varphi \neq \psi$ and accept $\varphi = \phi$.

From this it follows by a simple induction on the length of $\mathcal{R}$ and $\mathcal{S}$ that $\mathcal{R} = \mathcal{S}$. □

Now, we have dealt with the preliminary work, and the main result of this section is easily proved:

Theorem 6.6.6. *Let $\mathcal{R}, \mathcal{S}$ be reductions. $\mathcal{R} \simeq \mathcal{S}$ if and only if $\mathcal{R} \approx \mathcal{S}$.*

Proof. ($\Rightarrow$): Suppose $\mathcal{R} \simeq \mathcal{S}$. We assume, in order to derive a contradiction, that $\mathcal{S} \not\approx \mathcal{R}$. By Theorem 5.5.1, we may assume, without loss of generality, that $\mathcal{R}$ and $\mathcal{S}$ are different standard reductions, but this contradicts Prop. 6.6.5.

($\Leftarrow$): By Prop. 6.6.1. □

6.7 Related work

In the literature, residuals have been studied in various degrees of abstraction and for various forms of reduction, such as β-reduction in the λ-calculus, first-order term rewriting and concurrency theory. Below we review some related work on the subject.

As noted in the introduction to this chapter, in general two approaches to residual theory can be identified. The first relates specific redexes in the source and target of a step to each other. This is similar to the trace relation of Chapter 2 (in particular, Defs. 2.3.9 and 2.4.23). The second approach considers steps (and reductions) as a whole, and focusses on what remains of a step/reduction after some other reduction has been performed. We will call these approaches the *redex approach* and *step approach*, respectively. The two approaches are, of course, very much related by the observation that a step is performed by contracting a (set of) redex(es).

The redex approach. This approach is originally due to Lévy [29], who used it to define permutation equivalence for the λ-calculus. The work of Lévy was applied to TRSs by Huet & Lévy [19] and axiomatically generalized by Melliès [32]. The works mentioned above have in common that, for each redex v, a residual relation $u \,[\![v]\!]\, u'$ is defined on redexes, denoting that redex u' is the residual of u after v.

Related is the work of Laneve & Montanari [28], who give an axiomatic treatment of permutation equivalence by using residuals. They apply this to TRSs, and also to CRSs, a higher-order rewriting paradigm related to HRSs, by translating CRSs to TRSs. In this dissertation, however, residuals are defined for HRSs directly.

The step approach. The step approach originated with the work of Stark [51]. Stark gives all of the abstract axioms presented here, but also requires the projection order to be antisymmetric: if $\varphi \lesssim \varphi$ and $\varphi \lesssim \varphi$, then $\varphi = \varphi$. This is an undesired propery for most forms of rewriting. Consider for example the TRS:

$$\begin{aligned} \rho: &\quad \mathsf{f}(x) \to \mathsf{c} \\ \theta: &\quad \mathsf{a} \to \mathsf{b} \end{aligned}$$

Now, it is the case that $\rho(\theta) \lesssim \rho(\mathsf{a})$ and $\rho(\mathsf{a}) \lesssim \rho(\theta)$. It is undesired, however, to pose that $\rho(\theta) = \rho(\mathsf{a})$ (although, of course, they are, by definition, projection equivalent).

Van Oostrom & De Vrijer [42, 43] remove this axiom from Stark's framework and show that much of the theory goes through anyway. In this chapter, we basically show that HRSs satisfy their axioms. (But, as shown above, they do *not* satisfy Stark's axioms.)

Other. Without discussion, we would like to mention [18] and [23]. The first article gives a formal treatment in the proof assistent Coq of residuals in the λ-calculus, while the second is an axiomatic approach to residual theory for conflict-free rewrite systems.

6.8 Discussion

In this chapter we formalized the notions of *residual*, *projection* and *projection equivalence* for higher-order rewrite systems. Our approach was to build up the theory from an abstract point of view, and then define operations on higher-order proof terms which satisfy the laws of the abstact theory.

The formalizations, however, are currently restricted to *orthogonal* HRSs. There are two natural extensions to the theory to generalize some of the results to non-orthogonal HRSs:

- Drop the requirement that the projection operator be *total* on pairs of coinitial steps. This makes the theory more cumbersome to formulate,

because we have to keep track which occurrences of $\varphi \,/\, \psi$ are defined and which are not.

- Add an 'error symbol' to a residual system. This approach is followed, for the first-order case, in [53], and is easily adaptable to higher-order rewriting. The idea is that, for non-compatible proof terms φ, ψ, the projection term $\varphi \,/\, \psi$ simplifies to a proof term containing this error symbol, expressing the fact that the two proof terms are not compatible.

In both cases, of course, the confluence results cannot be directly generalized. The notion of projection equivalence (and accompanying projection order), however, can.

Also, we restricted our attention to *finite* reductions. This was not without reason: it is unclear how to extend projection to infinite reductions. For example, what is $\mathcal{S} \,/\, \mathcal{R}$ supposed to be if $\mathcal{R}$ is an infinite reduction? Abstract residual theory requires that $\mathrm{src}(\mathcal{S} \,/\, \mathcal{R}) = \mathrm{tgt}(\mathcal{R})$, but $\mathrm{tgt}(\mathcal{R})$ is undefined because $\mathcal{R}$ is infinite. It is possible to extend the notion of projection equivalence to infinite reductions by looking at their finite prefixes, just as we did for permutation equivalence in Def. 3.3.1.

Seven

Results

7.1 Summary

We quickly summarize the various notions of equivalence of reductions formalized in this dissertation:

Permutation equivalence. Permutation equivalence is the formalization of the intuition that two reductions are equivalent if the one can be obtained from the other by iteratively permuting steps. It is formalized by the meta-rewriting system of page 41. We proved the important property that every finite reduction (possibly containing multisteps) has an equivalent proper reduction. Additionally, for every infinite reduction there exists an infinite proper reduction with the same source.

Standardization equivalence. Standardization equivalence formalizes the idea that two reductions are equivalent if they have the same standard reduction. For this it is required that each permutation equivalence class of reductions contains a unique standard reduction. This fact is proved in the Standardization Theorem (Theorem 5.5.1).

The Standardization Theorem is proved by giving two procedure which produce an equivalent standard reduction when given an arbitrary reduction: selection standardization and inversion standardization, which correspond to weak and strong standardization of [25], respectively.

The fact that permutation equivalence and standardization equivalence are the same is simple corollary of the fact that the inversion standardization is defined by means of a meta-rewrite system on reductions, of which convertibility coincides with permutation equivalence.

Projection equivalence. If $\mathcal{R}$ and $\mathcal{S}$ are reductions, then $\mathcal{R} \,/\, \mathcal{S}$ represents the reduction which contains the steps of $\mathcal{R}$ except the one which were also part of $\mathcal{S}$. Projection equivalence captures the idea that two reductions are equivalent if the one projected over the other yields an empty reduction.

The projection operator, and as a result also projection equivalence, are only defined for orthogonal reductions. It is proved that, for orthogonal reductions, projection equivalence coincides with permutation and standardization equivalence.

Two procedures were given to standardize a given reduction. Also, a procedure was given to calculate the residual of one procedure after another. This fact makes the notions of standardization equivalence and projection equivalence decidable. By the main result of this thesis, the fact that all three notions of equivalence of reductions are the same, this result can be extended to permutation equivalence.

We briefly investigated how permutation equivalence and the notion of standardization can be extended to infinite reductions. In both cases, we did this by considering infinite reductions as the limit of their finite prefixes. The conclusion is that the notion of permutation equivalence can be extended to infinite reductions. However, since uniqueness of standard reductions is lost by our standardization procedure for infinite reductions, standardization cannot be used to decide equivalence in the infinite case. However, infinite standardization may be useful in other cases. For example, it was used in [24] to prove a preservation of strong normalization property.

The projection operator and projection equivalence are limited to orthogonal reductions. Extending the projection operator to non-orthogonal reduction can be done by including an error symbol, but this route has not been extensively investigated here. The definitions of permutation equivalence and standardization also work for non-orthogonal reductions.

An important auxiliary result of this dissertation, proved in Chapter 4, is the proof that HRSs enjoy the Finite Family Developments property. This property is used to define the standardization procedures of Chapter 5 and has possible applications for proving termination of HRSs.

7.2 Main result

The main result of this thesis, the equivalence of permutation equivalence (Def. 3.2.1), standardization equivalence (Def. 5.5.2) and projection equivalence (Def. 6.5.1) is a direct corollary of results proved in earlier chapters:

Theorem 7.2.1. *For finite, orthogonal reductions, permutation equivalence, standardization equivalence and projection equivalence coincide.*

Proof. Equivalence of permutation and projection equivalence is proved in Theorem 6.6.6 and equivalence of standardization equivalence and permutation equivalence is proved in Corollary 5.5.3. □

Note that the permutation and standardization equivalence relations are not restricted to orthogonal reductions. That they are the same for all reductions was proven in Corollary 5.5.3. Extending projection terms with an error

symbol, as suggested in Sect. 6.8, would yield an equivalent notion of projection equivalence which is not restricted to orthogonal reductions.

It has been suggested that projection equivalence can be trivially extended to non-orthogonal reductions, simply by defining that non-orthogonal reductions are not projection equivalent. Such a notion of projection equivalence, however, would not be equivalent to the other two notions. Consider the following TRS:

$$\begin{aligned} \mathsf{a} &\to \mathsf{b} \\ \mathsf{a} &\to \mathsf{c} \\ \mathsf{e}(x) &\to \mathsf{d} \end{aligned}$$

and the following two reductions:

$$\begin{aligned} \mathcal{R} &: \mathsf{e}(\mathsf{a}) \to \mathsf{e}(\mathsf{b}) \to \mathsf{d} \\ \mathcal{R} &: \mathsf{e}(\mathsf{a}) \to \mathsf{e}(\mathsf{c}) \to \mathsf{d} \end{aligned}$$

The two reductions are not orthogonal, but they are equivalent because the non-orthogonal part is erased by the last step.

Bibliography

[1] Thomas Arts and Jürgen Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236(1–2):133–178, 2000.

[2] Andrea Asperti, Paolo Coppola, and Simone Martini. (optimal) duplication is not elementary recursive. *Information and Computation*, 193(1):21–56, 2004.

[3] Andrea Asperti and Stefano Guerrini. *The Optimal Implementation of Functional Programming Languages*, volume 45 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1998.

[4] H.P. Barendregt. *The Lambda Calculus, Its Syntax and Semantics, Revised Edition*. North-Holland, 1984.

[5] Inge Bethke, Jan Willem Klop, and Roel de Vrijer. Descendants and origins in term rewriting. *Information and Computation*, 159(1–2):59–124, 2000.

[6] Frédéric Blanqui. Higher-order dependency pairs. In *Proceedings of WST*, 2006.

[7] Roel Bloo. *Preservation of Termination for Explicit Substitution*. PhD thesis, Technische Universiteit Eindhoven, 1997.

[8] H. J. Sander Bruggink. Residuals in higher-order rewriting. In *Proceedings of RTA 2003*. Springer, 2003.

[9] H. J. Sander Bruggink. A proof of finite family developments for higher-order rewriting using a prefix property. In *Proceedings of RTA 2006*, 2006.

[10] H. B. Curry and J. Feys. *Combinatory Logic*. North Holland, 1958.

[11] D.T. van Daalen. *The language theory of Automath*. PhD thesis, Technische Universiteit Eindhoven, 1980.

[12] Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Higher order unification via explicit substitutions. *Information and Computation*, 157(1–2):184–233, 2000.

[13] Alfons Geser, Dieter Hofbauer, and Johannes Waldmann. Match-bounded string rewriting. *Applicable Algebra in Engineering, Communication and Computing*, 15(3–4):149–171, 2004.

[14] Alfons Geser, Dieter Hofbauer, Johannes Waldmann, and Hans Zantema. On tree automata that certify termination of left-linear term rewriting systems. *Information and Computation*, 205, 2007.

[15] Juergen Giesl, René Thiemann, and Peter Schneider-Kamp. Proving and disproving termination of higher-order functions. In *Proceedings of FroCoS'05*. Springer, 2005.

[16] Georges Gonthier, Jean-Jacques Lévy, and Paul-André Melliès. An abstract standardisation theorem. In *Proceedings of the 8th Annual Symposium on Logic in Computer Science*, 1992.

[17] Barnaby P. Hilken. Towards a proof theory of rewriting: the simply typed 2λ-calculus. *Theoretical Computer Science*, 170:407–444, 1996.

[18] Gérard Huet. Residual theory in λ-calculus: a formal development. *J. of Functional Programming*, 4(3):371–394, 1994.

[19] Gérard Huet and Jean-Jacques Lévy. Computations in orthogonal rewriting systems, part I + II. In J.L. Lassez and G.D. Plotkin, editors, *Computational Logic – Essays in Honor of Alan Robinson*, chapter 11 + 12, pages 395–443. MIT Press, 1991.

[20] J.M.E. Hyland. A syntactic characterization of the equality in some models of the λ-calculus. *Journal of the London Mathematical Society*, 12(2):361–370, 1976.

[21] Jean-Pierre Jouannaud and Albert Rubio. Higher-order recursive path orderings "á la carte". In *Proceedings of WST*, 2001.

[22] Zurab Khasidashvili and John Glauert. Discrete normalization and standardization in deterministic residual structures. Technical Report SYS-C96-06, UEA Norwich, 1996.

[23] Zurab Khasidashvili and John Glauert. Relating conflict-free stable transition systems and event models via redex families. *Theoretical Computer Science*, 286(1):65–95, 2002.

[24] Zurab Khasidashvili, Mizuhito Ogawa, and Vincent van Oostrom. Uniform normalisation beyond orthogonality. In *Proceedings of RTA'01*. Springer, 2001.

[25] J. W. Klop. *Combinatory Reduction Systems*. PhD thesis, Utrecht University, 1980.

[26] Jan Willem Klop, Vincent van Oostrom, and Femke van Raamsdonk. Combinatory Reduction Systems: Introduction and survey. *Theoretical Computer Science*, 121(1-2):279–308, 1993.

[27] John Lamping. An algorithm for optimal lambda calculus reduction. In *17th ACM Symposium on Principles of Programming Languages*. ACM Press, 1990.

[28] Cosimo Laneve and Ugo Montanari. Axiomatizing permutation equivalence. *Mathematical Structures in Computer Science*, 6(3):219–215, 1996.

[29] Jean-Jacques Lévy. *Réductions correctes et optimales dans le λ-calcus*. PhD thesis, Université Paris VII, 1978.

[30] Luc Maranget. Optimal derivations in weak lambda-calculi and in orthogonal term rewriting systems. In *Principles of Programming Languages*, 1991.

[31] Richard Mayr and Tobias Nipkow. Higher-order rewrite systems and their confluence. *Theoretical Computer Science*, 192:3–29, 1998.

[32] Paul-André Melliès. Axiomatic rewriting theory VI: Residual theory revisited. In Sophie Tison, editor, *Rewriting Techniques and Applications 2002*, pages 24–50, 2002.

[33] Paul-André Melliès. Axiomatic rewriting theory I: A diagrammatic standardization theorem. In Aart Middeldorp, Vincent van Oostrom, Femke van Raamsdonk, and Roel de Vrijer, editors, *Processes, Terms and Cycles: Steps on the road to infinity*, pages 554–638. Springer, 2005.

[34] José Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96:73–155, 1992.

[35] Dale Miller. A logic programming language with lambda abstraction, function variables and simple unification. *Journal of Logic and Computation*, 1(4), 1991.

[36] Tobias Nipkow. Higher-order critical pairs. In *Proceedings on the 6th IEEE Symposium on Logic in Computer Science*. Springer, 1991.

[37] Tobias Nipkow. Orthogonal higher-order rewrite systems are confluent. In *Proceedings on the 1st International Conference of Typed Lambda Calculi and Applications*. Springer, 1993.

[38] Tobias Nipkow and Christian Prehofer. Higher-order rewriting and equational reasoning. In W. Bibel and P. Schmitt, editors, *Automated Deduction — A basis for Applications, Volume Foundations*. Kluwer, 1998.

[39] Vincent van Oostrom. *Confluence for Abstract and Higher-Order Rewriting*. PhD thesis, VU Amsterdam, 1994.

[40] Vincent van Oostrom. Higher-order families. In *Proceedings of RTA 1996*. Springer, 1996.

[41] Vincent van Oostrom. Finite family developments. In *Proceedings of RTA 1997*. Springer, 1997.

[42] Vincent van Oostrom and Roel de Vrijer. Four equivalent equivalences of reductions. *ENTCS*, 70(6), 2002.

[43] Vincent van Oostrom and Roel de Vrijer. *Equivalence of Reductions*, chapter 8 of [53]. 2003.

[44] G. D. Plotkin. Call-by-name, call-by-value and the λ-calculus. *Theoretical Computer Science*, 1, 1975.

[45] Femke van Raamsdonk. On termination of higher-order rewriting. In *Proceedings of RTA 2001*. Springer, 2001.

[46] Masahiko Sakai and Keiichirou Kusakari. On dependency pair method for proving termination of higher-order rewrite systems. *IEICE Transactions on Information and Systems*, E88-D(3), 2005.

[47] Masahiko Sakai, Yoshitsugu Watanabe, and Toshiki Sakabe. An extension of the dependency pair method for proving termination of higher-order rewrite systems. *IEICE Transactions on Information and Systems*, E84-D(8), 2001.

[48] John Staples. Computation on graph-like expressions. *Theoretical Computer Science*, 10(2), 1985.

[49] John Staples. Optimal evaluations of graph-like expressions. *Theoretical Computer Science*, 10(3), 1985.

[50] John Staples. Speeding up subtree replacement systems. *Theoretical Computer Science*, 11(1), 1985.

[51] Eugene W. Stark. Concurrent transition systems. *Theoretical Computer Science*, 64(3):221–269, 1989.

[52] W. W. Tait. Intensional interpretation of functionals of finite type I. *Journal of Symbolic Logic*, 32:198–212, 1967.

[53] TeReSe. *Term Rewriting Systems*. Number 55 in CTTCS. Cambridge University Press, 2003.

[54] C. P. Wadsworth. The relation between computational and denotational properties for Scott's D_∞-models of the λ-calculus. *SIAM Journal on Computing*, 5:488–521, 1976.

[55] J.B. Wells and Robert Muller. Standardization and evaluation in combinatory reduction systems. See: `http://www.macs.hw.ac.uk/~jbw/` or `http://www.cs.bc.edu/~muller`, 2000.

[56] D.A. Wolfram. *The Clausal Theory of Types*, volume 21 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1993.

[57] Hans Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24:89–105, 1995.

Nederlandse samenvatting

Van rekenen tot hogere-orde herschrijven

Het spannende scenario van dit proefschrift speelt zich af op het gebied van *hogere-orde termherschrijven*. Laat ik, voordat ik dieper inga op de eigenlijke inhoud van het proefschrift, de lezer eerst kort, op een informele manier, laten kennismaken met wat, in het algemeen, *herschrijven* eigenlijk inhoudt.

Herschrijven is een wiskundig model (een versimpelde weergave van de werkelijkheid) van hoe berekeningen uitgevoerd worden. Berekeningen worden tegenwoordig meestal uitgevoerd met behulp van een computerprogramma en herschrijftheorie kan gebruikt worden om die computerprogramma's beter te maken, bijvoorbeeld door te bewijzen dat ze altijd het goede antwoord op een berekening geven, of door te bewijzen dat ze altijd een antwoord geven (en dus niet vastlopen).

Hoe gaat herschrijven in zijn werk? Op de basisschool hebben we geleerd rekenkundige sommen uit te rekenen, en wel in stapjes. Als we bijvoorbeeld de som $(1+3)\times(4+1)$ moesten uitrekenen, konden we dat als volgt doen (ik schrijf $\to$ in plaats van $=$, omdat de opgave de hele tijd vereenvoudigd wordt):

$$\begin{aligned}(1+3)\times(4+1) &\to 4\times(4+1)\\ &\to 4\times 5\\ &\to 20\end{aligned}$$

Je ziet hoe we in een ingewikkelde opgave op zoek gaan naar simpele deelproblemen die we direct op kunnen lossen. Door deze deelproblemen uit te rekenen en te vervangen (te herschrijven) door hun antwoord, ontstaan weer nieuwe deelproblemen, die we ook weer uit kunnen rekenen. Zo komen we stap-voor-stap op het uiteindelijk antwoord. Dit is herschrijven – of preciezer *termherschrijven*. Een berekening zoals hierboven wordt een *reductie* genoemd, omdat we de term bij elke stap versimpelen (reduceren).

Drie belangrijke algemene eigenschappen van herschrijven kunnen we al in het bovenstaande voorbeeld ontdekken:

- Herschrijfstappen zijn plaatselijk. In de eerste stap van het bovenstaande voorbeeld herschreven we $(1 + 3)$ naar 4, maar de rest van de som veranderde niet.

- Herschrijven is niet deterministisch. In het voorbeeld hierboven hebben we ervoor gekozen om eerst $1 + 3$ uit te rekenen, en daarna pas $4 + 1$, maar we hadden er net zo goed voor kunnen kiezen om eerst $4 + 1$ en daarna $1 + 3$ uit te rekenen.
- Het is echter niet altijd zo dat we stappen zomaar kunnen omdraaien. Soms moeten we, voordat we een bepaalde stap kunnen doen, eerst een aantal andere stappen doen. In de bovenstaande berekening kon bijvoorbeeld de stap van (4×5) naar 20 niet eerder worden gedaan om de eenvoudige reden dat we toen 4 en 5 nog niet hadden uitgerekend.

Eerste-orde termherschrijven. Tot nu toe zijn we nog weinig in details getreden. Laten we nu iets preciezer definiëren wat termherschrijven precies is. We beginnen met de meest eenvoudige vorm van termherschrijven, namelijk eerste-orde termherschrijven.

In eerste-orde termherschrijven herschrijven we zogenaamde *termen*. In het voorbeeld op de vorige bladzijde herschreven we ook termen. We zagen daar dat een term ofwel een getal was, ofwel een optelling van de vorm $s + t$, ofwel een vermenigvuldiging van de vorm $s \times t$, waarbij s en t ook weer termen waren. We maken nu de notie "term" algemeen, zodat we er ook andere dingen mee kunnen beschrijven dan getallen, optellingen en vermenigvuldigingen.

Een term zoals die in eerste-orde herschrijven wordt gebruikt, is ofwel een constante (bijvoorbeeld een getal, of een letter a, b of c), ofwel een variabele (die we als x, y of z schrijven), ofwel een samengestelde term van de vorm $f(t_1, \ldots, t_n)$, waarbij f een functiesymbool is en $t_1, \ldots, t_n$ ook weer termen. Zo zijn $\mathsf{f}(\mathsf{g}(\mathsf{a}, x))$ en $\mathsf{g}(x, y)$ bijvoorbeeld termen.[1]

Een eerste-orde termherschrijfsysteem bestaat nu uit een aantal *herschrijfregels* van de vorm $l \to r$, waarbij er in r geen variabelen mogen voorkomen die niet ook in l voorkomen. De regel $l \to r$ betekent dat de term l door de term r vervangen mag worden.

We kunnen een herschrijfregel op een term toepassen als de linkerkant van de regel ergens in de term voorkomt (waarbij variabelen door een willekeurige term mogen worden vervangen). Het toepassen van een herschrijfregel op een term werkt nu als volgt: we vervangen simpelweg de linkerkant van de regel door de rechterkant, waarbij de variabelen in de rechterkant door dezelfde termen vervangen moeten worden als in de linkerkant.

Beschouw als voorbeeld het termherschrijfsysteem dat bestaat uit de volgende vier herschrijfregels:

$$\begin{aligned} &\text{regel 1}: && \mathsf{f}(x) \to \mathsf{g}(x, x) \\ &\text{regel 2}: && \mathsf{g}(\mathsf{b}, x) \to x \\ &\text{regel 3}: && \mathsf{a} \to \mathsf{b} \end{aligned}$$

[1] Merk op dat we, volgens deze regels, dus eigenlijk niet, zoals eerder, $x + y$ mogen schrijven, maar $+(x, y)$. Voor de duidelijkheid zal ik echter gewoon $2 + 3$ en 4×5 blijven schrijven.

We kunnen dan de volgende reductie uitvoeren:

$$\begin{aligned}
\mathsf{f}(\mathsf{a}) &\to \mathsf{g}(\mathsf{a},\mathsf{a}) && \text{(pas regel 1 toe, met } x \text{ vervangen door } \mathsf{a}\text{)}\\
&\to \mathsf{g}(\mathsf{b},\mathsf{a}) && \text{(pas regel 3 toe)}\\
&\to \mathsf{a} && \text{(pas regel 2 toe, met } x \text{ vervangen door } \mathsf{a}\text{)}\\
&\to \mathsf{b} && \text{(pas regel 3 toe)}
\end{aligned}$$

Als we bij de term b zijn aanbeland, kunnen we geen herschrijfstap meer verrichten. Deze term wordt een *normaalvorm* van het herschrijfsysteem genoemd. In de meeste gevallen komt de normaalvorm overeen met het antwoord van de berekening.

Door in het voorbeeld termen te gebruiken die voor de gemiddelde persoon geen betekenis hebben, wil ik aangeven dat de herschrijver in principe niet geïnteresseerd is in de betekenis van een term: herschrijven is een volkomen *syntactische* bezigheid. We voeren slechts een trucje uit met letters en cijfers, ofwel met symbolen – net als een computer, eigenlijk. Doordat we ons niet met de betekenis van termen bezighouden, is de theorie breed toepasbaar. Sterker nog, door af en toe expres de 'verkeerde' betekenis aan termen toe te kennen, kunnen we bepaalde eigenschappen van herschrijfsystemen bewijzen, bijvoorbeeld dat met een gegeven termherschrijfsysteem elke berekening op een gegeven moment stopt, of juist niet.

Hogere-orde termherschrijven. Met eerste-orde termherschrijven is het mogelijk om alles te berekenen wat je ook met een computer kunt berekenen. We kunnen er rekenkundige sommen mee uitrekenen, een lijst mee sorteren, de snelste route van Duisburg naar Utrecht mee berekenen, enzovoort. Dit lijkt misschien een reden om te denken dat we geen uitbreidingen van eerste-orde termherschrijven nodig hebben – we kunnen er immers alles al mee! Echter, het feit dat we elke berekenbare functie uit kunnen rekenen, betekent nog niet dat die berekening ook op een duidelijke en elegante manier gebeurt.

De termen in eerste-orde termherschrijen verwijzen altijd naar objecten, bijvoorbeeld getallen. De termen 3×2 en $(1+2)\times(1+1)$ refereren bijvoorbeeld beide naar het getal 6. In veel 'echte' programmeertalen kunnen variabelen echter ook naar functies verwijzen, bijvoorbeeld de functie die twee getallen neemt en deze bij elkaar optelt. Omdat dat in eerste-orde herschrijven niet mogelijk is, is het interessant om een uitbreiding ervan te onderzoeken waarin dat wel kan. Deze uitbreiding heet hogere-orde termherschrijven, en deze vorm van termherschrijven is het onderwerp van dit proefschrift.

Termen in hogere-orde termherschrijven zijn van een veel ingewikkeldere vorm dan termen in eerste-orde termherschrijven. Om termen op te kunnen schrijven die naar functies verwijzen gebruiken we de λ-notatie die door Church in de jaren 30 van de 20$^{\text{e}}$ eeuw werd verzonnen: termen van de vorm $\lambda x.t$, waarbij x een variabele is en t een term, moeten naar functies verwijzen. Hoe hogere-orde termherschrijven precies werkt, zal ik hier niet uitleggen; je kunt het lezen in Sect. 2.4. Ik geef hier alleen voorbeeld.

Laten we aannemen dat we al over herschrijfregels beschikken waarmee we kunnen rekenen.[2] Nu willen we een functiesymbool toevoegen, dat een functie en een object als argument neemt, en de functie twee keer op dat argument uitvoert. We voegen hiervoor de volgende herschrijfregel toe:

$$\mathsf{twee\text{-}keer}(\lambda y.z(y), x) \to z(z(y))$$

We kunnen nu de volgende reductie uitvoeren:

$$\begin{aligned}\mathsf{twee\text{-}keer}(\lambda y.(2+y), 4) &\to 2 + (2 + 4)\\ &\to 2 + 6\\ &\to 8\end{aligned}$$

In de eerste stap wordt x door a vervangen, en $\lambda y.z(y)$ door de functie die een argument y, neemt en de term $2 + y$ oplevert. Je ziet dat het resultaat is dat er twee keer 2 bij 4 wordt opgeteld. Je ziet ook dat, na de eerste stap, één voorkomen van de +'en als argument van de andere voorkomt. Dit wordt *nesten* genoemd, en is er de voornaamste reden van dat het veel lastiger is om dingen over hogere-orde herschrijven te bewijzen, dan over eerste-orde herschrijven.

Equivalentie van reducties

Zoal ik hierboven al zei, kun je vaak kiezen in welke volgorde je de deelproblemen van een som oplost, of, in herschrijf-terminologie, in welke volgorde je de *redexen* van een term *contraheert*. Meestal is die precieze volgorde echter helemaal niet van belang. In het voorbeeld op de eerste pagina van deze samenvatting, konden we bijvoorbeeld kiezen om eerst $1 + 3$ uit te rekenen, of om eerst $4 + 1$ uit te rekenen. Beide keuzes hebben verschillende reducties tot gevolg, maar die reducties komen eigenlijk op het zelfde neer, dat wil zeggen, ze zijn *equivalent*.

Equivalente reducties lijken soms niet op elkaar. Vaak zijn ze zelfs niet eens even lang. Laten we aannemen dat we een herschrijfsysteem hebben dat kan optellen en vermenigvuldigen en dat ook de regel bevat:

$$\mathsf{kwadraat}(x) \to x \times x$$

De volgende twee reducties zijn intuïtief equivalent, omdat ze 'hetzelfde werk doen', maar hun lengte verschilt:

$$\begin{aligned}\mathsf{kwadraat}(2+2) &\to (2+2) \times (2+2)\\ &\to 4 \times (2+2)\\ &\to 4 \times 4\\ &\to 16\end{aligned}$$

$$\begin{aligned}\mathsf{kwadraat}(2+2) &\to \mathsf{kwadraat}(4)\\ &\to 4 \times 4\\ &\to 16\end{aligned}$$

[2] Er bestaat een herschrijfsysteem waarmee kan worden opgeteld en vermenigvuldigd. In deze samenvatting ga ik er verder niet op in hoe dit herschrijfsysteem er uit ziet. In werkelijkheid zijn de rekenstappen geen enkelvoudige stappen, maar rijtjes van stappen.

In verschillende situaties is het handig om een goede notie van equivalentie van reducties te hebben. Vaak willen we bijvoorbeeld een berekening zo snel mogelijk uitvoeren, maar dan willen we er wel zeker van zijn dat die snelle berekening die we uiteindelijk vinden, equivalent is aan andere berekeningen, die weliswaar minder efficiënt zijn, maar in ieder geval correct. Ook is een notie van equivalentie handig als we willen bewijzen dat alle mogelijke reducties een bepaalde eigenschap hebben: voor veel eigenschappen geldt dat als je hem bewezen hebt voor één reductie, hij dan automatisch ook voor alle equivalente reducties geldt.

Voordat we het begrip equivalentie echter in formele bewijzen kunnen gebruiken, moeten we hem eerst op een formele manier beschrijven. We kunnen vanuit verschillende gezichtspunten naar de notie equivalentie van reducties kijken, en verschillende gezichtspunten leiden tot heel verschillende formele definities die nuttig zijn in verschillende situaties. In dit proefschrift heb ik vanuit drie gezichtspunten naar equivalentie van reducties gekeken, en drie verschillende formele definities van deze notie gegeven. Het hoofdresultaat (Stelling 7.2.1) is uiteindelijk dat al deze drie definities voor een belangrijke klasse van hogere-orde herschrijven op het zelfde neerkomen, dat wil zeggen dat twee reducties permutatie-equivalent zijn dan en slechts dan als ze standaardisatie-equivalent zijn, en dan en slechts dan als ze projectie-equivalent zijn.

Samenvatting van de hoofdstukken

Hoofdstuk 2: Herschrijven en bewijstermen

In dit hoofdstuk worden de verschillende noties van herschrijven die in dit proefschrift worden gebruikt geïntroduceerd, in het bijzonder hogere-orde termherschrijven. Bovendien definieer ik zogenaamde *bewijstermen*. Een bewijsterm is een term waarvan de betekenis een herschrijfstap is. Door herschrijfstappen als termen op te vatten, kan ik in de rest van het proefschrift technieken en bewijsmethodes uit de herschrijftheorie gebruiken, om reducties te beschrijven en er dingen over te bewijzen.

Hoofdstuk 3: Permutatie-equivalentie

In dit hoofdstuk definieer ik de eerste vorm van equivalentie van reducties, permutatie-equivalentie: twee reducties zijn equivalent, als we de ene reductie in de andere kunnen omvormen door stappen om te wisselen. Eerst definieer ik permutatie-equivalentie van *eindige* reducties met behulp van een stelsel van vergelijkingen op bewijstermen. In de tweede helft van het hoofdstuk kijk ik hoe deze notie van equivalentie kan worden uitgebreid naar *oneindige* reducties.

Hoofdstuk 4: Eindige familieontwikkelingen

In dit hoofdstuk bewijs ik een belangrijke eigenschap van hogere-orde herschrijfsystemen, namelijk de eigenschap dat alle zogenaamde familie-ontwikkelingen eindig zijn. Deze eigenschap heeft niet direct iets met equivalentie van reducties te maken, maar ik heb haar in het volgende hoofdstuk nodig om te bewijzen dat de definitie van standaardisatie-equivalentie die ik daar geef correct is.

In elke herschrijfstap worden de symbolen die in de linkerkant van de toegepaste regel voorkomen weggehaald, en vervangen door de symbolen van de rechterkant van de regel. Die nieuwe symbolen behoren als het ware tot de volgende generatie. Een vorm van herschrijven bezit de eindige-familie-ontwikkelingeneigenschap, als het zo is dat er in geen enkele oneindige reductie een bovengrens op de generatie van de erin voorkomende functiesymbolen is. Dat eerste-orde herschrijven deze eigenschap bezit is al langer bekend. Het bewijs dat hogere-orde herschrijfsystemen de eigenschap ook hebben, is echter niet triviaal.

De eindige-familieontwikkelingeneigenschap komt goed van pas als we willen bewijzen dat een bepaald herschrijfsysteem geen oneindige reducties toelaat: soms is het namelijk makkelijker te bewijzen dat er wel een bovengrens op de generaties van de functiesymbolen bestaat, dan direct te laten zien dat alle reducties eindig zijn.

In het laatste deel van het hoofdstuk pas ik de ontwikkelde theorie toe om te bewijzen dat de eindige-ontwikkelingeneigenschap geldt voor hogere-orde herschrijven, en dat elke reductie in de simpel getypeerde λ-calculus eindig is.

Hoofdstuk 5: Standaardisatie

In dit hoofdstuk definieer ik de tweede vorm van equivalentie van reducties, standaardisatie-equivalentie.

Eerst definieer ik wat een standaard-reductie is. Dit is een reductie waarin de stappen in een vantevoren vastgestelde volgorde worden gedaan. Je kunt dit vergelijken met een gesorteerde lijst: dat is een lijst waarin de elementen zich in een vantevoren vastgestelde volgorde, namelijk van klein naar groot, bevinden. Op dezelfde manier worden de stappen in een standaard-reductie uitgevoerd 'van buiten naar binnen', ofwel tekstueel van links naar rechts.

Ik beschrijf twee manieren om een eindige reductie in een standaard-reductie om te zetten. Bij selectiestandaardisatie, wordt een reductie naar standaardvorm omgezet, door in de reductie de hele tijd die stap te kiezen die de buitenste redex contraheert. De reductie die precies uit de gekozen stappen bestaat, is de standaardreductie. Bij inversiestandaardisatie, worden steeds twee stappen die in de verkeerde volgorde staan omgewisseld. Als er geen stappen meer in de verkeerde volgorde staan, hebben we de standaardreductie bereikt. Ik bewijs dat de twee manieren van standaardisatie hetzelfde resultaat opleveren.

Tenslotte definieer ik dat twee reducties equivalent zijn, wanneer ze dezelfde

standaardvorm hebben, en beschrijf een aantal problemen die de kop opsteken bij zogenaamde niet-lokale herschrijfsystemen.

Aan het einde van het hoofdstuk probeer ik ook een vorm van standaardisatie voor oneindige reducties te ontwikkelen. Hoewel dit ten dele lukt, blijkt deze methode niet geschikt om standaardisatie-equivalentie voor oneindige reducties te definiëren.

Hoofdstuk 6: Residuen

In dit hoofdstuk wordt de laatste vorm van equivalentie van reducties gedefinieerd, projectie-equivalentie.

Eerst definieer ik een vorm van "aftrekken" voor reducties, wat in dit hoofdstuk *projectie* wordt genoemd. Gegeven een reductie $\mathcal{R}$ en een reductie $\mathcal{S}$, wordt het zogenaamde *residu* van $\mathcal{R}$ na $\mathcal{S}$ gevormd door uit $\mathcal{R}$ precies die stappen te verwijderen die ook in $\mathcal{S}$ voorkomen.

Met deze notie, definieer ik eerst een *orde* op reducties: een reductie $\mathcal{R}$ is kleiner dan of gelijk aan een reductie $\mathcal{S}$, als het zo is dat het residu van $\mathcal{R}$ na $\mathcal{S}$ de lege reductie is (dat wil zeggen, de reductie waarin geen enkele reductiestap wordt gedaan). We definiëren vervolgens projectie-equivalentie van reducties met behulp van deze orde: twee reducties zijn equivalent als de ene kleiner dan of gelijk aan de andere is en omgekeerd.

Hoofdresultaat

Het hoofdresultaat van dit proefschrift is het volgende:

> Voor eindige, orthogonale reducties (in lokale hogere-orde herschrijfsystemen) komen permutatie-equivalentie, standaardisatie-equivalentie en projectie-equivalentie overeen.