Confluence for Abstract and Higher-Order Rewriting

VRIJE UNIVERSITEIT

Confluence for Abstract and Higher-Order Rewriting

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor aan
de Vrije Universiteit te Amsterdam,
op gezag van de rector magnificus
prof.dr E. Boeker,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de faculteit der wiskunde en informatica
op dinsdag 29 maart 1994 te 15.30 uur
in het hoofdgebouw van de universiteit, De Boelelaan 1105

door

VINCENT VAN OOSTROM

geboren te Barendrecht

Promotor: prof.dr J.W. Klop
Referent: dr J.R. Hindley

cover design René Tosari

Acknowledgements

The person I am indebted to most, is my supervisor Jan Willem Klop who let me step through the scientific looking glass into the garden of rewriting. He guided me to the interesting places of that garden and gave me the freedom to wander about as I pleased. I thank Aart Middeldorp for showing me how to walk, both literally and figuratively, in the first year of my appointment. His interest in my work and his pertinent questions have always been stimulating. Wandering and wondering is nice, but exploring is nicer. Femke van Raamsdonk explored several routes (and dead ends) into the realm of higher-order rewriting together with me. Working with her has always been joyful and never monotonous. This thesis might have been written without her support, but I am not sure.

I thank Roel de Vrijer for always letting me drop in to discuss my work, and for letting me take 'drop' out of his room. 'Extremely long distance walker' Fer-Jan de Vries took the trouble of reading previous versions of the thesis while standing on his head. His mumbling and grumbling have been a great incentive to finish my thesis (just) in time, for which my thanks. I thank Erik de Vink and Zurab Khasidashvili for being my co-authors. The regular participants of the TeReSe-meetings (apart from the people already mentioned): Hans Zantema, Jaco van de Pol, Steffen van Bakel, Maria Ferreira and Thomas Arts, I thank for their interest in my work. The talks I attended at both these and the λ-intercity seminar meetings were always interesting.

I am very grateful to Roger Hindley who was willing to be the referee for this thesis. I thank him for many helpful comments on a draft version and for his presence in the promotion committee. Also I am very grateful to the other members of the committee, in particular Henk Barendregt, Aart Middeldorp, Rob Nederpelt, Roel de Vrijer and Hans Zantema.

While wandering about, one meets a lot of people. I took the first steps in theoretical computer science at the Free University together with Franck van Breugel. Apart from being the ideal roommate, he also endured the food of the university restaurant together with me. This arduous task was taken over by Erik Hamoen, who never ceased to amaze me by his way of doing things, and Piero d' Altan, who knows more about Dutch than most Dutch people do.

My roommates, Gerard Vreeswijk and Marcello Bonsangue, I thank for supplying me with their own kind of humour. Thanks to the other members of the theory group: Jaco de Bakker, John-Jules Meijer, Wiebe van der Hoek, Bernd van Linder and Frank de Boer, for pleasant discussions on a broad range of subjects.

On a personal level, I am most grateful to my parents and my brother for their support. They always kept me in touch with the other side of the looking glass. In this respect, I am also grateful to my soccer teammates both from B.V.V. Barendrecht 5 and D.V.V.A. 8. I thank them for the necessary distraction.

Contents

Introduction **i**

1 Preliminaries **1**

1.1 Sets . . . 1
1.2 Relations . . . 1
1.3 Natural Numbers . . . 4
1.4 Multisets . . . 4
1.4.1 Multiset Extensions . . . 7

2 Abstract Rewriting **15**

2.1 Abstract Rewriting Systems . . . 16
2.2 Properties of Rewriting Systems . . . 20
2.3 Confluence by Decreasing Diagrams . . . 25
2.3.1 Many Decreasing Diagrams . . . 28
2.3.2 DCR implies CR . . . 32
2.3.3 More Decreasing Diagrams . . . 39
2.3.4 Is DCR equivalent to CR? . . . 42
2.4 Confluence by Orthogonality . . . 44
2.5 Confluence Modulo . . . 54

3 Higher-Order Rewriting **59**

3.1 Higher-Order Rewriting Systems . . . 60
3.1.1 Associating an RRS to a HORS . . . 72
3.1.2 Orthogonal HORSs . . . 76
3.1.3 Weakly Orthogonal HORSs . . . 87
3.1.4 Combinations of HORSs . . . 88
3.2 Simply Typed Lambda Calculus . . . 90
3.3 Term Rewriting Systems . . . 105
3.3.1 From TRS to HORS . . . 106
3.3.2 Orthogonal TRSs . . . 111
3.3.3 Weakly Orthogonal TRSs . . . 112
3.3.4 Combinations of TRSs . . . 113
3.4 Klop's Combinatory Reduction Systems . . . 115
3.4.1 From CRS to HORS . . . 118
3.4.2 Orthogonal CRSs . . . 121
3.4.3 Weakly Orthogonal CRSs . . . 125
3.4.4 Combinations of CRSs . . . 126
3.5 Nipkow's Higher-order Rewrite Systems . . . 128
3.5.1 From HRS to HORS . . . 129
3.5.2 Orthogonal HRSs . . . 130
3.5.3 Weakly Orthogonal HRSs . . . 132
3.5.4 Combinations of HRSs . . . 134

3.5.5 From HRSs to CRSs 134
3.5.6 A list-extension of lambda calculus 135

Bibliography **137**

Samenvatting **147**

Introduction

Rewriting Rewriting is a method of computation. Computation is performed by successively performing elementary computation steps, transforming objects into other objects. It depends on the person/machine what these elementary computation steps are. For example, some people are able to compute the result, 27, of the expression 3^3 in one step, using the rule $27 \to 3$. Others apply more elementary rules and compute the answer in several steps, e.g. $3^3 \to 3^2 \times 3 \to (3 \times 3) \times 3 \to 9 \times 3 \to 27$, using the rules $n^3 \to n^2 \times n$, for $n = 3$, then $n^2 \to n \times n$, for $n = 3$, then $3 \times 3 \to 9$, and finally $9 \times 3 \to 27$. The first two rules employ the variable n, where n stands for an arbitrary (but fixed during a computation step) number. In both cases, the essence is that there is a collection of *rewrite rules* specifying how one object may be rewritten to another one. Such a collection is called a *rewriting system*.

Rewriting is not restricted to computing with numbers. As soon as there is a process transforming something in something else, one can try to extract the transformation rules, thus obtaining a rewriting system. The purpose of viewing a coherent set of transformations as a rewriting system is, of course, to derive properties of the transformation process from properties of the rewriting system. In the case of expressions such as 3^3, everyone expects that successively applying the rules taught at secondary school will eventually produce a result (a *normal form*) and that this result is unique. The first property is called the *termination* property and the second one is called the *uniqueness of normal forms* property. These are the two main properties studied in rewriting. In this thesis we will be concerned with the *confluence* property; a condition on rewriting systems ensuring uniqueness of normal forms.

Two methods for proving confluence are considered. One applies to *abstract* rewriting systems and the other one to the more restricted class of (higher-order) *term* rewriting systems. Before explaining these methods, we first introduce these rewriting systems.

Abstract and Term Rewriting In the example above, both computations rewrite the object 3^3 to the object 27.

The first computation uses the rewrite rule $3^3 \to 27$. This can be viewed as a step of *abstract rewriting*; the structure of the objects is not important at all, one just knows that 3^3 can be rewritten to 27.

The second computation starts with the step $3^3 \to 3^2 \times 3$, using the rule $n^3 \to n^2 \times n$ for $n = 3$. Actually, this step should be viewed as a three phase process.

1. In order to see that the rule $n^3 \to n^2 \times n$ can be applied to 3^3, its left-hand side n^3 must be *matched* with the expression 3^3, that is, 3^3 is decomposed as the expression n^3 *where* $n = 3$.

2. Because the left-hand side n^3 of the rule $n^3 \to n^2 \times n$ is present in the expression n^3 *where* $n = 3$, one can *replace* the left-hand side by the

right-hand side $n^2 \times n$ of the rule, resulting in the expression $n^2 \times n$ *where* $n = 3$.

3. The expression $n^2 \times n$ *where* $n = 3$ can be composed, by *substituting* 3 for n in $n^2 \times n$, resulting in $3^2 \times 3$.

So, a computation step consists of a matching, a replacement and a substitution phase. In this process, complex objects are decomposed into simpler ones in the matching phase and complex objects are composed of simple ones in the substitution phase. The composed (complex) objects are called *terms*. Therefore, computation steps making use of rules such as $n^3 \to n^2 \times n$ are called *term rewriting steps* and the rewriting system is called a *term rewriting system*.

To every term rewriting system, an abstract rewriting system can be associated by 'forgetting' the structure of terms, but *labelling* rewrite steps to distinguish applications of different rewrite rules. An abstract rewriting system endowed with such a labelling we call a *labelled rewriting system*. Let us give an example of how such a labelled rewriting system can be naturally obtained from a term rewriting system. Consider the term rewriting rules $n \times 0 \to 0$, $1 \times n \to n$, and the term rewriting step from 1×0 to 0, which can be obtained by applying either of the rules. These applications can be discriminated by labelling instances of the first rule by zero, giving $1 \times 0 \to_{\mathsf{zero}} 0$ and instances of the second rule by one, giving $1 \times 0 \to_{\mathsf{one}} 0$. In this way, one obtains a labelled rewriting system from the original term rewriting system.

Having introduced the rewriting systems of interest, we proceed by explaining the property we study for them.

Confluence As its etymology suggests, confluence has to do with the flowing together of streams. One can call two streams having a common source confluent if for every two water molecules anywhere in them there is a possible point of confluence (note that water usually doesn't flow upward). Viewing computations as streams, this corresponds to the property of rewriting systems that for every two computations starting from some expression, a so-called *divergence*, it is possible to continue the computations to a point where they result in the same expression, a so-called *convergence*. Confluence not only ensures that normal forms are unique, it also guarantees that random computations are correct in the sense that it is always possible to continue a random computation to reach the normal form (if it exists).

How can one prove a rewriting system to be confluent? This can be done by studying how the computation rules interact with each other. How two computation rules interact can be studied by looking at situations in which they can both be performed. In the above situation, both rules zero and one can be applied to the term 1×0. Performing one of them erases the possibility of doing the other, but fortunately they produce the same expression. Other ways for these rules to interact are also possible. For example, zero can also interact with one in a non-erasing way, like:

$$\begin{tikzcd}
& 1 \times (2 \times 0) \arrow[dl, "\mathsf{one}"'] \arrow[dr, "\mathsf{zero}"] & \\
2 \times 0 \arrow[dr, "\mathsf{zero}"'] & & 1 \times 0 \arrow[dl, "\mathsf{one}"] \\
& 0 &
\end{tikzcd}$$

Doing either of them, does not destroy the possibility of doing the other. Because the two streams $1 \times (2 \times 0) \to_{\mathsf{one}} 2 \times 0$ and $1 \times (2 \times 0) \to_{\mathsf{zero}} 1 \times 0$ along the peak $\swarrow\searrow$ flow together via $2 \times 0 \to_{\mathsf{zero}} 0$ and $1 \times 0 \to_{\mathsf{one}} 0$ in the valley $\searrow\swarrow$, such a diagram is called a *confluence diagram.* In this case we also speak of a *local* peak because both sides of the peak consist of single computation steps, so the diagram shows *local confluence* in this particular case of interaction between zero and one.

One could hope that it is sufficient for proving confluence that every local peak can be completed (by adjoining an appropriate valley) to form a confluence diagram. That this condition is not sufficient in general is shown by the abstract rewriting system $a \leftarrow b \leftrightarrow c \rightarrow d$ due to Hindley (obtained by condensating an infinite example in Schroer's thesis, due to Rosser). Trying to complete the peak:

$$\begin{tikzcd}
c \arrow[r] \arrow[d] & b \arrow[r] & a \\
d & &
\end{tikzcd}$$

by successively completing local peaks to confluence diagrams, one gets nowhere:

$$\begin{tikzcd}
c \arrow[r] \arrow[dd] & b \arrow[rr] \arrow[d] & & a \arrow[d, equal] \\
& c \arrow[r] \arrow[d] & b \arrow[r] & a \\
d \arrow[r, equal] & d & &
\end{tikzcd}$$

We end up in the same situation we started with! (For ease of drawing we have rotated the diagrams $\pi/4$ such that all arrows are from top to bottom and from left to right.) So, although for every local peak a valley can be found, the system is not confluent.

The usual solution to this problem is to put extra conditions on the 'form' of valleys completing local peaks to confluence diagrams. In this thesis we consider two such conditions, one for abstract rewriting systems and one for term rewriting systems.

Confluence for Abstract Rewriting The first (and most important) sufficient condition for deducing confluence from local confluence was found by Newman as early as 1942. He showed that for terminating rewriting systems, proving confluence can be reduced to proving local confluence. This is still the main tool for proving confluence.

Another way to prove confluence from local confluence is expressed by the Lemma of Hindley-Rosen. It states that if the rules in a system have no 'interaction' at all, i.e. doing a step according to one rule, does not affect the possibility of doing a step according to another one, then the system is confluent. The local confluence diagrams look like:

$$\begin{tikzcd} a \arrow[r, "j"'] \arrow[d, "i"] & c \arrow[d, "i"] \\ b \arrow[r, "j"'] & d \end{tikzcd}$$

(as in the non-erasing interaction between zero and one). Hindley's counterexample violates this condition, because to complete the local peak $d \leftarrow c \rightarrow b$ at least two steps are required starting from b (see the previous picture).

In Chapter 2 we introduce a method called 'confluence by decreasing diagrams' which can be viewed as a combination of Newman's Lemma and the Lemma of Hindley-Rosen, allowing local confluence diagrams such as:

$$\begin{tikzcd} a \arrow[rr] \arrow[d] & & c \arrow[d] \\ b \arrow[r] & a \arrow[r] & d \end{tikzcd}$$

which cannot be dealt with by either of the methods above. The power of this method is illustrated in Chapter 2 by many examples of applications. Furthermore, we show that the method is maybe a bit too powerful: every countable abstract rewriting system which is confluent can be proved to be so by the method. Nevertheless, we think the method is valuable, because it can be used quite straightforwardly to prove many (known) abstract 'confluence by local confluence' results.

Confluence for Term Rewriting The interaction between the rewrite rules of a term rewriting systems can be studied by looking at which *parts* of a term the rules can operate on. Roughly speaking, if two rules can never operate on the same part of a term, then the rewriting system is called orthogonal. This gives rise to a well-known method called 'confluence by orthogonality'. This method is due to Church and Rosser, who employed it in 1936 to show that λI-calculus is confluent. For this reason, confluence is also known as the *Church-Rosser property*.

How can one formalise that rewrite rules 'operate on different parts' of a term? As explained above a term rewriting step consists of a matching, a replacement and a substitution phase. In the first phase a term is decomposed into some parts, then some part is replaced by another one, and finally the parts are composed again. Consider the term $1 \times (1 \times 3)$. This term can be decomposed as $1 \times n$ *where* $n = (1 \times m$ *where* $m = 3)$. The rule one can be applied to the two *distinct* parts $1 \times n$ and $1 \times m$, giving rise to steps ending in (1×3) and $1 \times (3)$ respectively. These two steps are said to be *simultaneous*. Because every pair of one-steps is simultaneous, the rule is called *orthogonal* (to itself).

As an example of a pair of non-orthogonal rewrite steps, consider the term 1×0 and the rules zero and one. Both rules can be applied to the term 1×0, but 1×0 cannot be decomposed into parts such that both rules can be applied to distinct parts, as one easily verifies. The steps are said to be *overlapping* and the rules are called non-orthogonal with respect to each other.

In order to study orthogonality, some further analysis of the substitution phase of a rewrite step is required. In term rewriting systems it makes sense to *trace* parts of a term along a rewrite step. For example, in the rewrite step $1 \times 4 \rightarrow_{\mathsf{one}} 4$, the part 4 of the term 1×4 can be traced in an obvious way to the 4 of the result. For this reason, the latter is called a *descendant* of the former along the rewrite step. Three things can happen to a part of a term along a rewrite step.

1. In the rewrite step $3 \times 0 \rightarrow_{\mathsf{zero}} 0$, the part 3 in 3×0 does not have descendants in 0, it is *erased*.

2. In the rewrite step $2 \times 1 \rightarrow_{\mathsf{two}} 1 + 1$, obtained by an application of the rule $\mathsf{two} : 2 \times n \rightarrow n + n$, the part 1 in 2×1 has two descendants in $1+1$, it is *duplicated*.

3. In the rewrite step $1 \times 4 \rightarrow_{\mathsf{one}} 4$, the part 4 in 1×4 has precisely one descendant in 4.

If each part has precisely one descendant along some rewrite step, the step is called *linear*.

Having introduced this terminology, we can continue our study of orthogonality. Orthogonality of a rule would immediately imply confluence if all the rules were linear, because applying one rule always produces one descendant or *residual* of the part to be replaced by the other one. By applying this other rule to the residual in both cases, this produces a local confluence diagram as in the Lemma of Hindley-Rosen. For example:

$$\begin{tikzcd} 1 \times (1 \times 3) \arrow[r, "\mathsf{one}"'] \arrow[d, "\mathsf{one}"] & (1 \times 3) \arrow[d, "\mathsf{one}"] \\ 1 \times (3) \arrow[r, "\mathsf{one}"'] & (3) \end{tikzcd}$$

So, in order to prove confluence, local confluence is proved by rewriting the residual left by the application of the other rule. A sequence of rewrite steps in which only residuals of parts present in the original term are rewritten, is called a *development* of those parts. The Finite Developments theorem (FD) expresses that all developments are finite and end in the same term in an orthogonal system. From FD confluence can be obtained easily.

How can one prove FD? The difficult part in proving FD, is proving that all developments are finite. In the case of linear rewrite steps this is not difficult, because the number of residuals in a development decreases by doing a rewrite step. Unfortunately, in a rewrite step such as $2 \times (1 \times 3) \rightarrow_{\mathsf{two}} (1 \times 3) + (1 \times 3)$, the part (1×3) has two residuals. In general, applying a rule may duplicate some part many times, increasing the number of *residuals* of that part. In the case of the rules above, developments are indeed finite, but this is not always easy to show.

Abstracting from the particular case of term rewriting, we present an abstract notion of orthogonality in Chapter 2 and give a confluence theorem for such *residual rewriting systems*. The abstract notion of orthogonality consists of the following three ingredients. First, distinct rules act on distinct parts ('consistency'). Second, rules may interact (i.e. duplicate each other) as long as this interaction is finitary ('finite developments' (FD)). Finally, the order in which distinct rules are applied does not influence the effect on other parts ('parametricity'). In other words, no matter in which order the rules which can be applied to some term are in fact applied the effect on the other parts is always the same.

In Chapter 3 we show how, to term rewriting systems which are orthogonal in the usual sense, an abstract rewriting system which is orthogonal in the sense of Chapter 2 can be associated. FD is known to hold for various classes of orthogonal term rewriting systems, because, roughly speaking, rewriting a residual can only duplicate parts which are 'below' it (in the usual tree representation of a term). In the case of first-order term rewriting systems, known as TRSs, termination is easy to show for that reason. The situation changes dramatically for so-called *higher-order* term rewriting systems, such as lambda calculus, Klop's CRSs, and Nipkow's HRSs. Proving FD for these systems is non-trivial, because rewrite rules may express complex substitution behaviour. In Chapter 3 we present a class of rewriting systems, even more general than CRSs and HRSs, which we call *Higher-Order Rewriting Systems*

(HORSs) and present a number of sufficient conditions on these allowing to reduce proving FD to proving termination of the *substitution calculus* of the HORS (which is simply typed λ-calculus in the case of CRSs and HRSs).

Higher-Order Rewriting Systems What are higher-order rewriting systems? Although there is, as far as we know, no universally accepted definition of higher-order rewriting systems, for the moment one can think of them as the rewriting systems obtained when transformations of mathematical functions are modelled in a term rewriting system. The standard encoding of a mathematical function $f : x \mapsto f(x)$ (f is the function which maps x to $f(x)$) as a term, is by writing it as $\lambda x.f(x)$, this is Church's λ-notation. The defining property of functions is that they can be applied to an argument in order to yield a result. The notation for the application of the function $\lambda x.f(x)$ to some argument a is simply $(\lambda x.f(x))(a)$. This is just notation, the function is not evaluated yet. Evaluation is specified by the rule $(\lambda x.f(x))(a) \to f(a)$, this rule is called the β-rule of λ-calculus.

The examples from mathematics usually employed to illustrate higher-order rewriting, are rewriting systems for integration and differentiation. For example, the product rule for differentiation is $d(\lambda x.f(x)\cdot g(x)) \to \lambda y.(d(\lambda x.f(x)))(y)\cdot g(y)+f(y)\cdot(d(\lambda x.g(x)))(y)$. An example of a symbolic computation using this rule and some auxiliary rules is:

$$
\begin{array}{lll}
d(\lambda x.x\cdot 3) & \to & \lambda y.(d(\lambda x.x))(y)\cdot 3 + y\cdot(d(\lambda x.3))(y) \\
 & \to_{d(\lambda x.x)\to\lambda x.1} & \lambda y.(\lambda x.1)(y)\cdot 3 + y\cdot(d(\lambda x.3))(y) \\
 & \to_{\beta} & \lambda y.1\cdot 3 + y\cdot(d(\lambda x.3))(y) \\
 & \to_{1\cdot m\to m} & \lambda y.3 + y\cdot(d(\lambda x.3))(y) \\
 & \to_{d(\lambda x.m)\to\lambda x.0} & \lambda y.3 + y\cdot(\lambda x.0)(y) \\
 & \to_{\beta} & \lambda y.3 + y\cdot 0 \\
 & \to_{m\cdot 0\to 0} & \lambda y.3 + 0 \\
 & \to_{m+0\to m} & \lambda y.3
\end{array}
$$

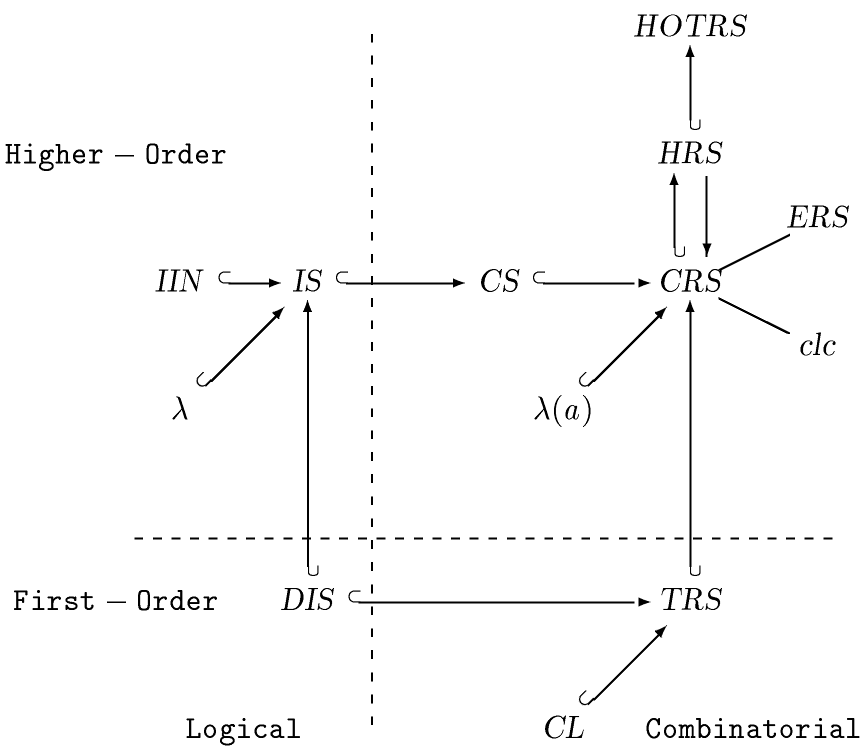
Note that evaluation of a function must be performed explicitly using the β-rule, whereas in common mathematical practice no distinction is made (in notation) between the application of a function to its argument and the result of such an application. Mathematicians work modulo the β-rule, or more precisely they work with the β-normal forms of terms as representatives of β-equivalence classes. Moreover, they usually live in a typed world, disallowing such oddities as $\lambda x.x(x)$, the function of x having as result the application of x to itself.

In Chapter 3 we try to capture this situation by introducing *Higher-Order Rewriting Systems*. These are rewriting systems equipped with a *substitution calculus*. Rewriting is performed modulo this calculus, so the hypothetical mathematicians above can be modelled by taking the (typed) β-rule as rule of the substitution calculus.

Our main objective in Chapter 3 is to give sufficient (syntactic) conditions on rewrite rules, on the substitution calculus and on the interaction between them to ensure orthogonality (in the sense of Chapter 2). Among these conditions are termination and confluence of the substitution calculus. Both these conditions are essential in our proof of Finite Developments for HORSs. Actually, the orthogonality constraint can be relaxed somewhat to *weak orthogonality*, without losing confluence. Weak orthogonality does not require rules to be consistent, but instead in case of an inconsistency between two rules they are required to produce the same result. Finally, we show that confluence is preserved when combining two systems which are weakly orthogonal with respect to each other.

We conclude by showing that TRSs, CRSs and HRSs which are orthogonal in the usual sense, i.e. having left-linear and non-ambiguous rules, satisfy these conditions and hence are also orthogonal in our sense.

A Rewriting Road Map In the picture below, based on [OR93], we show relationships between the historically most important classes of term rewriting systems. We have classified them along two dimensions. Horizontally, we distinguish between *logical* and *combinatorial* systems. The logical systems are the ones for which the Curry-Howard isomorphism still makes sense, i.e. the left-hand sides of rules satisfy a constructor-destructor discipline. If the left-hand sides consist of possibly complex combinations of symbols, then we call the system combinatorial. Vertically, we distinguish between *first-order* and *higher-order* in the usual way.



In the diagram, an arrow $\hookrightarrow$ between two systems means direct embedding (preserving computation steps), a line – means related and the arrow $\rightarrow$ denotes an encoding (preserving computations).

We will not discuss these systems extensively here, but give references to the literature instead. It turns out that *all* these systems can be viewed as HORSs having simply typed λ-calculus as substitution calculus. For an overview of rewriting systems until 1980 see also [Klo80, pp. 132,133]. For a nice account of the history of lambda calculus, see the article by Rosser [Ros82] and the introduction of the book by Barendregt [Bar84].

In historical order we have:

- *CL* = Combinatory Logic, introduced by Curry in 1930 [Cur30a, Cur30b], but already present in the work of Schönfinkel [Sch24].
- λ-calculus, introduced by Church in [Chu32].
- *TRS* = Term Rewriting System. I don't know who introduced this name. Cf. Rosen [Ros73].
- *CS* = Contraction Scheme. Introduced by Aczel in 1978 [Acz78].
- λ(a)-reductions. Introduced by Hindley [Hin78a].
- *CRS* = Combinatory Reduction System. Introduced by Klop in his PhD thesis [Klo80].
- *HOTRS* = Higher-Order Term Rewriting System. Introduced by Wolfram in his PhD thesis (see his [Wol93]).
- *ERS* = Expression Reduction System. Introduced by Khasidashvili [Kha90].
- *IIN* = Intuitionistic Interaction Net. Introduced by Lafont [Laf90].
- *HRS* = Higher-order Rewrite System. Introduced by Nipkow [Nip91].
- *(D)IS* = (Discrete) Interaction System. Introduced by Asperti and Laneve [AL92].
- *clc* = conditional λ-calculi. Introduced by Takahashi [Tak93].

Main Results We conclude this introduction with a summary of what we consider to be the main results of this thesis.

1. The method of confluence by decreasing diagrams.
2. The introduction of the concept HORS. This concept allows for a uniform presentation of notions and results for various paradigms of term rewriting (CL, lambda calculus, TRSs, CRSs, HRSs).
3. Some results for HORSs:

(a) The Finite Developments theorem for orthogonal HORSs. This is proved via a reduction to termination of the substitution calculus of the HORS.

(b) A confluence by weak orthogonality result for HORSs, which allows to derive confluence directly for systems such as lambda beta-eta calculus and PCF with 'parallel or'.

(c) Modularity of confluence for HORSs which are weakly orthogonal with respect to each other.

1 Preliminaries

In these preliminaries we introduce the notation we employ for standard notions. The only section containing original material is the one on multisets which is based on [Oos94].

1.1. Sets

We just introduce the notation we use for sets and operations on them.

DEFINITION 1.1.1. We use A, B, C and D to range over sets and a, b, c, d to denote elements of sets. The operations powerset, complement, intersection, union, product, difference are denoted by $\mathfrak{P}$, $\complement$, $\cap$, $\cup$, $\times$, $-$. Set membership and cardinality are denoted by $\in$ and $\#$. An element $(a,b) \in A \times B$ is called a pair. The left- and right-projections are denoted by π_1 and π_2.

1.2. Relations

In defining relations and the operations on them, we are somewhat more extensive than in the previous section, because the *abstract rewriting systems* studied in the next chapter are relations.

DEFINITION 1.2.1. Let A, B be sets. A *relation from A to B* is a triple $\langle A, B, C\rangle$ where $C \subseteq A \times B$. The relations from A to B are denoted by $\mathbf{Rel}(A, B)$. The variables R, S, T range over relations. Let $R =^{\mathrm{def}} \langle A, B, C\rangle$. We say that a is *R-related* to b, $a\ R\ b$, if $a \in A$ & $b \in B$ & $(a,b) \in C$. The components of R have the following names: $\mathsf{dom}(R) =^{\mathrm{def}} A$ is its *domain*, $\mathsf{cod}(R) =^{\mathrm{def}} B$ is its *codomain* and $\boldsymbol{R} =^{\mathrm{def}} C$ is its *graph*. We often confuse a relation and its graph. In the special case of a relation from A to itself, we speak of a relation *on* A. Such a relation is represented by a $\langle$domain, graph$\rangle$-tuple.

DEFINITION 1.2.2. (*operations on relations*) Let R be a relation from A to B. Let S be a relation from C to D. Let $a \in A$, $b \in B$, $c \in C$, $d \in D$.

1. The relation R is a *sub-relation* of S, if $A \subseteq C$, $B \subseteq D$ and $\boldsymbol{R} \subseteq \boldsymbol{S}$. If moreover, $A = C$ and $B = D$, then R is called a *restriction* of S and S is called an *extension* of R,

2. The *total* relation $\mathsf{tot}_{A,B}$ from A to B has $A \times B$ as graph,

3. The *identity* id_A on A is defined by $a\ \mathsf{id}_A\ a$,

4. The *inverse* (or *converse*) of R, R^{-1} from B to A is defined by $b\ R^{-1}\ a \iff a\ R\ b$,

5. The *complement* (or *negation*) of R, $\complement R$ from A to B has the complement of $\boldsymbol{R}$ as graph,

6. The *union* of R and S, $R \cup S$ is defined to be the relation $\langle A \cup C, B \cup D, \boldsymbol{R} \cup \boldsymbol{S} \rangle$, that is, we take the componentwise union,

7. The *difference* of R and S, $R - S$ is defined to be the relation $\langle A, B, \boldsymbol{R} - \boldsymbol{S} \rangle$,

8. The *intersection* of R and S, $R \cap S$ is the componentwise intersection,

9. If $B = C$, then the *sequential composition*, $R\,;S$ from A to D is defined by $a\ R\,;S\ d \iff \exists b \in B.a\ R\ b\ \&\ b\ S\ d$,

10. Let R be a relation on A. The *m-fold composition* of R, R^m on A is defined by

    (a) $R^0 =^{\mathrm{def}} \mathsf{id}_A$,
    (b) $R^{m+1} =^{\mathrm{def}} R\,;R^m$,

11. If R is a relation on A, then the *restriction* of R to B, $R\restriction B$ is the relation $\langle A \cap B, \boldsymbol{R} \cap B \times B \rangle$,

12. The *product* of R and S, $R \times S$ is the componentwise product,

13. The *lexicographic product* of R and S, $R \times_{lex} S$ from $A \times C$ to $B \times D$ is defined by $(a,c)\ R \times_{lex} S\ (b,d) \iff (a\ R\ b)$ or $(a = b\ \&\ c\ S\ d)$,

14. The set of *R-images* of a is defined by $\{a\ R\} =^{\mathrm{def}} \{b \in B.a\ R\ b\}$,

15. The set of *R-inverse images* of b is defined by $\{R\ b\} =^{\mathrm{def}} \{a \in A.a\ R\ b\}$.

DEFINITION 1.2.3. A relation R on A is

1. *reflexive*, if $\mathsf{id}_A \subseteq R$,
2. *irreflexive*, if $\mathsf{id}_A \subseteq \complement R$,
3. *total*, if $\mathsf{tot}_{A,A} = R \cup R^{-1}$,
4. *symmetric*, if $R = R^{-1}$,
5. *anti-symmetric*, if $R \cap R^{-1} \subseteq \mathsf{id}_A$,
6. *transitive*, if $R\,;R \subseteq R$,

A transitive relation R on A, also called *order*, is

1. a *quasi*-order, if it is reflexive,
2. a *strict* order, if it is irreflexive,
3. an *equivalence* relation, if it is a symmetric quasi-order,
4. a *partial* order, if it is an anti-symmetric quasi-order.

5. *well-founded*, if there is no infinite sequence $a_1 \ S \ a_2, a_2 \ S \ a_3, \ldots$, where $S =^{\text{def}} R - (R \cap R^{-1})$.

It is well-known that the (lexicographic) product of two well-founded orders is a well-founded order again.

DEFINITION 1.2.4. Let $\sim$ be an equivalence relation on A and $a, b \in A$. The *equivalence class of a modulo* $\sim$, $\{a\}_\sim$ is defined to be $\{a \sim\}$. The *quotient set of A modulo* $\sim$, $A/\!\sim$ is defined to be $\{\{a\}_\sim . a \in A\}$. Let R be a relation on A. The *quotient relation of A modulo* $\sim$, $R/\!\sim$ on $A/\!\sim$ is defined by $\{a\}_\sim \ R/\!\sim \ \{b\}_\sim \iff a \sim ; R ; \sim b$.

DEFINITION 1.2.5. An operation mapping relations to relations is *monotone* (*anti-monotone*) in some argument if extending that argument extends (restricts) the image.

DEFINITION 1.2.6. Let P be a property of relations. Let R be a relation on A. The *P-closure* of R is the least relation extending R which has property P. We now list some well-known relation closures.

1. The reflexive closure of R is equal to $R \cup \mathsf{id}_A$ and denoted by $R^=$,

2. The transitive closure of R is equal to $\bigcup m \in \mathbb{N}^+ . R^m$ and denoted by R^+.

3. The transitive, reflexive closure of R is equal to $\bigcup m \in \mathbb{N} . R^m$ and denoted by R^*.

4. The symmetric closure of R is equal to $R \cup R^{-1}$.

5. The equivalence closure of R is equal to the transitive, reflexive closure of the symmetric closure of R.

Taking the reflexive closure commutes with both taking the symmetric and the transitive closure.

NOTATION 1.2.7. We use $\succ$, $\sqsupset$ and $>$ to range over orders and we employ the following notations.

R	R^{-1}	$R^=$	$R^{=-1}$
$\succ$	$\prec$	$\succeq$	$\preceq$
$\sqsupset$	$\sqsubset$	$\sqsupseteq$	$\sqsubseteq$
$>$	$<$	$\leq$	$\geq$

1.3. Natural Numbers

The variables α, β, γ range over the ordinals $\mathcal{O}$. The ordinals are ordered by $\ni$, also denoted by $>$. The order $\geq$ is a well order, that is, a well-founded total anti-symmetric order. The symbol ω denotes the first limit ordinal. Ordinals smaller than ω are called *finite* ordinals or *natural numbers.* The set consisting of the natural numbers 0, 1, 2, ... is denoted by $\mathbb{N}$. We abbreviate $\mathbb{N} - \{0\}$ by $\mathbb{N}^+$. The variables m, n, k, i and j range over natural numbers. In the sequel we use the following extension of the natural numbers.

Definition 1.3.1. Let $\mathbb{N}_\infty$ be $\mathbb{N}$ extended with a new element ∞. The element ∞ is the top element of the natural ordering $>$ on $\mathbb{N}$ extended to $\mathbb{N}_\infty$. The operation *complement* ($\neg$) is defined by $\neg(m) =^{\text{def}} \infty$ and $\neg(\infty) =^{\text{def}} 0$. The operations *minimum* ($\wedge$), *maximum* ($\vee$), *addition* ($+$) and *cutoff-subtraction* or *minus* ($\dot{-}$) on $\mathbb{N}_\infty$ are defined as for $\mathbb{N}$, extended as specified in Table 1.1.

Definition 1.3.2. Let A be a set. A map $f : \alpha \to A$ for some ordinal α, is called an *A-sequence of length α*. A *string* is a sequence of finite length, i.e. a map $f : m \to A$, for some natural number m. The variables σ, τ, υ range over the set A^* of A-strings. The *empty* string ε is the (unique) string of length 0. Prefixing a string σ by $a \in A$ is denoted by $a\sigma$. Concatenating two strings σ, τ is denoted by $\sigma \,;\, \tau$. The *left division* $\sigma\backslash\tau$ of τ *by* σ is defined to be τ', if $\tau = \sigma \,;\, \tau'$, and undefined otherwise. The left division gives rise to a well-founded partial order on strings called the *prefix order* $\succeq$, defined by $\tau \succeq \sigma$ if $\sigma\backslash\tau$ is defined. If $\tau \succeq \sigma$ ($\tau \succ \sigma$), then σ is called a (*proper*) *prefix* of τ. Two incomparable strings are called *disjoint.*

1.4. Multisets

A *multiset* is a collection in which objects are allowed to occur more than once or even infinitely often.

Definition 1.4.1. Let A be a set. Then, a *multiset on* A is a function in $A \to \mathbb{N}_\infty$. The class of multisets (on A) is denoted by **Mst** (**Mst**(A)). We use M, N, X, Y and Z to range over multisets. Let $a \in A$. We say that *a has multiplicity $M(a)$ in M*. We define $a \in M =^{\text{def}} M(a) > 0$, and say that *$a$ is an element of M*. The *multiset inclusion* relation is defined by $M \subseteq N =^{\text{def}} \forall a \in A. N(a) \geqslant M(a)$. We say that M is a *submultiset* of N.

Remark 1.4.2. Our notion of multiset is more general than the usual notion of multiset, where only finite multiplicities are allowed (cf. [DJ90]), but less general than the one considered e.g. by Blizard [Bli93], who allows multiplicities of any cardinality. For our purposes it suffices to consider multiplicities up to ∞.

Two kinds of multisets are of special interest to us: *finite* multisets and *set* multisets. Our finite multisets correspond to what are usually called just multisets. We lift some standard results for those to our more general notion. Our set multisets will be used to interpret sets as multisets. The idea is that an element of a set can be used any finite number of times so should have infinite multiplicity when the set is represented as a multiset. Usually sets are interpreted as multisets where the elements of the set occur exactly once in the multiset. In the usual interpretation multiset sum corresponds to disjoint union, in our interpretation it corresponds to ordinary union.

DEFINITION 1.4.3. Let A be a set.

1. A *finite* multiset on A is a multiset M such that $\sum a \in A.M(a) < \infty$. The class of finite multisets (on A) is denoted by **FMst** (**FMst**(A)). We use F, G, C, D, E and H to range over finite multisets. A finite multiset has finitely many distinct elements which occur finitely often.

2. A *set* multiset on A is a multiset M such that $\forall a \in A.M(a) \in \{0,\infty\}$. The class of set multisets (on A) is denoted by **SMst** (**SMst**(A)). We use S, T, U, V and W to range over set multisets. The elements of set multisets occur infinitely often.

natural number	$*$	$\infty * m$	$m * \infty$	$\infty * \infty$	$\circledast$	multiset
minimum	$\wedge$	m	m	∞	$\cap$	intersection
maximum	$\vee$	∞	∞	∞	$\cup$	union
addition	$+$	∞	∞	∞	$\uplus$	sum
minus	$\dot{-}$	∞	0	0	$-$	difference

Table 1.1: Binary Operations

To denote an operation on multisets, we will use the usual denotation of the corresponding operation on sets. That is, if a set is interpreted as a multiset, the operation on the interpretation is denoted by the same symbol as the operation on the set. To distinguish between set multiset comprehension and finite multiset comprehension, braces ($\{,\}$) will be used to denote the former and square brackets ([,]) to denote the latter.

DEFINITION 1.4.4. Let A be a set. Let $M \in \mathbf{Mst}(A)$, $N \in \mathbf{Mst}(A)$.

1. The *empty* multiset $\emptyset \in \mathbf{Mst}(A)$ is the constant 0 function.

2. We use $A \in \mathbf{Mst}(A)$ to denote the constant ∞ function.

3. The unary operation *complement* ($\complement$) is defined by: $\complement(M) =^{\mathrm{def}} \boldsymbol{\lambda} a \in A.\neg(M(a))$.

4. The binary operations *intersection* ($\cap$), *union* ($\cup$), *sum* ($\uplus$) and *difference* ($-$) are defined by: $M \circledast N =^{\mathrm{def}} \boldsymbol{\lambda} a \in A \boldsymbol{.} M(a) * N(a)$ via the correspondence in Table 1.1.

5. Let P be a property on A. We employ the notation for set comprehension to denote *set multiset comprehension*: $\{a \in A \boldsymbol{.} P(a)\}$ denotes the multiset such that $a \in A$ has multiplicity ∞ in case $P(a)$ and multiplicity 0 otherwise. As usual, a string σ on A enclosed in set multiset forming braces, $\{f\}$, denotes the set multiset in which $a \in A$ has multiplicity ∞ if it is in the image of σ.

6. Similar to the notation for set multisets, a sequence f enclosed in finite multiset forming brackets, $[f]$, denotes the finite multiset in which a has a multiplicity according to how many times it occurs in the image of f, i.e. its multiplicity is $\# f^{-1}(a)$.

EXAMPLE 1.4.5. All three of $[\,]$, $\{\,\}$ and $\emptyset$ can be used to denote the empty multiset. The finite multiset in which a has multiplicity one and b has multiplicity two is denoted by, among others, $[a, b, b]$ and $[b, a, b]$. The order of the elements is not important, but the number of occurrences is. In case of set multisets also the number of occurrences is not important. We have $\{a, a\} = \{a\}$, since both denote the set multiset in which a has multiplicity ∞, but $[a, a] \neq [a]$. Multiset subtraction and complement are somewhat tricky because they may involve subtracting infinity from infinity. In our case it is convenient to define this to be zero. We then have $[a, a] - [a] = [a]$, $\{a\} - [a, a] = \{a\}$, and $[a, a] - \{a\} = \{a\} - \{a\} = \emptyset$.

Next we state some properties of the operations. For the terminology employed, see any book on lattices, e.g. [DP90].

PROPOSITION 1.4.6. *Let A be a set.*

1. *The class* $\mathbf{FMst}(A)$ *of finite multisets on A is closed with respect to arbitrary intersections and minima, and with respect to finite unions and maxima.*

2. *The class* $\mathbf{SMst}(A)$ *of set multisets on A is closed with respect to complements and with respect to arbitrary intersections, minima, unions and maxima.*

3. *The multiset intersection, union and sum are monotone, i.e. they preserve the submultiset relation, in both arguments. The multiset complement is anti-monotone. The multiset difference is monotone in its first and anti-monotone in its second argument.*

4. *The structure* $\langle \mathbf{Mst}(A), \cap, \cup \rangle$ *is a distributive lattice. The same holds for the substructure induced by* $\mathbf{FMst}(A)$.

5. *The structure* $\langle \mathbf{SMst}(A), \cap, \cup, \complement, \emptyset, A\rangle$. *is a Boolean algebra.*

6. *The structure* $\langle \mathbf{Mst}(A), \uplus, \emptyset\rangle$ *is a commutative monoid.*

7. *Let* $M \in \mathbf{Mst}(A)$, $N \in \mathbf{Mst}(A)$ *and* $X \in \mathbf{Mst}(A)$.
   (a) $(M \cap N) \uplus X = (M \uplus X) \cap (N \uplus X)$
   (b) $(M \cap N) - X = (M - X) \cap (N - X)$
   (c) $(M - N) - X = M - (N \uplus X)$
   (d) *Split:*
   $M = (M \cap N) \uplus (M - N)$

8. *Let* $M \in \mathbf{Mst}(A)$, $N \in \mathbf{Mst}(A)$ *and* $S \in \mathbf{SMst}(A)$.
   (a) $(M \uplus N) \cap S = (M \cap S) \uplus (N \cap S)$
   (b) $(M - S) \cap N = (M \cap N) - (S \cap N)$
   (c) *Exchange:*
   $(M - N) \cap S = (M \cap S) - N$
   (d) *Distribute:*
   $(M \uplus N) - S = (M - S) \uplus (N - S)$
   (e) $(M \uplus S) - N = (M - N) \uplus (S - N)$
   (f) $\emptyset = (M \cap S) \cap (M - S)$

PROOF The properties are easily verified by checking the corresponding properties for the extended natural numbers. For example, for 7d, $m = (m \wedge n) + (m \dot{-} n)$ must be shown. Four cases are distinguished, depending on whether m and n are natural numbers (i.e. not equal to ∞) or not. In case both are, we distinguish two cases: $m = n + (m \dot{-} n)$, if $m > n$ and $m = m + 0$, otherwise. $\odot$

1.4.1. Multiset Extensions

We now consider multiset extensions, that is, functions mapping a relation on a set A to a relation on the set $\mathbf{Mst}(A)$ of multisets over A. In the next chapter we need multiset extensions satisfying a number of properties in order to obtain a 'confluence by local confluence' proof method using such an extension. The most important property we need for the extension, is that it preserves well-foundedness. We present two multiset extensions which are known to be well-foundedness preserving. The first one is the *standard* multiset extension which corresponds to the *multiset ordering* $\gg$ of Dershowitz and Manna [DM79]. The other one is the *maximal* multiset extension which appears in Jouannaud and Lescanne [JL82] and is denoted there by $\prec\!\!\prec_{\mathfrak{M}}$. We show that the standard multiset extension satisfies all the properties needed in the next chapter, but that, contrary to what was claimed in [Oos94], the maximal multiset extension does not. We introduce some special notation to allow for algebraic proofs of the properties we need.

Standard Extension

DEFINITION 1.4.7. Let $\succ$ be an order on a set A.

1. Let $M \in \mathbf{Mst}(A)$. The set multiset $\curlyvee|M$ is the *down-multiset* (cf. [DP90]) generated by M, and is defined by $\curlyvee|M =^{\mathrm{def}} \{b \in A.\exists a \in M.a \succeq b\}$. Similarly, the *strict* down-multiset generated by M is defined by $\curlyvee M =^{\mathrm{def}} \{b \in A.\exists a \in M.a \succ b\}$. Note that the result of these operations are set multisets. We write $\curlyvee|a$ and $\curlyvee a$ for $\curlyvee|\{a\}$ and $\curlyvee\{a\}$. A multiset M is ($\curlyvee|$-)*closed* if $M = \curlyvee|M$.

2. The *standard* multiset extension of $\succ$, is the relation $\succ_{mul}$ on $\mathbf{Mst}(A)$ which is defined by $N \succ_{mul} M =^{\mathrm{def}} \exists X.\exists Y.\exists Z.M = Z \uplus X \;\&\; N = Z \uplus Y \;\&\; X \subseteq \curlyvee Y \;\&\; Y \neq \emptyset$ for multisets M and N on A. We say that X, Y, Z is a *proof* for $N \succ_{mul} M$. Furthermore, $\succeq_{mul}$ will be used to denote the reflexive closure of $\succ_{mul}$. Note that this is different from first taking the reflexive closure of $\succ$ and then the standard multiset extension. It is easy to see that $\succeq_{mul}$ can also be obtained by leaving out the condition $Y \neq \emptyset$ in the definition of $\succ_{mul}$ and this gives rise to a notion of *proof* of $N \succeq_{mul} M$. The relations can be seen as the closure under contexts $(Z \uplus \cdot)$ of the *domination* order $(X \subseteq \curlyvee Y)$.

Note that a proof X, Y, Z of $M \succ_{mul} N$ ($M \succeq_{mul} N$) need not be unique. Both $[0]$, $[1]$, $[0]$, and $[0,0]$, $[0,1]$, $\emptyset$, are proofs of $[0,1] \succ_{mul} [0,0]$. The first proof satisfies the extra condition that $X \cap Y = [0] \cap [1] = \emptyset$. In general, if C, D, E is a proof of $F \succ_{mul} G$ ($F \succeq_{mul} G$) then $C' =^{\mathrm{def}} C - D$, $D' =^{\mathrm{def}} D - C$, $E' =^{\mathrm{def}} (C \cap D) \uplus E$ is the unique proof of that relation, satisfying $C' \cap D' = \emptyset$.

1. $C' \uplus E' = (C - D) \uplus (C \cap D) \uplus E = C \uplus E = F$,

2. $D' \uplus E' = (D - C) \uplus (C \cap D) \uplus E = D \uplus E = G$,

3. $C - D \subseteq \curlyvee D - D \subseteq \curlyvee D = \curlyvee(D - \curlyvee D) \subseteq \curlyvee(D - C)$,

4. $D - C = \emptyset \iff D \subseteq C \implies D \subseteq \curlyvee D \implies \bot$.

That it indeed is the unique proof satisfying $C' \cap D' = \emptyset$ follows easily. In the proof, we have made essential use of finiteness (remember that F and G range over finite multisets) in the third step: $\curlyvee D = \curlyvee(D - \curlyvee D)$ only holds for multisets which do not have infinite ascending chains of elements.

In the next proposition, we show that $\curlyvee|$ is a so-called *topological closure operator* (see e.g. [Smy92]) and we show its relationship with $\curlyvee$.

PROPOSITION 1.4.8. *Let M, N be multisets. Then the operations $\curlyvee|$ and $\curlyvee$ have the following properties:*

1. *extensivity:* $M \subseteq \curlyvee|M$,

2. *monotonicity:* $M \subseteq N \implies \curlyvee|M \subseteq \curlyvee|N \;\&\; \curlyvee M \subseteq \curlyvee N$,

3. *idempotence:* $\curlyvee\!\shortmid(\curlyvee\!\shortmid M) = \curlyvee\!\shortmid M$ *and* $\curlyvee\!\shortmid(\curlyvee M) = \curlyvee M = \curlyvee(\curlyvee\!\shortmid M)$,

4. *strictness:* $\curlyvee\!\shortmid\emptyset = \emptyset$,

5. *additivity:* $\curlyvee\!\shortmid(M \cup N) = \curlyvee\!\shortmid M \cup \curlyvee\!\shortmid N$ *and* $\curlyvee\!\shortmid(M \uplus N) = \curlyvee\!\shortmid M \uplus \curlyvee\!\shortmid N$,

6. $\curlyvee(M - N) \supseteq \curlyvee M - \curlyvee N$,

7. $\curlyvee M \cap \curlyvee N \supseteq \curlyvee(M \cap N)$,

PROOF 1. by reflexivity of $\succeq$,

2. by definition,

3. by transitivity and reflexivity of $\succeq$ ($\succ$).

4. by definition,

5. easy, using $a \in M \cup N \iff a \in M$ or $a \in N \iff a \in M \uplus N$,

6. if $b \in \curlyvee M - \curlyvee N$, then there exists a a in $M - N$ such that $a \succ b$. This suffices, because both sides are set multisets,

7. if $b \in M \cap N$, then also $b \in M$ and $b \in N$.

$\odot$

The following proposition is standard. Nevertheless we give its proof, because we like it for its algebraic nature.

PROPOSITION 1.4.9. *The standard multiset extension preserves strict orders on* $\mathbf{FMst}(A)$. *For strict orders it preserves well-foundedness.*

PROOF Let A be a set. Let $\succ$ be a relation on A. The variables F, G, C, D and E range over $\mathbf{FMst}(A)$.

1. We prove: if $\succ$ is a strict order, then $\succ_{mul}$ is irreflexive on $\mathbf{FMst}(A)$.

$$
\begin{array}{ll}
F \succ_{mul} F & \\
\iff & \exists C.\exists D.\exists E.F = E \uplus C, F = E \uplus D, C \subseteq \curlyvee D \;\&\; D \neq \emptyset \\
\implies & \exists C.\exists D.C = D, C \subseteq \curlyvee D \;\&\; D \neq \emptyset \\
\implies & \exists C.C \subseteq \curlyvee C \;\&\; C \neq \emptyset \\
\implies & \bot
\end{array}
$$

2. We prove: if $\succ$ is transitive, then $\succ_{mul}$ is transitive on $\mathbf{FMst}(A)$. Let $F_1 \succ_{mul} F_0$ by the proof C_0, D_0, E_0 and let $F_2 \succ_{mul} F_1$ by the proof C_1, D_1, E_1. We claim that $C =^{\text{def}} C_0 \uplus (E_0 - E_1)$, $D =^{\text{def}} D_1 \uplus (E_1 - E_0)$, $E =^{\text{def}} E_0 \cap E_1$ is a proof for $F_2 \succ_{mul} F_0$.

   (a) $C \uplus E = (C_0 \uplus (E_0 - E_1)) \uplus (E_0 \cap E_1) = C_0 \uplus E_0 = F_0$

(b) $D \uplus E = (D_1 \uplus (E_1 - E_0)) \uplus (E_1 \cap E_0) = D_1 \uplus E_1 = F_2$

(c)

$$\begin{array}{l} C \subseteq \curlyvee D \\ \quad \iff \; C_0 \uplus (E_0 - E_1) \subseteq \curlyvee D \\ \quad \iff \; C_0 \subseteq \curlyvee D \;\&\; E_0 - E_1 \subseteq \curlyvee D \end{array}$$

First note that we have

$$\begin{array}{l} D_0 \uplus E_0 = F_1 = C_1 \uplus E_1 \\ \quad \Longrightarrow \; (D_0 \uplus E_0) - (E_0 \uplus C_1) = (C_1 \uplus E_1) - (E_0 \uplus C_1) \\ \quad \Longrightarrow \; ((D_0 \uplus E_0) - E_0) - C_1 = ((E_1 \uplus C_1) - C_1) - E_0 \\ \quad \Longrightarrow \; D_0 - C_1 = E_1 - E_0 \end{array}$$

Now we can compute

$$\begin{array}{rcl} C_0 & \subseteq & \curlyvee D_0 \\ & = & \curlyvee((D_0 \cap C_1) \uplus (D_0 - C_1)) \\ & = & \curlyvee(D_0 \cap C_1) \uplus \curlyvee(D_0 - C_1) \\ & \subseteq & \curlyvee C_1 \uplus \curlyvee(D_0 - C_1) \\ & \subseteq & \curlyvee D_1 \uplus \curlyvee(E_1 - E_0) \end{array}$$

and

$$\begin{array}{rcl} E_0 - E_1 & \subseteq & (C_1 \uplus E_1) - E_1 \\ & \subseteq & C_1 \\ & \subseteq & \curlyvee D_1 \end{array}$$

(d) Finally, $D \neq \emptyset$ because $D_1 \neq \emptyset$.

3. Preservation of well-foundedness is what makes the standard multiset extension interesting. It can be proved in various ways. See [DM79] for a proof which uses König's Lemma. A proof can also be obtained from Kruskal's Tree Theorem. For example, the elegant proof by example in [NW63] can easily be adapted if one takes instead of the orders $\geqslant$ on Q and its extension $\geqslant$ on SQ there, the orders $\succeq$ on A and $\succeq_{mul}$ on $\mathbf{FMst}(A)$.

$\odot$

REMARK 1.4.10. The standard multiset extension as defined above does preserve transitivity, but the argument is more complicated because we do not have $(M \uplus N) - N = M$ in general, as exemplified by $M =^{\text{def}} N =^{\text{def}} \{a\}$. On the other hand, irreflexivity is not preserved as witnessed by $\{a\} \succ_{mul} \{a\}$, by the proof $\emptyset$, $[a]$, $\{a\}$. This shows that the definition is not the right one if one is interested in preservation of (well-founded) strict orders. An interesting question is of course whether one can devise a multiset extension which does preserve well-founded strict orders on $\mathbf{Mst}$.

In the next proposition some multiset operations are shown to be $\succ_{mul}$-preserving (in a limited way) and we prove an interpolation result.

PROPOSITION 1.4.11. *1.* $N \supseteq M \implies N \succeq_{mul} M$,

2. Extensivity: $\curlyvee\!|M \succeq_{mul} M$,

3. Monotonicity: $N \succeq_{mul} M \implies \curlyvee\!|N \succeq_{mul} \curlyvee\!|M$,

4. Noise reduction:
$G \succ_{mul} F$, *if* $F \subseteq \curlyvee G$ *and* $G \neq \emptyset$,

5. Downward closed set subtraction:
$G \succeq_{mul} F \implies G - S \succeq_{mul} F - S$, *if* $\curlyvee S \subseteq S$,

6. Finite submultiset subtraction:
$G \succeq_{mul} F \iff G - H \succeq_{mul} F - H$, *if* $H \subseteq F$ *and* $H \subseteq G$. *The statement also holds if one replaces* $\succeq_{mul}$ *by* $\succ_{mul}$,

7. Interpolate:
$G \succeq_{mul} F \iff G \succeq_{mul} F \uplus H$, *if* $H \subseteq \curlyvee G - \curlyvee F$. *The statement also holds if one replaces* $\succeq_{mul}$ *by* $\succ_{mul}$.

PROOF 1. $\emptyset$, $N - M$, M is a proof for $N \succeq_{mul} M$ if $N \supseteq M$, because $M \uplus (N - M) = (N \cap M) \uplus (N - M) = N$,

2. By $\curlyvee\!|M \supseteq M$ and the first item.

3. Let X, Y, Z be a proof for $N \succeq_{mul} M$. Then $\curlyvee\!|X$, $\curlyvee\!|Y$, $\curlyvee\!|Z$ is a proof for $\curlyvee\!|N \succeq_{mul} \curlyvee\!|M$ by additivity of $\curlyvee\!|$ and because $\curlyvee\!|X \subseteq \curlyvee(\curlyvee\!|Y) = \curlyvee\!|(\curlyvee Y)$ using monotonicity and idempotence of $\curlyvee\!|$,

4. F, G, $\emptyset$ is a proof for $G \succ_{mul} F$,

5. '$\implies$' Let C, D, E be a proof for $G \succeq_{mul} F$ and let $\curlyvee S \subseteq S$. Then $C - S$, $D - S$, $E - S$ is a proof for $G - S \succeq_{mul} F - S$, because

   (a) $(C - S) \uplus (E - S) = (C \uplus E) - S = F - S$,
   (b) $(D - S) \uplus (E - S) = (D \uplus E) - S = G - S$,
   (c) $C - S \subseteq \curlyvee D - \curlyvee S \subseteq \curlyvee(D - S)$

6. '$\impliedby$' Let C, D, E be a proof of $G - H \succeq_{mul} F - H$. Then C, D, $E \uplus H$ is a proof of $G \succeq_{mul} F$, because

   (a) $C \uplus (E \uplus H) = (F - H) \uplus H = F$,
   (b) $D \uplus (E \uplus H) = (G - H) \uplus H = G$.

The other condition(s) hold(s) trivially.
'$\Longrightarrow$' Let C, D, E be a proof of $G \succeq_{mul} F$ such that $C \cap D = \emptyset$. Then C, D, $E - H$ is a proof of $G - H \succeq_{mul} F - H$. Because $H \subseteq F$ and $H \subseteq G$, we have $H = H \cap (C \uplus E) \cap (D \uplus E) = H \cap ((C \cap D) \uplus E) = H \cap E$, i.e. $H \subseteq E$.

(a) $C \uplus (E - H) = (C \uplus E) - H = F - H$,

(b) $D \uplus (E - H) = (D \uplus E) - H = G - H$.

The other condition(s) hold(s) trivially.

7. '$\Longleftarrow$' $G \succeq_{mul} F \uplus H \succeq_{mul} F$.
'$\Longrightarrow$' Let C, D, E be a proof of $G \succeq_{mul} F$. Then $H \uplus C$, D, E is a proof of $G \succeq_{mul} F \uplus H$, if $H \subseteq \curlyvee G - \curlyvee F$. The first two conditions and the last condition are trivial.

(a)
$$
\begin{aligned}
& (H \uplus C) \subseteq \curlyvee D \Longleftarrow \\
\Longleftarrow \quad & H \subseteq \curlyvee D \\
\Longleftarrow \quad & \curlyvee G - \curlyvee F \subseteq \curlyvee D \\
\Longleftrightarrow \quad & (\curlyvee D \uplus \curlyvee E) - (\curlyvee C \uplus \curlyvee E) \subseteq \curlyvee D
\end{aligned}
$$

$\odot$

REMARK 1.4.12. The inverse implication of Proposition 1.4.11(5) does not hold, as witnessed by $\emptyset - \{a\} \succeq_{mul} [a] - \{a\}$, but not $\emptyset \succeq_{mul} [a]$.

Maximal Extension

DEFINITION 1.4.13. Let A be a set. Let $\succ =^{\mathrm{def}} \langle A, \succ \rangle$ be an order on A.

1. Let $M \in \mathbf{Mst}(A)$. Define the *boundary* of M by $\partial M =^{\mathrm{def}} M - \curlyvee M$. The boundary of a multiset consists of its maximal elements. Define the *interior* of M by $\overset{\circ}{M} =^{\mathrm{def}} M \cap \curlyvee M$. The interior of a multiset consists of its non-maximal elements.

2. The *maximal* multiset extension of $\succ$, is the relation $\succ_m$ on $\mathbf{Mst}(A)$ which is defined by $N \succ_m M =^{\mathrm{def}} \partial N \succ_{mul} \partial M$ or $(\partial N = \partial M \ \& \ \overset{\circ}{N} \succ_m \overset{\circ}{M})$ for multisets M and N on A. In words this reads, first compare the maximal elements of the multisets and only if this is not decisive apply the method recursively to the non-maximal elements.

The reader might enjoy checking that the maximal multiset extension satisfies all properties which were shown to hold for the standard extension. All properties, save 1.4.11(6). Submultiset subtraction and, worse, multiset sum do not preserve order as shown by the following example.

EXAMPLE 1.4.14. Let $b \succ a$ and $3 \succ 2 \succ 1$. The order $\succ_m$ is not closed with respect to submultiset subtraction. A counterexample is provided by $[2, b] \succ_m [2, a, 1]$, but not $[b] \succ_m [a, 1]$. The order $\succ_m$ is not closed with respect to multiset sum. A counterexample is provided by $[2, b] \succ_m [1, b, a]$, but not $[3, 2, b] \succ_m [3, 1, b, a]$.

The example entails that the maximal multiset extension is useless for the purposes of the next chapter. One might wonder whether there are useful multiset extensions other than the standard one. We do not know whether this is the case, but see [Mar87, Mar89] for examples of multiset extensions.

2 Abstract Rewriting

The intention of this chapter is to provide methods for proving confluence of computation systems. A *computation system* transforms *input* to *output* via a *computation*. A computation is a sequence of *computation steps*. From this definition several questions arise naturally. The exact questions depend on the kind of transformations considered.

The 'simplest' kind of computation system implements transformations from finite input to finite output. For example, computing the square of a natural number. The output of such computations can be seen 'at once' and is called the result of the computation. Some natural questions for such computation systems are:

1. Is a result always obtained after a finite number of steps? (*strong normalisation*)

2. Somewhat weaker, does an effective computation strategy exist which always yields a result after finitely many steps? (*normalising strategy*)

3. Or, still weaker, do we know (one way or another) that it is always possible to reach a result in a finite number of steps? (*weak normalisation*)

More 'difficult' computation systems implement transforming infinite input into infinite output. For example, computing the square root of a natural number. The output of such computations can not be seen at once, instead the outcome is a growing sequence of symbols, approximating the infinite result. In general one can consider (sequential) computation systems to transform an input stream into an output stream. The above questions now generalise to:

1. Does it always take a finite number of steps to produce the next symbol on the output? (*infinite strong normalisation*)

2. Does an effective computation strategy exists which always produces a symbol after a finite number of steps? (*infinite normalising strategy*)

3. Is it always possible to produce the next symbol on the output after finitely many steps? (*infinite weak normalisation*)

(Of course, one can vary these questions, e.g. by expressing dependency of the output on the input.)

In general the theory of computation should comprise both finite and infinite computation systems, but the rewriting theory seems to be well-developed for the finite case only. However interesting the infinite case, we restrict ourselves to the finite one.

In this thesis we do not address the above *termination* questions, but concentrate on *uniqueness questions*, arising in the case of e.g. *ambiguous* or *concurrent* computations. That is, at some computation stage there are several possible computation steps that can be performed next (concurrently). The question is whether the actual choice of which step to perform next influences the result. A positive answer to the any of the following questions suffices for uniqueness.

1. If at some computation stage there is a choice, is it for every pair of 'diverging' computations possible to find a pair of 'converging' computations? (*confluence*)
2. If there is a choice between two steps, and the first one leads to a computation of a result, is the second one also the start of a computation of that result? (*normal form property*)
3. If at some computation stage there is a choice of which computation step to perform next, is it not possible for the actual choice to influence the result? (*uniqueness of normal forms*)
4. Like the third item, but now one is also allowed to choose between 'backward' computation steps (like in backtracking, but without the necessity to undo the last 'forward' step). (*uniqueness of normal forms with respect to conversion*)

2.1. Abstract Rewriting Systems

The first thing to do is to set up an appropriate framework, first for stating the questions above, and then for answering them. It is standard to view the finite computation systems mentioned above as relations.

DEFINITION 2.1.1. An *abstract rewriting system* (ARS) is structure $\langle A, \longrightarrow \rangle$, where A is a set of *objects* and $\longrightarrow \in \mathfrak{P}(A \times A)$ is a set of *rewrite steps*. We use $\rightarrow$, $\rightsquigarrow$, $\Rightarrow$, and $\rightarrowtail$ to range over ARSs and u, v, w, r to range over rewrite steps.

Note that an abstract rewriting system is just a binary relation on a set. For an ARS to be useful as a computation system one also wants everything to be *effective*, i.e. whether two objects are equal, the first can be rewritten in one step to the latter, or whether an object can not be rewritten at all, should be decidable. Cf. also [Klo90, Exe. 1.9.4].

REMARK 2.1.2. Abstract rewriting systems occur under various names in the literature. They are called *general replacement systems* by Rosen in [Ros73], *replacement systems* by Staples in [Sta74], and *reduction systems* by Jantzen in [Jan88] and by Curien and Ghelli in [CG91]. I prefer to call them abstract rewriting systems. First, nowadays 'abstract' and 'rewriting' are more common than 'general' and 'replacement'. Second, 'reduction' has an unwanted termination connotation.

Both Klop ([Klo92]) and Gonthier, Lévy and Melliès ([GLM92]) introduce *abstract reduction systems* as ARSs having some extra (but different in both cases) structure. That is, they are not as abstract as possible. I prefer to add extra structure when needed and then to change the name of the considered class of rewrite systems accordingly. For example, when the objects are (first-order) terms over some alphabet and the rewrite steps are induced by a set of rules, we will speak of *term rewriting systems*.

Because an abstract rewriting system is just a relation, the usual operations on relations, such as product, sum and composition, pertain. For the operations on and properties of abstract rewriting systems specific to the area of rewriting, we fix our terminology in this section.

There are essentially two ways to restrict a relation on a set of objects. First, one can restrict the set of objects and take the appropriate restriction on the graph. Second, one can restrict the graph of the relation. Analogous to category theory, one can call the former a *full* subrelation and the latter a *wide* subrelation. Of course, one can get any subrelation by taking first a full and then a wide subrelation, but most of the time only one way of restriction is needed.

For an abstract rewriting system a *substructure* ARS, or *sub*-ARS for short, is a full subrelation which is closed with respect to rewriting. A *strategy* is nothing more or less than a wide subrelation. If one views a strategy as the restriction of the set of choices offered at each object, it is natural to require that a strategy preserves choice, that is, the restriction of a non-empty set of choices is non-empty again.

Some special notation supporting the rewriting intuition is in common use nowadays.

DEFINITION 2.1.3. Let $\rightarrow$ be an abstract rewriting system. Then

1. $\rightarrow$ is the *one-step rewrite* relation.
2. $\leftarrow =^{\mathrm{def}} \rightarrow^{-1}$.
3. $\leftrightarrow =^{\mathrm{def}} \rightarrow \cup \leftarrow$.
4. $\twoheadrightarrow =^{\mathrm{def}} \rightarrow^{*}$. This is the *rewrite* or *reachability* relation.
5. $\leftrightarrow^{*}$ is the *convertibility* relation.
6. $\downarrow =^{\mathrm{def}} \rightarrow ; \leftarrow$. This is the *one-step joinability* relation.
7. $\mathrel{\downarrow\!\!\!\!\downarrow} =^{\mathrm{def}} \twoheadrightarrow ; \twoheadleftarrow$. This is the *common descendant* or *joinability* relation.
8. $\uparrow =^{\mathrm{def}} \leftarrow ; \rightarrow$. This is the *one-step meetability* or *local divergence* relation.
9. $\mathrel{\uparrow\!\!\!\!\uparrow} =^{\mathrm{def}} \twoheadleftarrow ; \twoheadrightarrow$. This is the *common ancestor*, *meetability* or *divergence* relation.

The symbol to denote the inverse of an ARS is obtained by 'inverting' the symbol to denote the ARS. The symbol to denote the symmetric closure of an ARS, i.e. the union of an ARS and its inverse, is obtained by 'uniting' the symbols to denote the ARS and its inverse. The symbol to denote the reflexive, transitive closure of an ARS, i.e. the union of arbitrarily often repeated rewrite steps, is obtained by 'repeating' the symbol to denote the ARS.

REMARK 2.1.4. The notations in rewriting theory are still not very well-established. In [DJ91] a contribution is made to the development of good notations for term rewriting and related areas. We will use the notations presented there most of the time, and try to justify our deviation from these otherwise.

In [DJ91] $\uparrow$ and $\downarrow$ are used to denote meetability and joinability, instead of our $\twoheaduparrow$ and $\twoheaddownarrow$. We think our notation is more compositional.

DEFINITION 2.1.5. Let $\rightarrow =^{\text{def}} \langle A, \longrightarrow \rangle$ be an ARS. Let α be an ordinal not larger than ω. A $\rightarrow$-*conversion* of *length* α, *issuing from* $a \in A$ is a structure $\langle a, d \rangle$ such that

1. d is a $(\longrightarrow \times \{0\}) \cup (\longleftarrow \times \{1\})$-sequence of length α, i.e. ordinals are mapped into the disjoint union of the sets of rewrite steps and inverse rewrite steps. The length of a conversion is the length of its sequence.

2. If $\alpha \neq 0$, then $\pi_1(d(0)) = a$. The sequence of a conversion issuing from a must start with a.

3. $\forall \beta . \beta + 1 < \alpha \implies \pi_2(d(\beta)) = \pi_1(d(\beta + 1))$. The startings and endings of the individual steps in the sequence must match.

The *start* a of the conversion is denoted by $\mathsf{start}(\langle a, d \rangle)$. For defining its *end*, $\mathsf{end}(\langle a, d \rangle)$, we distinguish three cases: If $\alpha = 0$, then its end is a. If $\alpha = \beta + 1$, for some ordinal β, then its end is $\pi_2(d(\beta))$. Otherwise the conversion is infinite and has an open ending, i.e. end is not defined. The former two are called *finite* conversions, the last one an *infinite* conversion. If the start of a conversion is understood from the context, we will confuse the conversion with its sequence. The only case in which the conversion cannot be recovered from its sequence is for the *empty conversion* $0_a =^{\text{def}} \langle a, \varepsilon \rangle$. We use d, e, f, g and h to range over the class of $\rightarrow$-conversions $\mathbf{Cnv}(\rightarrow)$. If d consists completely out of rewrite steps, then it is called a $\rightarrow$-*rewrite*, *derivation*, *path*, or *walk*. The class of $\rightarrow$-rewrites issuing from a is called the *derivations space of* a and denoted by $\mathbf{Rew}(a)$ (cf. [HL91a]). The class of all $\rightarrow$-rewrites is denoted by $\mathbf{Rew}(\rightarrow)$. Let d of length α and e of length β be conversions.

1. To state that d is a finite conversion issuing from a, ending in b and of length α, we write $d\colon a \leftrightarrow^{\alpha} b$. To indicate that it is moreover a rewrite, we write $d\colon a \rightarrow^{\alpha} b$. To state that d is an infinite conversion (rewrite) issuing from a (so of length ω), we write $d\colon a \leftrightarrow^{\omega} \ldots$ ($d\colon a \rightarrow^{\omega} \ldots$).

2. The *inverse* of d, d^{-1} of length α is defined by inverting its sequence, as expected. Formally, one can define: $\langle \mathsf{end}(d), \boldsymbol{\lambda}\gamma \in \alpha.d(\alpha \dot{-} (\gamma+1))^{-1}\rangle$, where $((a,b),0)^{-1} =^{\text{def}} ((b,a),1)$ and vice versa. Inverse is involutive.

3. If $\mathsf{end}(d) = \mathsf{start}(e)$, then the *concatenation* of d and e of length $\alpha + \beta$ is defined as expected and denoted by $d\,;e$. Concatenation is associative.

4. Two rewrites having the same start (end) are said to be *coinitial* (*cofinal*). A tuple of coinitial (cofinal) rewrites is called a *divergence* or *peak* (*convergence* or *valley*). A quadruple consisting of a divergence and a convergence such that the rewrites can be pairwise concatenated is called a *confluence diagram*.

Remark 2.1.6. The definition above only defines conversions up to infinity. It is clearly unsatisfactory for transfinite conversions, because nothing is specified how to connect steps at limit ordinals to the steps approaching them. A solution is to introduce a notion of approximation on the domain, that is, the domain of the ARS should be made into a topological space rather than just a set. Moreover, one can refine this by taking the 'distance' between the starting and ending objects of rewrite steps into account. For such an abstract approach using metric spaces see [Ken92]. For the case of transfinite rewriting of infinite first-order terms the theory is rather well-developed (see [DKP91, KKSV93, FW91], but in general the topic of transfinite rewriting seems to be largely unexplored. It would be interesting to extend the existing theory to structures such as Graph Rewriting Systems or the Higher-Order Rewriting Systems of the next chapter.

Remark 2.1.7. Transfinite rewriting presents problems. For one, it destroys a lot of commutativity properties. Whereas for finite rewriting, one easily checks $(\rightarrow^{-1})^* = (\rightarrow^*)^{-1}$ (which justifies writing $\leftarrow^*$ instead of ${}^*\!\leftarrow$), this fails in the case of transfinite rewriting (cf. [Ken91]). Intuitively, suppose the ARS $\rightarrow$ computes the square root of 2 by producing one digit at the time. We then have $a \rightarrow^\omega \sqrt{2}$, but not $\sqrt{2} \leftarrow^\omega a$, because no steps can be performed starting from the infinite result $\sqrt{2}$. So $\leftarrow^\omega \neq {}^\omega\!\leftarrow$.

Note that we have introduced some confusion in notation between m-fold composition of an ARS $\rightarrow$ ($\leftrightarrow$) on A and a $\rightarrow$-rewrite ($\rightarrow$-conversion) of length m. This is harmless as one easily checks

$$a \rightarrow^m b \iff \exists d \in \mathbf{Rew}(\rightarrow).d\colon a \rightarrow^m b$$

$$a \leftrightarrow^m b \iff \exists d \in \mathbf{Cnv}(\rightarrow).d\colon a \leftrightarrow^m b$$

for $a \in A$ and $b \in B$.

Definition 2.1.8. Let $\rightarrow =^{\text{def}} \langle A, \longrightarrow\rangle$ be an ARS. Let $a \in A$, $b \in B$.

1. The set $\mathsf{Stp}(a)$ of *rewrite steps issuing from* a is defined by $\mathsf{Stp}(a) =^{\mathrm{def}} \rightarrow \cap (\{a\} \times A)$.
2. The object a is a ($\rightarrow$-)*normal form*, $a \not\rightarrow$, if $\mathsf{Stp}(a) = \emptyset$.
3. If $a \twoheadrightarrow b$ and $b \not\rightarrow$, then we write $a \twoheadrightarrow^{!} b$ and say that a *has the normal form* b or b *is a normal form of* a.
4. If a has a unique normal form b, then $a{\downarrow} =^{\mathrm{def}} b$. Otherwise $a{\downarrow}$ is undefined.
5. The *rewrite graph below* a is defined by $\langle a \twoheadrightarrow \rangle =^{\mathrm{def}} \rightarrow \restriction \{a \twoheadrightarrow\}$.

The rewrite graph below an object a is the abstract rewriting system consisting of the objects reachable from a by rewriting, and the appropriately restricted graph.

REMARK 2.1.9. The expressions $\{a{\downarrow}\}$ and $\{a\ {\downarrow}\}$ mean different things. The former denotes the one-element set containing the normal form of a, while the latter denotes the set of objects which are one-step joinable with a. The distinction between these expressions is rather subtle. Fortunately, we never use such (almost) ambiguous expressions.

2.2. Properties of Rewriting Systems

Abstract rewriting systems are an abstraction from particular rewriting systems derived from equational theories such as Curry's Combinatory Logic ([Cur30a, Cur30b]) and Church's Lambda Calculus ([Chu32, Chu33]). Properties of particular rewriting systems corresponding to these theories were used to prove a weak form of *consistency*, that is, to prove that there are non-equivalent objects. For a more complete survey of abstract rewriting properties and their interrelationship, see [Klo92].

DEFINITION 2.2.1. Let $\rightarrow$ be an abstract rewriting system on A.

1. if ${}^{!}\twoheadleftarrow\, ;\, \twoheadrightarrow^{!} \subseteq \mathsf{id}_A$, then $\rightarrow$ *has unique normal forms*, $UN(\rightarrow)$.
2. if $\uparrow \subseteq \downarrow$, then $\rightarrow$ has the *diamond* property, $\Diamond(\rightarrow)$,
3. if $\uparrow \subseteq$ ↡, then $\rightarrow$ is *locally confluent*, $LCON(\rightarrow)$,
4. if ↟ $\subseteq$ ↡, then $\rightarrow$ is *confluent*, $CON(\rightarrow)$,
5. if $\leftrightarrow^{*} \subseteq$ ↡, then $\rightarrow$ has the *Church-Rosser* property, or is Church-Rosser, $CR(\rightarrow)$,

These notions specialise to objects a in A, by taking a as origin of the divergence, for example, a is confluent, if $\twoheadleftarrow a \twoheadrightarrow \subseteq$ ↡, using some obvious notation.

REMARK 2.2.2. In the literature local confluence is also called *weak confluence* or *Weak Church-Rosser*. The definition of confluence as given here originates from [Hue80]. The definition of the Church-Rosser property is the generalisation to abstract rewriting systems of the main property proved by Church and Rosser in [CR36] for λ-calculus. In the literature (cf. [New42]) the last two notions are often used interchangeably because they are easily shown to be equivalent. We prefer to keep them distinct.

DEFINITION 2.2.3. Let $\rightarrow$ be an ARS on A. Let $a \in A$.

1. The object a is *weakly normalising*, $WN(a)$, if it has a normal form. This is extended to $\rightarrow$ by defining $WN(\rightarrow) =^{\mathrm{def}} \forall a \in A. WN(a)$.

2. If there is no infinite rewrite issuing from a then a is called *strongly normalising* (or *terminating*), denoted by $SN(a)$. This is extended to $\rightarrow$ by defining $SN(\rightarrow) =^{\mathrm{def}} \forall a \in A. SN(a)$.

3. *Completeness* is defined by $COMP =^{\mathrm{def}} SN \,\&\, UN$.

In case an abstract rewriting system is normalising, we will sometimes call it a *reduction* relation and its inverse relation an *expansion* relation. Note that in a weakly normalising ARS having unique normal forms, $a{\downarrow}$ is defined for every object a.

An important motivation for our study of confluence is the relationship $CON \iff CR \implies UN$. Uniqueness of normal forms can be used to show the consistency of an equivalence relation and together with termination it can be used to show its decidability.

DEFINITION 2.2.4. Let $\sim$ be an equivalence relation on A. The relation $\sim$ is *consistent*, if there exist two non-$\sim$-equivalent objects. Let $\rightarrow$ be an ARS on A. Then, $\rightarrow$ *implements* $\sim$, if $\sim\, = \,\leftrightarrow^*$. If, moreover $\rightarrow$ is effective and complete, then we say $\rightarrow$ *decides* $\sim$.

Let $\rightarrow$ implement the equivalence relation $\sim$. If $\rightarrow$ has unique normal forms and there are two distinct $\rightarrow$-normal forms, these cannot be $\sim$-equivalent, hence $\sim$ is consistent. If $\rightarrow$ decides $\sim$, one can decide whether $a \sim b$. Just $\rightarrow$-reduce both a and b to their respective $\rightarrow$-normal forms and test these for equality.

The following lemma states some folklore confluence results.

LEMMA 2.2.5. *Let* $\rightarrow, \rightsquigarrow$ *be ARSs.*

1. $\Diamond(\rightarrow) \implies \leftarrow ; \twoheadrightarrow \,\subseteq\, \downarrow$.

2. $\leftarrow ; \twoheadrightarrow \,\subseteq\, \downarrow \implies CON(\rightarrow)$.

3. *Let* $\rightsquigarrow \,\subseteq\, \rightarrow \,\subseteq\, \rightsquigarrow^*$. *Then* $CON(\rightarrow) \iff CON(\rightsquigarrow)$.

PROOF 1. We will give a formal proof for this trivial property, then draw a picture explaining the proof, and conclude that a picture is often both shorter and more intuitive than text. First we prove $\forall m \in \mathbb{N}. \leftarrow ; \rightarrow^m \subseteq \rightarrow^m ; \leftarrow$, by induction on m. The base case follows because the 0-fold composition is a (left- and right-) neutral element for ;.

$$\leftarrow ; \rightarrow^0 = \leftarrow = \rightarrow^0 ; \leftarrow$$

The induction step follows by the assumption and monotonicity and associativity of ;.

$$\begin{aligned}
&\leftarrow ; \rightarrow^{m+1} = \\
&\quad = \quad \uparrow ; \rightarrow^m \\
&\quad \subseteq \quad \downarrow ; \rightarrow^m \\
&\quad = \quad \rightarrow ; (\leftarrow ; \rightarrow^m) \\
&\quad \subseteq \quad \rightarrow ; (\rightarrow^m ; \leftarrow) \\
&\quad = \quad \rightarrow^{m+1} ; \leftarrow
\end{aligned}$$

Now the property follows because $\bigcup$ is monotonic and $\cdot^*$ is increasing.

$$\begin{aligned}
&\leftarrow ; \twoheadrightarrow = \\
&\quad = \quad \leftarrow ; (\bigcup m \in \mathbb{N}. \rightarrow^m) \\
&\quad = \quad \bigcup m \in \mathbb{N}. (\leftarrow ; \rightarrow^m) \\
&\quad \subseteq \quad \bigcup m \in \mathbb{N}. (\rightarrow^m ; \leftarrow) \\
&\quad \subseteq \quad (\bigcup m \in \mathbb{N}. \rightarrow^m) ; \leftarrow \\
&\quad = \quad \twoheadrightarrow ; \leftarrow \\
&\quad \subseteq \quad \twoheaddownarrow
\end{aligned}$$

Note that we actually proved something stronger than needed, in order to make the induction work. The above is all very well, but what really is going on is seen in the following diagram.

$$\begin{tikzcd}
\cdot \arrow[r] \arrow[d] \arrow[dr, phantom, "\mathrm{Ass}"] & \cdot \arrow[r, "m"] \arrow[d, dashed] \arrow[dr, phantom, "\mathrm{IH}"] & \cdot \arrow[d, dashed] \\
\cdot \arrow[r, dashed] & \cdot \arrow[r, dashed, "m"] & \cdot
\end{tikzcd}$$

Henceforth we will draw diagrams instead of giving tedious induction proofs if possible.

2. By the following diagram.

$$\begin{tikzcd} \cdot \arrow[r] \arrow[d, ->>] \arrow[dr, phantom, "\mathrm{Ass}"] & \cdot \arrow[r, "m"] \arrow[d, dashed, ->>] \arrow[dr, phantom, "\mathrm{IH}"] & \cdot \arrow[d, dashed, ->>] \\ \cdot \arrow[r, dashed, ->>] & \cdot \arrow[r, dashed, ->>] & \cdot \end{tikzcd}$$

3. By monotonicity and idempotence of $\cdot^*$, we have $\twoheadrightarrow \, = \, \rightsquigarrow^*$.

$\odot$

Remark 2.2.6. The first property is usually proved via an informal 'diagram chasing' or 'tiling' argument. The second property is called the Strip Lemma in the setting of lambda calculi (cf. [Bar84, pp. 279] and [Bar92, pp. 141]). One can find the third property in any text in which confluence of a rewriting system is proved via the so-called Tait & Martin-Löf method.

From the third property we immediately conclude that proving confluence of $\rightarrow$ is equivalent to proving confluence of either one of $\rightarrow^=$, $\rightarrow^+$ or $\twoheadrightarrow$. Combining the three properties, we see that for proving confluence of $\rightsquigarrow$, it suffices to find some intermediate relation $\rightarrow$ such that $\Diamond(\rightarrow)$ and $\rightsquigarrow \, \subseteq \, \rightarrow \, \subseteq \, \rightsquigarrow^*$.

Indeed, this last fact is exploited in the standard way to prove confluence of 'orthogonal' rewriting systems. Intuitively, in orthogonal systems we have that if there are two possibilities of doing a rewrite step starting from some initial object, these steps are 'consistent' with each other. That is, doing one of these steps does not destroy the possibility of doing the other step, but rather leaves a number of 'residuals' of it. Moreover, repeatedly rewriting residuals starting from both objects leads to really the same final object. By really the same, we mean that if a 'position' in the initial object is traced in both ways, it will result in the same set of 'descendant' positions in the final object. The intermediate relation $\rightarrow$ is then chosen such that $a \rightarrow b$ if $a \rightsquigarrow^* b$ by rewriting only residuals of steps in a.

To make this precise, we need at least some way to name or label steps. First, to be able to speak of orthogonal steps. Second, to speak of residuals of a step. Such labelled rewriting systems, are discussed in the next section. Here, we continue with abstract rewriting systems.

For ARSs which are not terminating, the concept of normal form no longer suffices. In an ARS computing an 'infinite' result, the objects in a rewrite are ever better approximations to the result. For such ARSs the rôle of a unique normal form is played by a so-called *cofinal* rewrite. A cofinal rewrite is a rewrite starting from an object, such that the rewrite can be reached from every other reachable object. The concept was introduced by Klop in [Klo80, Def. I.5.13] and he shows that the existence of cofinal rewrites is equivalent to confluence, for countable systems. (An ARS $\rightarrow$ is *countable*, $\mathit{CNT}(\rightarrow)$, if its domain is.) In the next section we employ this characterisation to classify the notion of *decreasing Church-Rosser* (DCR) introduced there. More precisely, we show that $\mathit{CP} \implies \mathit{DCR} \implies \mathit{CR}$.

DEFINITION 2.2.7. Let $\to$ be an ARS on A. A rewrite d is *cofinal* in $\to$ if every object in A can be rewritten to some object in d. The ARS $\to$ on A has the *cofinality* property, $CP(\to)$, if for every a in A, there is a rewrite which is cofinal in $\langle a \twoheadrightarrow \rangle$.

REMARK 2.2.8. We have overloaded the word 'cofinal', by using it both to express that some rewrites end in the same term (Definition 2.1.5) and to express that a rewrite is a kind of infinite normal form. When speaking about more than one rewrite the former concept is meant, and otherwise the latter.

PROPOSITION 2.2.9. *Without loss of generality, we may assume that the cofinal rewrite in Definition 2.2.7 does not contain cycles.*

PROOF We show that a cofinal rewrite d with cycles can be transformed into one without. First, suppose there is an $a \in A$ which occurs infinitely often in d. Then this object is universal in the sense that every other object rewrites to it. Hence, the empty rewrite issuing from a is cofinal and we may assume that all the elements in d occur only finitely often. We construct a cofinal sequence e not containing cycles by the following process.

1. Consider the first element of d. This is the first element of e.
2. Let a be the considered element. Now consider the object following the last occurrence of a in d. This is the next element of e. Repeat this step.

The so constructed sequence e is again cofinal. ⊙

PROPOSITION 2.2.10.

1. *$CP \implies CR$,*
2. *$CR \not\Longrightarrow CP$,*
3. *(WN or CNT) & $CR \implies CP$.*

PROOF

1. [Klo80, Thm. I.5.14],
2. [Klo80, Rem. I.5.15.(i)],
3. The implication CNT & $CR \implies CP$ was already established in [Klo80, Thm. I.5.14], so it remains to prove that for any ARS $\to$ on A, $WN(\to)$ & $CR(\to) \implies CP(\to)$. Let $a \in A$, because $WN(\to)$, a rewrites to some normal form b, which is unique by $CR(\to)$. This rewrite is cofinal in $\langle a \twoheadrightarrow \rangle$, because every object rewrites to b.

⊙

The last statement shows that for confluent systems, the cofinality property is a generalisation of weak normalisation.

Remark 2.2.11. As was noted by K. Mano [Man93], the above 'rewriting' definition of the cofinality property is equivalent to the following 'conversion' definition of it: An ARS $\to$ has the *cofinality* property, $CP(\to)$, if for every a in A, there is a rewrite which is cofinal in $\langle a \leftrightarrow^* \rangle$, where $\langle a \leftrightarrow^* \rangle =^{\text{def}} \to\restriction\{a \leftrightarrow^*\}$.

Because rewrites which are cofinal in $\langle a \leftrightarrow^* \rangle$ are certainly cofinal in $\langle a \twoheadrightarrow \rangle$ the equivalence is trivial in one direction. In the other direction we use the implication $CP \implies CR$ to establish that a cofinal rewrite in $\langle a \twoheadrightarrow \rangle$ is also cofinal in $\langle a \leftrightarrow^* \rangle$.

2.3. Confluence by Decreasing Diagrams

This section is an extended version of [Oos94]. In it we present a confluence theorem that subsumes a number of classical confluence lemmata. A typical way to check confluence is to investigate in which way rewrite steps interact. The idea is that this can be expressed abstractly by labelling rewrite steps with an ordered set of labels. Rewrites then can be labelled with certain multisets of labels, ordered by the standard multiset extension of the label order. A divergence $b \twoheadleftarrow a \twoheadrightarrow c$ is labelled by the multiset sum of the labels of the rewrites $a \twoheadrightarrow b$ and $a \twoheadrightarrow c$. A confluence diagram is labelled by its divergence and is said to be decreasing if the measures of the convergent rewrites $a \twoheadrightarrow b \twoheadrightarrow d$ and $a \twoheadrightarrow c \twoheadrightarrow d$ are both less than or equal to the measure of the diagram.

We define the measure of a rewrite to be the multiset of the lexicographically maximal step labels of the sequence (step labels not less than the label of an earlier step). Decreasing diagrams then can be pasted to yield decreasing diagrams. The main theorem states that if the label order is well-founded and every local confluence diagram is decreasing, then confluence holds.

The first thing we need is to label the set of rewrite steps, i.e. to index the rewrite step relation by a set of labels. By indexing the graph of a relation, we get the standard notion of a directed labelled graph.

Definition 2.3.1. A *labelled rewriting system* (LRS) is a structure $\langle A, I, \longrightarrow \rangle$, where I is a set of *labels* and $\longrightarrow : I \to \mathfrak{P}(A \times A)$, maps labels to sets of pairs of objects. We use $\mathcal{I}$, $\mathcal{J}$, $\mathcal{K}$, $\mathcal{L}$ and $\mathcal{M}$ to range over labelled rewriting systems. Let $\mathcal{I} =^{\text{def}} \langle A, I, \longrightarrow \rangle$. For every $i \in I$, we define the ARS $\to_i =^{\text{def}} \langle A, \longrightarrow_i \rangle$. The *underlying* ARS of $\mathcal{I}$ is $\to_{\mathcal{I}} =^{\text{def}} \bigcup i \in I. \to_i$. Properties of ARSs are extended to LRSs via their underlying ARS. The other way around, every ARS can be lifted to an LRS by choosing a one-element set of labels. *Step-equality* of two LRSs $\mathcal{I}$ and $\mathcal{J}$ is defined via the underlying ARS: $\to_{\mathcal{I}} = \to_{\mathcal{J}}$

Remark 2.3.2. Labelled rewriting systems are the *abstract reduction systems* of [Klo92].

Step-equal LRSs can be viewed as different presentations of the same ARS. In particular, they always have the same abstract properties.

DEFINITION 2.3.3. A *rewrite step* in an LRS $\mathcal{I}$ is a structure $\langle a, i, b \rangle$ such that $a \rightarrow_i b$. In the sequel we will abuse the latter notation to denote the step. The first component is the *start* and the last component is the *end* of the step. The definitions of conversion and rewrite then carry over straightforwardly from the ARS case. If a step is invert, its label is not changed. If $u =^{\mathrm{def}} a \rightarrow_i b$ is a rewrite step, $\mathsf{lab}(u) =^{\mathrm{def}} i$ is the *label* of u. This mapping from rewrite steps to labels is extended to a mapping $\mathsf{lab} : \mathbf{Cnv}(\mathcal{I}) \rightarrow I^*$ from conversions to strings of labels as follows

1. $\mathsf{lab}(0) =^{\mathrm{def}} \varepsilon$,
2. $\mathsf{lab}(ud) =^{\mathrm{def}} \mathsf{lab}(u)\mathsf{lab}(d)$.

Labels can be used to classify the rewrite steps into classes which interact nicely.

DEFINITION 2.3.4. Let $\mathcal{I} =^{\mathrm{def}} \langle A, I, \longrightarrow \rangle$ be a LRS. Let $i \in I$ and $j \in I$.

1. if $\leftarrow_i ; \rightarrow_j \subseteq \rightarrow_j ; \leftarrow_i$, then $\rightarrow_i$ *is independent from* $\rightarrow_j$, $\mathit{IND}(\rightarrow_i, \rightarrow_j)$,
2. if $\leftarrow_i ; \rightarrow_j \subseteq \twoheadrightarrow_j ; \twoheadleftarrow_i$, then i *locally commutes with* j, $\mathit{LCOM}(\rightarrow_i, \rightarrow_j)$,
3. if $\twoheadleftarrow_i ; \twoheadrightarrow_j \subseteq \twoheadrightarrow_j ; \twoheadleftarrow_i$, then i *commutes with* j, $\mathit{COM}(\rightarrow_i, \rightarrow_j)$,

Note that the 'confluence'-notions for abstract rewriting systems are special cases of the 'commutativity'-notions:

$$\begin{array}{rcl} \mathit{IND}(\rightarrow, \rightarrow) & \Longleftrightarrow & \diamond(\rightarrow) \\ \mathit{LCOM}(\rightarrow, \rightarrow) & \Longleftrightarrow & \mathit{LCON}(\rightarrow) \\ \mathit{COM}(\rightarrow, \rightarrow) & \Longleftrightarrow & \mathit{CON}(\rightarrow) \end{array}$$

We have now introduced sufficient terminology to state our main result.

THEOREM 2.3.5. *Let $\mathcal{I} =^{\mathrm{def}} \langle A, I, \longrightarrow \rangle$ be a LRS and let $\succ$ be a well-founded partial order on I. Let I_v and I_h be (not necessarily disjoint) subsets of I, with $\rightarrow_v =^{\mathrm{def}} \bigcup i \in I_v . \rightarrow_i$ and $\rightarrow_h =^{\mathrm{def}} \bigcup i \in I_h . \rightarrow_i$. The v and h stand for vertical and horizontal. If, for all i in I_v and j in I_h, the following diagram holds, then $\rightarrow_v$ commutes with $\rightarrow_h$*

$$
\begin{tikzcd}
{} \arrow[rrr, "j"'] \arrow[ddd, "i"] & & & {} \arrow[d, dashed, two heads, "\curlyvee j"] \\
& & & {} \arrow[d, dashed, "="', "i"] \\
& & & {} \arrow[d, dashed, two heads, "\curlyvee\{i,j\}"] \\
{} \arrow[r, dashed, two heads, "\curlyvee i"'] & {} \arrow[r, dashed, "=", "j"'] & {} \arrow[r, dashed, two heads, "\curlyvee\{i,j\}"'] & {}
\end{tikzcd}
$$

The diagram should be read as follows. For every local divergence $b \leftarrow_i a \rightarrow_j c$, there exists a convergence $b \twoheadrightarrow_\sigma d \twoheadleftarrow_\tau c$, such that the strings σ and τ satisfy the following conditions. The 'horizontal' string σ consists of 'horizontal' labels in I_h. It starts with a number of labels smaller than i ($\twoheadrightarrow_{\curlyvee i}$), followed by at most one j ($\rightarrow_j^=$) and concludes with a number of labels smaller than either i or j ($\twoheadrightarrow_{\curlyvee\{i,j\}}$). The 'vertical' string τ must satisfy analogous conditions. A diagram satisfying these conditions is called locally decreasing.

Proof This will be the topic of Section 2.3.2. $\odot$

Example 2.3.6. The following diagrams are locally decreasing, if we take $2 \succ 1$, but not so if we take $1 \succ 2$

$$
\begin{tikzcd}
a \arrow[rr, "2"'] \arrow[d, "1"] & & c \arrow[d, "1"] \\
b \arrow[r, "2"'] & b' \arrow[r, "1"'] & d
\end{tikzcd}
\qquad
\begin{tikzcd}
a \arrow[rr, "2"'] \arrow[dd, "2"] & & c \arrow[d, "1"] \\
& & c' \arrow[d, "2"] \\
b \arrow[r, "2"'] & b' \arrow[r, "1"'] & d
\end{tikzcd}
$$

The labels in the following diagram, arising from Hindley's counterexample $a \leftarrow b \leftrightarrow c \rightarrow d$ against the implication $LCON \implies CON$, cannot be ordered to make it locally decreasing.

$$
\begin{tikzcd}
b \arrow[rr, "2"'] \arrow[d, "1"] & & a \arrow[d, equal] \\
c \arrow[r, "1"'] & b \arrow[r, "2"'] & a
\end{tikzcd}
$$

In fact, because the example is not confluent, there is no labelling possible making all local confluence diagrams decreasing.

A special case arises when we take the sets I_v and I_h to be equal to the set of all labels I, then confluence of $\rightarrow_{\mathcal{I}}$ can be concluded. Another useful observation is that for proving confluence of an ARS the presentation of an ARS can be chosen freely.

DEFINITION 2.3.7. A labelled rewriting system $\mathcal{I} =^{\mathrm{def}} \langle A, I, \longrightarrow \rangle$ is defined to be *decreasing Church-Rosser* (*DCR*), if there exist a relation $\succ$ and sets I_v and I_h satisfying the assumptions of Theorem 2.3.5, such that $\rightarrow_{\mathcal{I}} = \rightarrow_v = \rightarrow_h$. An ARS is decreasing Church-Rosser if it is the underlying ARS of some decreasing Church-Rosser LRS.

COROLLARY 2.3.8. *A decreasing Church-Rosser ARS is confluent.*

PROOF Let $\rightarrow$ be a decreasing Church-Rosser ARS. By definition there exists a decreasing Church-Rosser LRS $\mathcal{I} =^{\mathrm{def}} \langle A, I, \longrightarrow \rangle$, such that $\rightarrow_{\mathcal{I}} = \rightarrow$. Because the assumptions of Theorem 2.3.5 are satisfied by definition, we conclude $COM(\rightarrow_{\mathcal{I}}, \rightarrow_{\mathcal{I}})$, that is, $CON(\rightarrow)$. $\odot$

Before giving the full proof of this result, we first present many applications in the next subsection.

2.3.1. Many Decreasing Diagrams

Proving a rewriting system confluent by the method of decreasing diagrams, depends upon finding suitable transformations to transform a rewriting system into an equivalent one which is decreasing Church-Rosser. In dealing with every day confluence proofs, it is nice to have a large stock of 'standard transformations'. In this subsection, we present some such transformations. We prove all the 'confluence by local confluence' results for abstract rewriting systems in [Klo92] by the method of decreasing diagrams. Furthermore, we present transformations to deal with some other abstract confluence lemmata we have spotted in the literature.

Most of the time we present a commutativity result. A confluence result is then obtained in the case that the 'horizontal' and 'vertical' rewrite relations are the same.

EXAMPLE 2.3.9. Let $I =^{\mathrm{def}} \{i,j\}$. Let $\mathcal{I} =^{\mathrm{def}} \langle A, I, \longrightarrow \rangle$ be a labelled rewriting system.

1. If we have

$$\begin{tikzcd} a \arrow[r, "j"'] \arrow[d, "i"] & c \arrow[d, dashed, "i"] \\ b \arrow[r, dashed, "j"'] & d \end{tikzcd}$$

then $\mathcal{I}$ is confluent.

Proof Any well-founded ordering on I will do, because the diagram is easily seen to be decreasing (independent from the order). $\odot$

This can be readily extended to an arbitrary (even infinite) set of labels. The resulting lemma is then called the *Lemma of Hindley-Rosen* (cf. [Klo92, Exe. 2.0.8.(2)], [Hin64], [Ros73]). If the relations $\rightarrow_i$ and $\rightarrow_j$ are both equal to $\rightarrow$, we obtain a proof of $\diamond(\rightarrow) \implies CON(\rightarrow)$.

2. If we have

$$\begin{tikzcd}
a \arrow[r, "j"] \arrow[d, "i"'] & c \arrow[d, dashed, "i"] \\
b \arrow[r, dashed, two heads, "j"'] & d
\end{tikzcd}$$

then $\rightarrow_i$ and $\rightarrow_j$ commute.

Proof Order $i \succ j$. $\odot$

If the relations $\rightarrow_i$ and $\rightarrow_j$ are the same, one arrives at the notion of *strong confluence* of [Hue80]. So, every strongly confluent ARS is confluent ([Klo92, Exe. 2.0.8.(10)]).

3. If we have that

$$\begin{tikzcd}
a \arrow[rr, "j"] \arrow[d, "i"'] & & c \arrow[d, dashed, "i"] \\
b \arrow[r, dashed, "j"'] & \cdot \arrow[r, dashed, "i"'] & d
\end{tikzcd}$$

then we say that $\rightarrow_j$ *requests* $\rightarrow_i$ ([Ros73]). If $\diamond(\rightarrow_i)$ and $\rightarrow_j$ requests $\rightarrow_i$, then $\mathcal{I}$ commutes with $\rightarrow_i$.

Proof Order $j \succ i$. $\odot$

If $\diamond(\rightarrow_j)$ then $\mathcal{I}$ is confluent by the same ordering (cf. [Ros73], [Klo92, Exe. 2.0.8.(5)]). Moreover, it follows that instead of $\diamond(\rightarrow_j)$,

$$\begin{tikzcd}
a \arrow[rr, "j"] \arrow[dd, "j"'] & & c \arrow[d, dashed, "j"] \\
 & & \cdot \arrow[d, dashed, "i"] \\
b \arrow[r, dashed, "j"'] & \cdot \arrow[r, dashed, "i"'] & d
\end{tikzcd}$$

suffices for proving confluence of $\mathcal{I}$ ([Sta74], [Klo92, Exe. 2.0.8.(7)]).

In the case that for each local diagram, the rewrites in the conclusion are special cases of the corresponding rewrites in the hypothesis, then the commutativity result can be strengthened by considering the reflexive closures of all the rewrite relations in the conclusion. For example, to prove commutativity in the second example above, it suffices to show that (cf. [Klo92, Exe. 2.0.8.(3)])

$$\begin{tikzcd} a \arrow[r, "j"'] \arrow[d, "i"] & c \arrow[d, dashed, "="', "i"] \\ b \arrow[r, dashed, two heads, "j"'] & d \end{tikzcd}$$

The (three) extra cases to consider are local divergences consisting of at least one empty (reflexive) step. These are dealt with by using the assumption that the rewrites in the original conclusion were special cases of the corresponding rewrites in the assumption, as in the following diagrams.

$$\begin{tikzcd} a \arrow[r, "=", "j"'] \arrow[d, "="', "i"] & a \arrow[d, dashed, "="', "i"] \\ a \arrow[r, dashed, two heads, "j"'] & a \end{tikzcd} \qquad \begin{tikzcd} a \arrow[r, "j"'] \arrow[d, "="', "i"] & c \arrow[d, dashed, "="', "i"] \\ a \arrow[r, dashed, two heads, "j"'] & c \end{tikzcd} \qquad \begin{tikzcd} a \arrow[r, "=", "j"'] \arrow[d, "i"] & a \arrow[d, dashed, "="', "i"] \\ b \arrow[r, dashed, two heads, "j"'] & b \end{tikzcd}$$

To complete the second diagram, one uses that $\rightarrow_j$ is a special case of $\twoheadrightarrow_j$. To complete the third diagram, one uses that $\rightarrow_i$ is a special case of itself.

Another way to obtain commutativity results is to replace some rewrite relations in the LRS by other ones having the same reflexive, transitive closure. For example, if we replace $\rightarrow_i$ by its reflexive, transitive closure in the same example, then to prove commutativity it suffices to show that (cf. [Klo92, Exe. 2.0.8.(4)])

$$\begin{tikzcd} a \arrow[r, "j"'] \arrow[d, two heads, "i"] & c \arrow[d, dashed, two heads, "i"] \\ b \arrow[r, dashed, two heads, "j"'] & d \end{tikzcd}$$

In the examples above one could see immediately just by comparing diagrams how the local confluence diagrams could be made decreasing by choosing an appropriate ordering. In the next cases, one has to apply certain transformations to obtain the desired result.

Example 2.3.10. Let $\mathcal{I} =^{\mathrm{def}} \langle A, I, \rightarrow \rangle$ be a labelled rewriting system, where $I =^{\mathrm{def}} \{i,j\}$.

1. If we have $SN(\mathcal{I})$ and $LCOM(\rightarrow_i, \rightarrow_j)$ then $COM(\rightarrow_i, \rightarrow_j)$.

Proof The idea is to construct a well-founded order on the labels from the strongly normalising rewrite relation on the objects. Take $\mathcal{J} =^{\mathrm{def}} \langle A, J, \rightsquigarrow\rangle$, where $J =^{\mathrm{def}} A \times I$ and $\rightsquigarrow_{(a,k)} =^{\mathrm{def}} \longrightarrow_k \cap\, \mathsf{Stp}(a)$, for each $(a,k) \in J$. So rewrite steps are additionally labelled with the object they start from. Obviously we have $\rightarrow_{\mathcal{I}} = \rightarrow_{\mathcal{J}}$. If we define the order $\succ =^{\mathrm{def}} \mathcal{I}^+ \times_{lex} \varnothing$, then a local commutativity $\mathcal{I}$-diagram gives rise to the following decreasing $\mathcal{J}$-diagram

$$\begin{tikzcd} a \arrow[r, "j"] \arrow[d, "i"'] & c \arrow[d, dashed, two heads, "i"] \\ b \arrow[r, dashed, two heads, "j"'] & d \end{tikzcd} \quad\Longrightarrow\quad \begin{tikzcd} a \arrow[rr, "(a{,}j)"] \arrow[d, "(a{,}i)"'] & & c \arrow[d, dashed, two heads, "\Upsilon(a{,}i)"] \\ b \arrow[rr, dashed, two heads, "\Upsilon(a{,}j)"'] & & d \end{tikzcd}$$

and we are done. Note that the second component of the labels in $\mathcal{J}$ is not used in the ordering. ⊙

The symmetrical version: *SN* & *LCON* $\Longrightarrow$ *CON*, first appeared in [New42], and is therefore known as *Newman's Lemma*. See also [Hue80, Bar84, Klo92] for other proofs of Newman's Lemma.

2. A strengthening of the preceding lemma due to Nipkow appears in [Nip91, Lem. 3.9]. Let $\sqsupset$ be a well-founded strict order on A such that $\rightarrow_i \subseteq \sqsupset$ and $\rightarrow_j \subseteq \sqsupset$. If $LCOM(\rightarrow_i, \rightarrow_j)$ then $COM(\rightarrow_i, \rightarrow_j)$

   Proof We can use the proof of the first item, by changing the order on J into $\succ =^{\mathrm{def}} \sqsupset \times_{lex} >$, where $i > j$. ⊙

3. The following strengthening of Newman's Lemma by Winkler and Buchberger which we cannot deal with by our method appears in [WB83, Lem. 3.1] (cf. [Klo92, Exe. 2.0.8.(12)]). Let $\rightarrow$ be an ARS on A. Let $\sqsupset$ be a well-founded partial order on A such that $\rightarrow\, \subseteq\, \sqsupset$. Then $CON(\rightarrow)$ iff for every local divergence $b \leftarrow a \rightarrow c$ there exists a conversion $b = d_1 \leftrightarrow d_2 \leftrightarrow \ldots \leftrightarrow d_{m-1} \leftrightarrow d_m = c$, such that $\forall n \in \{1, \ldots, m\}.a > d_n$. Since the assumption is weaker than local confluence we *cannot* use the method of decreasing diagrams. Nevertheless, the proof is straightforward.

   Proof One proves by well-founded induction on a ordered by $\sqsupset$, that $CON(\langle a \twoheadrightarrow\rangle)$. ⊙

4. Another variation appears in [CHL92, Lem. 4.5]; a lemma which was coined in [YH89]. Let $COMP(\rightarrow_i)$ and $\diamond(\rightarrow_j)$. If we have

$$\begin{tikzcd} a \arrow[rrr, "j"] \arrow[d, "i"'] & & & c \arrow[d, dashed, two heads, "i"] \\ b \arrow[r, dashed, two heads, "i"'] & \cdot \arrow[r, dashed, "j"'] & \cdot \arrow[r, dashed, two heads, "i"'] & d \end{tikzcd}$$

Then both $\mathcal{I}$ and $\twoheadrightarrow_i ; \rightarrow_j ; \twoheadrightarrow_i$ are confluent.

Proof First, we transfer the well-founded order induced by the terminating relation $\to_i$ to the labels as in the proofs above and observe that ordering j above all those labels makes all the diagrams decreasing. Hence we have proved $CON(\mathcal{I})$. In order to prove $CON(\twoheadrightarrow_i\,;\to_j\,;\twoheadrightarrow_i)$ we need more than confluence of $\mathcal{I}$, because the converging rewrites should contain at least one j-step if they contain i-steps. The easiest way to obtain this is by strengthening the 'diagram-invariant' of Theorem 2.3.5 in the following way. The diagrams constructed should not only be decreasing, but should preserve at least one j-step, that is, in a decreasing diagram

$$\begin{tikzcd} a \arrow[r, twoheadrightarrow, "\tau"] \arrow[d, twoheadrightarrow, "\sigma"] & b \arrow[d, twoheadrightarrow, "\sigma'"] \\ b \arrow[r, twoheadrightarrow, "\tau'"'] & d \end{tikzcd}$$

we must have that if σ (τ) contains a j, then also σ' (τ'). The local diagrams obviously satisfy this property and since it is also preserved by pasting (see the diagram in Lemma 2.3.17) we are done. $\odot$

2.3.2. DCR implies CR

In this subsection we present the proof of Theorem 2.3.5 for deriving confluence from local confluence. We do this by gluing together small 'decreasing' tiles into bigger ones having that same property. The diagrams are decreasing in the sense that their conclusion is less than or equal to their hypothesis. First we define a measure on strings of labels and hence on reduction sequences labelled by them. In this subsection we assume the set of labels I to be strictly partially ordered by $\succ$.

Definition 2.3.11. The (*lexicographic maximum*) *measure* grades strings by finite multisets and is denoted by $|\cdot| : I^* \to \mathbf{FMst}(I)$. Its inductive definition is as follows.

1. $|\varepsilon| =^{\mathrm{def}} \emptyset$,
2. $|i\sigma| =^{\mathrm{def}} [i] \uplus (|\sigma| - \curlyvee i)$.

The measure of a rewrite d is the measure of its label, i.e. $|d| =^{\mathrm{def}} |\mathsf{lab}(d)|$. The measure of a divergence (d,e) is the multiset sum of the measures of its rewrites, i.e. $|(d,e)| =^{\mathrm{def}} |d| \uplus |e|$.

Intuitively, the lexicographic maximum measure takes the multiset of elements which are maximal (in the $\succ$ ordering) with respect to the elements to their left in the string. Operationally, one can think of filtering out the noise before proceeding to the right.

EXAMPLE 2.3.12. Taking the measures of the strings of digits 132343 and 211 yields $|132343| = [1, 3, 3, 4]$ and $|211| = [2]$. Taking the measure of the rewrite $a \to_2 b \to_1 c$ yields $|a \to_2 b \to_1 c| = |21| = [2]$.

The next lemma shows how the measure of a string can be decomposed into the sum of the measures of its substrings.

LEMMA 2.3.13. *1.* $\Upsilon|\sigma| = \Upsilon\sigma$,

2. (de)compose:
$|\sigma ; \tau| = |\sigma| \uplus (|\tau| - \Upsilon\sigma)$,

3. non-emptiness:
$\varepsilon \neq \sigma \implies |\varepsilon| \prec_{mul} |\sigma|$,

4. left-monotonicity:
$|\sigma| \prec_{mul} |\sigma'| \implies |\sigma ; \tau| \prec_{mul} |\sigma' ; \tau|$,
$|\sigma| \preceq_{mul} |\sigma'| \implies |\sigma ; \tau| \preceq_{mul} |\sigma' ; \tau|$,

5. (almost) right-monotonicity:
$|\tau| \prec_{mul} |\tau'| \not\Longrightarrow |\sigma ; \tau| \prec_{mul} |\sigma ; \tau'|$,
$|\tau| \preceq_{mul} |\tau'| \implies |\sigma ; \tau| \preceq_{mul} |\sigma ; \tau'|$.

PROOF 1. The sets of maxima of $|\sigma|$ and $[\sigma]$ are easily shown to be the same. In the sequel we will often make use of this fact without mention.

2. By induction on the length of σ. The base case being trivial, we only show the induction step.

$$\begin{array}{lll}
|i\sigma ; \tau| = & & \\
= & [i] \uplus (|\sigma ; \tau| - \Upsilon i) & \text{Definition} \\
= & [i] \uplus ((|\sigma| \uplus (|\tau| - \Upsilon\sigma)) - \Upsilon i) & \text{Induction Hypothesis} \\
= & [i] \uplus (|\sigma| - \Upsilon i) \uplus ((|\tau| - \Upsilon\sigma) - \Upsilon i) & \text{Distribute 1.4.6(8d)} \\
= & |i\sigma| \uplus ((|\tau| - \Upsilon\sigma) - \Upsilon i) & \text{Definition} \\
= & |i\sigma| \uplus (|\tau| - \Upsilon i\sigma) & \text{1.4.6(7c) and additivity}
\end{array}$$

3. Trivial.

4.
$$\begin{array}{lll}
|\sigma ; \tau| = & & \\
= & |\sigma| \uplus (|\tau| - \Upsilon\sigma) & (2) \\
= & |\sigma| \uplus ((|\tau| - \Upsilon\sigma) \cap \Upsilon\sigma') \uplus & \\
 & \quad ((|\tau| - \Upsilon\sigma) - \Upsilon\sigma') & \text{Split 1.4.6(7d)} \\
= & |\sigma| \uplus ((|\tau| - \Upsilon\sigma) \cap \Upsilon\sigma') \uplus (|\tau| - \Upsilon\sigma') & \\
= & |\sigma| \uplus ((|\tau| \cap \Upsilon\sigma') - \Upsilon\sigma) \uplus (|\tau| - \Upsilon\sigma') & \text{Exchange 1.4.6(8c)} \\
\subseteq & |\sigma| \uplus (\Upsilon\sigma' - \Upsilon\sigma) \uplus (|\tau| - \Upsilon\sigma') & \\
\prec_{mul} & |\sigma'| \uplus (|\tau| - \Upsilon\sigma') & \text{Interpolate 1.4.11(7)} \\
= & |\sigma' ; \tau| & (2)
\end{array}$$

The second implication is obtained by replacing $\prec_{mul}$ by $\preceq_{mul}$ in the proof.

5. A counterexample is provided by $|0| \succ_{mul} |\varepsilon| \not\Longrightarrow |10| \succ_{mul} |1|$, if $1 \succ 0$. The second implication is trivial from (2) and (left) monotonicity of $\uplus$ and $-$.

$\odot$

REMARK 2.3.14. Combining the last part of the lemma with the assumption that $\succ$ is a strict order, we find that $|\cdot| \succ_{mul} |\cdot|$ is almost a division ordering on finite strings (see e.g. [Mar90]), but not quite. Hence, we can not conclude its well-foundedness directly. Nevertheless, well-foundedness follows by well-foundedness of $\succ_{mul}$ on finite multisets (Proposition 1.4.9).

The lexicographic maximum measure is designed to make pasting decreasingness preserving (Lemma 2.3.17) and hypothesis decreasing (Lemma 2.3.19). The intuition for this measure is that labels below a label in front of them do not matter in proving confluence. At the time we get to them to finish the confluence diagram we already know they 'behave nicely' because the bigger label does so. We now define decreasing diagrams with respect to this measure.

DEFINITION 2.3.15. The diagram

$$\begin{tikzcd} \cdot \arrow[r, twoheadrightarrow, "\tau"] \arrow[d, twoheadrightarrow, "\sigma"'] \arrow[dr, phantom, "DCR"] & \cdot \arrow[d, twoheadrightarrow, "\sigma'"] \\ \cdot \arrow[r, twoheadrightarrow, "\tau'"'] & \cdot \end{tikzcd}$$

is *decreasing* (*DCR*) if the following decreasingness condition is satisfied:

$$|\sigma\,;\tau'| \preceq_{mul} |\tau| \uplus |\sigma| \succeq_{mul} |\tau\,;\sigma'|$$

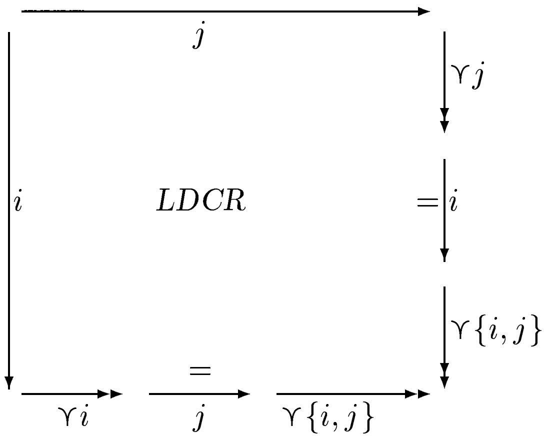
A diagram is *locally decreasing* (*LDCR*) if it is local and decreasing.

Using the Decomposition Lemma 2.3.13(2) and the property in Proposition 1.4.11(6) we can reformulate the decreasingness condition as

$$|\tau'| - \curlyvee\sigma \preceq_{mul} |\tau| \ \&\ |\sigma| \succeq_{mul} |\sigma'| - \curlyvee\tau$$

One can think of these inequalities in the following way. The labels in the measure of the conclusion (σ') all trace back to the labels (in the measure) of the opposite side in the hypothesis (σ), except for the noise (elements of $\curlyvee\tau$) which has been generated by the adjacent side in the hypothesis (τ). The next proposition gives a characterisation of the shape of a locally decreasing diagram.

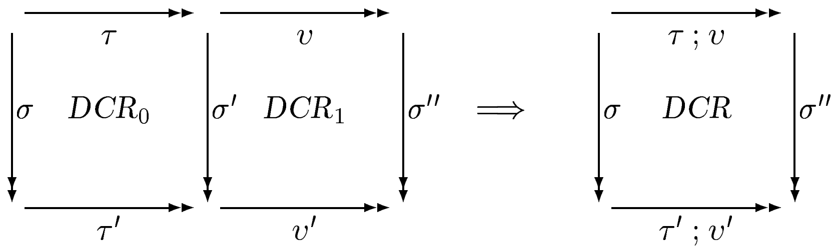
PROPOSITION 2.3.16. *The form of a locally decreasing diagram is*



PROOF The reformulation of decreasingness in the case of a local diagram where $\sigma = i$ and $\tau = j$, reads $|\tau'| - \Upsilon i \preceq_{mul} [j]$ & $[i] \succeq_{mul} |\sigma'| - \Upsilon j$. It is apparent that σ' is a string less than or equal to i, i.e. a string headed by at most one i followed by a number of labels less than i, interspersed with noise from j, i.e. labels less than j. This is described exactly by the right-hand side of the *LDCR*-diagram. The same holds, mutatis mutandis, for τ'. $\odot$

A nice property of *DCR*-diagrams is that they can be pasted together to form bigger *DCR*-diagrams.

LEMMA 2.3.17.



PROOF We have to prove that the diagram on the right is decreasing. Continuing on the informal explanation of decreasingness, the proofs are guided by tracing back the reductions in the conclusion to the reductions in the hypothesis. For the right-hand side of the conclusion, the labels of σ'' are either noise from υ or trace back to σ'. The labels of σ' are either noise from τ or trace back to σ. We can combine these observations by noting that all the noise generated by τ and υ can also be generated by $\tau ; \upsilon$, and that tracing back is transitive. Formally

$$
\begin{array}{lll}
\multicolumn{3}{l}{|\tau ; \upsilon ; \sigma''| =} \\
= & |\tau ; \upsilon| \uplus (|\sigma''| - \Upsilon \tau ; \upsilon) & \text{Decompose (Lemma 2.3.13(2))} \\
= & |\tau ; \upsilon| \uplus ((|\sigma''| - \Upsilon \upsilon) - \Upsilon \tau) & \text{1.4.6(7c) and additivity} \\
\preceq_{mul} & |\tau ; \upsilon| \uplus (|\sigma'| - \Upsilon \tau) & DCR_1 \\
\preceq_{mul} & |\tau ; \upsilon| \uplus |\sigma| & DCR_0
\end{array}
$$

Observe the close relationship between the informal and formal proof. For the other side of the conclusion the situation is more complicated. For the first half of the conclusion (τ') everything is straightforward. However, for the second half of the conclusion (υ'), the noise generated by σ' either traces back to σ or it is noise generated by τ. The former case is not problematic, because σ is allowed to generate noise inside $\tau' ; \upsilon'$. The latter case is problematic, because it is not clear why steps in υ' should trace back to τ. We are saved by the lexicographic maximum measure because, roughly speaking, some of the steps in υ' are filtered out by τ' in taking the measure of $\tau' ; \upsilon'$ and the other ones can safely be traced back to τ (safely, because they were *not* filtered out).

$$
\begin{array}{lll}
\multicolumn{3}{l}{|\sigma ; \tau' ; \upsilon'| =} \\
= & |\sigma ; \tau'| \uplus (|\upsilon'| - \Upsilon\sigma ; \tau') & \text{Decompose 2.3.13(2)} \\
= & |\sigma ; \tau'| \uplus ((|\upsilon'| - \Upsilon\sigma ; \tau') \cap \Upsilon\tau) \uplus & \\
 & \qquad ((|\upsilon'| - \Upsilon\sigma ; \tau') - \Upsilon\tau) & \text{Split 1.4.6(7d)} \\
\preceq_{mul} & |\sigma| \uplus |\tau| \uplus ((|\upsilon'| - \Upsilon\sigma ; \tau') - \Upsilon\tau) & \text{see below} \\
\subseteq & |\sigma| \uplus |\tau| \uplus ((|\upsilon'| - \Upsilon\sigma') - \Upsilon\tau) & \text{see below} \\
\preceq_{mul} & |\sigma| \uplus |\tau| \uplus (|\upsilon| - \Upsilon\tau) & DCR_1 \\
= & |\sigma| \uplus |\tau ; \upsilon| & \text{Compose 2.3.13(2)}
\end{array}
$$

The correspondence between the informal proof and the formal one is here more difficult to find, but still present. For example the first not yet justified step corresponds exactly to the problematic case above.

CLAIM If we take $M =^{\text{def}} |\sigma;\tau'|$, $N =^{\text{def}} |\sigma|\uplus|\tau|$, and $X =^{\text{def}} (|\upsilon'|-\Upsilon\sigma;\tau')\cap\Upsilon\tau$, then the assumptions of Proposition 1.4.11(7) are satisfied. Hence the step is justified.

PROOF OF CLAIM The first assumption, $M \preceq_{mul} N$, follows directly from DCR_0. The second one, $X \subseteq \Upsilon N - \Upsilon M$, is shown by the following simple argument.

$$
\begin{array}{lll}
\multicolumn{3}{l}{(|\upsilon'| - \Upsilon\sigma ; \tau') \cap \Upsilon\tau =} \\
= & (|\upsilon'| \cap \Upsilon\tau) - \Upsilon\sigma ; \tau' & \text{Exchange 1.4.6(8c)} \\
\subseteq & \Upsilon\tau - \Upsilon\sigma ; \tau' & \\
\subseteq & (\Upsilon\sigma \uplus \Upsilon\tau) - \Upsilon\sigma ; \tau' & \\
= & \Upsilon(|\sigma| \uplus |\tau|) - \Upsilon|\sigma ; \tau'| & \text{additivity and Lemma 2.3.13(1)}
\end{array}
$$

$\odot$

This proves the claim. The other step reformulates which steps in υ' need to trace back to υ. This is just as simple.

$$
\begin{array}{lll}
\multicolumn{3}{l}{(|\upsilon'| - \Upsilon\sigma ; \tau') - \Upsilon\tau =} \\
= & |\upsilon'| - \Upsilon\sigma ; \tau' ; \tau & \text{1.4.6(7c)} \\
\subseteq & |\upsilon'| - \Upsilon\sigma ; \tau & \\
\subseteq & |\upsilon'| - \Upsilon(|\sigma'| \uplus |\tau|) & DCR_0 \\
= & (|\upsilon'| - \Upsilon\sigma') - \Upsilon\tau & \text{1.4.6(7c)}
\end{array}
$$

$\odot$

REMARK 2.3.18. Note that 'cutting' does not preserve decreasingness. That is, if we have a decreasing diagram composed of two diagrams, one of which is decreasing, the other one is not necessarily decreasing. As exemplified by the following diagrams

$$
\begin{tikzcd}
\cdot \arrow[r, "0"] \arrow[d, "0"'] & \cdot \arrow[r, "0"] \arrow[d, "1"'] \arrow[dr, phantom, "DCR"] & \cdot \arrow[d, "0"] \\
\cdot \arrow[r, "0"'] & \cdot \arrow[r, "0"'] & \cdot
\end{tikzcd}
\qquad
\begin{tikzcd}
\cdot \arrow[r, "0"] \arrow[d, "1"'] \arrow[dr, phantom, "DCR"] & \cdot \arrow[r, "0"] \arrow[d, "0"'] & \cdot \arrow[d, "1"] \\
\cdot \arrow[r, "0"'] & \cdot \arrow[r, "0"'] & \cdot
\end{tikzcd}
$$

where 1 is greater than 0.

We will prove the main theorem by well-founded induction on the measure of a divergence. The next lemma states that by filling in a decreasing diagram, the measure is decreased.

LEMMA 2.3.19. *If τ is non-empty and we have the following situation*

$$
\begin{tikzcd}
\cdot \arrow[r, two heads, "\tau"] \arrow[d, two heads, "\sigma"'] \arrow[dr, phantom, "DCR"] & \cdot \arrow[r, two heads, "\upsilon"] \arrow[d, two heads, "\sigma'"] & \cdot \\
\cdot \arrow[r, two heads, "\tau'"'] & \cdot &
\end{tikzcd}
$$

then $|\sigma'| \uplus |\upsilon| \prec_{mul} |\sigma| \uplus |\tau ; \upsilon|$, i.e. the measure of the hypothesis is decreased.

PROOF What labels are in the measure of the new hypothesis? A label in the measure σ' either traces back to σ or is noise generated by τ. The labels in the measure of υ either were also present in the measure of $\tau ; \upsilon$ or were filtered out by τ. In the last case they can be considered noise generated by τ. Summing up, the only 'created' step labels in the new hypothesis can be seen as noise generated by τ.

$$
\begin{array}{lll}
\multicolumn{3}{l}{|\sigma'| \uplus |\upsilon| =} \\
= & ((|\sigma'| \uplus |\upsilon|) \cap \curlyvee\tau) \uplus & \\
 & ((|\sigma'| \uplus |\upsilon|) - \curlyvee\tau) & \text{Split 1.4.6(7d)} \\
\prec_{mul} & |\tau| \uplus ((|\sigma'| \uplus |\upsilon|) - \curlyvee\tau) & \text{Noise Reduction 1.4.11(4)} \\
= & |\tau| \uplus (|\sigma'| - \curlyvee\tau) \uplus (|\upsilon| - \curlyvee\tau) & \text{Distribute 1.4.6(8d)} \\
= & |\tau ; \sigma'| \uplus (|\upsilon| - \curlyvee\tau) & \text{Compose 2.3.13(2)} \\
\preceq_{mul} & |\sigma| \uplus |\tau| \uplus (|\upsilon| - \curlyvee\tau) & DCR \\
= & |\sigma| \uplus |\tau ; \upsilon| & \text{Compose 2.3.13(2))}
\end{array}
\qquad \odot
$$

Now the two preceding lemmas can be used in a straightforward way to obtain the theorem. It is only here that we have to assume that the strict partial order $\succ$ is well-founded.

THEOREM 2.3.20. *Let $\mathcal{I} =^{\text{def}} \langle A, I, \longrightarrow\rangle$ be a LRS and let $\succ$ be a well-founded partial order on I. Let I_v and I_h be (not necessarily disjoint) subsets of I, with $\rightarrow_v =^{\text{def}} \bigcup i \in I_v . \rightarrow_i$ and $\rightarrow_h =^{\text{def}} \bigcup i \in I_h . \rightarrow_i$. The v and h stand for vertical and horizontal. If, for all i in I_v and j in I_h, the following diagram holds, then $\rightarrow_v$ commutes with $\rightarrow_h$*

$$\begin{tikzcd}
\cdot \arrow[r, "j"] \arrow[d, "i"'] \arrow[dr, phantom, "LDCR"] & \cdot \arrow[d, dashed, two heads, "\sigma"] \\
\cdot \arrow[r, dashed, two heads, "\tau"'] & \cdot
\end{tikzcd}$$

In the diagram σ must consist of vertical labels ($\in I_v$) and τ of horizontal ones ($\in I_h$).

PROOF The theorem is proved by well-founded induction on the measure of diagrams, showing that we always obtain *DCR*-diagrams. The proof is expressed by the following diagram.

$$\begin{tikzcd}
\cdot \arrow[r, "j"] \arrow[d, "i"'] \arrow[dr, phantom, "LDCR"] & \cdot \arrow[r, two heads] \arrow[d, dashed, two heads, "\sigma"] \arrow[dr, phantom, "IH_0"] & \cdot \arrow[d, dashed, two heads] \\
\cdot \arrow[r, dashed, two heads, "\tau"'] \arrow[d, two heads] \arrow[drr, phantom, "IH_1"] & \cdot \arrow[r, dashed, two heads] & \cdot \arrow[d, dashed, two heads] \\
\cdot \arrow[rr, dashed, two heads] & & \cdot
\end{tikzcd}$$

The diagram IH_0 can be completed to a *DCR*-diagram because of the induction hypothesis, using Lemma 2.3.19. The diagrams *LDCR* and IH_0 together form a *DCR*-diagram by Lemma 2.3.17, hence the diagram IH_1 can be completed again to a *DCR*-diagram because of the induction hypothesis, using (the mirrored version of) Lemma 2.3.19. Now the complete diagram is a *DCR*-diagram by another appeal to (the mirrored version of) Lemma 2.3.17. $\odot$

The above proof can be made more formal, by showing that the predicate P on divergences, defined by: $P(d,e) =^{\text{def}} \exists d'.\exists e'.DCR(d,e,d',e')$, is a $|\cdot| \succ_{mul} |\cdot|$-complete predicate (see [Hue80]).

2.3.3. More Decreasing Diagrams

In this subsection we present an asymmetrical version of Theorem 2.3.5 and state examples of its use. The notion of strong confluence we introduce is a generalisation of Huet's notion ([Hue80]).

DEFINITION 2.3.21. Let $\mathcal{I} =^{\text{def}} \langle A, I, \rightarrow \rangle$ be a LRS. The LRS is *strongly confluent* if there exists a well-founded partial order $\succ$ on I such that for all $i \in I$, $j \in J$ the following *local strong confluence* diagram holds

$$
\begin{tikzcd}
\cdot \arrow[rrr, "j"'] \arrow[ddd, "i"'] & & & \cdot \arrow[d, dashed, twoheadrightarrow, "\curlyvee\!\mid j"] \\
& & & \cdot \arrow[d, dashed, "="', "i"] \\
& & & \cdot \arrow[d, dashed, twoheadrightarrow, "\curlyvee i \cup \curlyvee\!\mid j"] \\
\cdot \arrow[r, dashed, twoheadrightarrow, "\curlyvee i"'] & \cdot \arrow[r, dashed, "=", "j"'] & \cdot \arrow[r, dashed, twoheadrightarrow, "\curlyvee\{i,j\}"'] & \cdot
\end{tikzcd}
$$

Call an ARS strongly confluent if it underlies a strongly confluent LRS. The difference between strongly confluent and decreasing Church-Rosser is that in the case of strong confluence the 'vertical' conclusion may have labels which are equal ($\curlyvee\!\mid j$) to the label (j) of the horizontal assumption, which is not allowed in the case of a decreasing diagram, because there the 'vertical' labels must be in $\curlyvee j$.

THEOREM 2.3.22. *A strongly confluent ARS is confluent.*

PROOF Let the ARS underly the strongly confluent LRS $\mathcal{I} =^{\text{def}} \langle A, I, \rightarrow \rangle$. Let $\succ$ be the well-founded partial order on I making the LRS strongly confluent. We define another LRS $\mathcal{J} =^{\text{def}} \langle A, I \times \{h, v\}, \rightsquigarrow \rangle$, by creating for every rewrite relation $\rightarrow_i$ two copies, a 'horizontal' and a 'vertical' one: $\rightsquigarrow_{(i,h)} =^{\text{def}} \rightsquigarrow_{(i,v)} =^{\text{def}} \rightarrow_i$. If we take $\sqsupset =^{\text{def}} \succ \times_{lex} \succ'$, where $h \succ' v$, then $\sqsupset$ is a well-founded relation on $I \times \{h, v\}$. If we take $I_h =^{\text{def}} I \times \{h\}$ and $I_v =^{\text{def}} I \times \{v\}$ in Theorem 2.3.5, then $\rightsquigarrow_h$ and $\rightsquigarrow_v$ commute, because we can construct the following decreasing diagram by the strong confluence assumption and by choosing appropriately labelled (horizontal/vertical) versions of the rewrite relations.

$$
\begin{tikzcd}
\cdot \arrow[rrr, "{(j,h)}"] \arrow[ddd, "{(i,v)}"] & & & \cdot \arrow[d, dashed, two heads, "{\curlyvee(j,h)}"] \\
& & & \cdot \arrow[d, dashed, "{(i,v)}", "="'] \\
& & & \cdot \arrow[d, dashed, two heads, "{\curlyvee\{(i,v),(j,h)\}}"] \\
\cdot \arrow[r, dashed, two heads, "{\curlyvee(i,v)}"'] & \cdot \arrow[r, dashed, "{(j,h)}"', "="] & \cdot \arrow[r, dashed, two heads, "{\curlyvee\{(i,v),(j,h)\}}"'] & \cdot
\end{tikzcd}
$$

In the labelling of the steps, we use that $\curlyvee\!\mid(j,h) = \curlyvee(j,h) = \curlyvee\!\mid(j,v)$ when restricted to vertical labels. Since $\rightsquigarrow_h = \rightarrow = \rightsquigarrow_v$ (they are just copies), $\rightarrow$ commutes with itself, as wanted. $\odot$

We next state two corollaries of this theorem. The first one is a lemma by De Bruijn [Bru78]. It was the search for a simple proof of this lemma, instead of the complicated combinatorial proof given in his paper, which led to our notion of decreasing diagram.

COROLLARY 2.3.23. *Let $\mathcal{I} =^{\mathrm{def}} \langle A, I, \longrightarrow\rangle$ be an LRS. If there exists a well-founded total order $\succ$ on I, such that for every $i \in I$ and $j \in J$, such that $j \succ i$, we have the following local confluence diagrams, then $\mathcal{I}$ is confluent.*

$$
\begin{tikzcd}
\cdot \arrow[rrr, two heads, "j"] \arrow[d, "j"] & & & \cdot \arrow[d, dashed, two heads, "\curlyvee\!\mid j"] \\
\cdot \arrow[r, dashed, two heads, "\curlyvee j"'] & \cdot \arrow[r, dashed, "j"', "="] & \cdot \arrow[r, dashed, two heads, "\curlyvee j"'] & \cdot
\end{tikzcd}
$$

$$
\begin{tikzcd}
\cdot \arrow[rrr, two heads, "j"] \arrow[d, "i"] & & & \cdot \arrow[d, dashed, two heads, "\curlyvee j"] \\
\cdot \arrow[r, dashed, two heads, "\curlyvee i"'] & \cdot \arrow[r, dashed, "j"', "="] & \cdot \arrow[r, dashed, two heads, "\curlyvee j"'] & \cdot
\end{tikzcd}
$$

PROOF A well-founded total order is of course a well-founded partial order. One easily checks that both diagrams and the mirrored version of the second diagram are all strong confluence diagrams, so we can apply Theorem 2.3.22 and obtain confluence. Note that because of the totality of the order $\succ$, these three cases cover all the possible local divergences (we cannot have that two labels are incomparable). The mirrored version of the second diagram is needed because of the condition $j \succ i$ in its hypothesis. $\odot$

Next, we show a non-trivial application of the strong confluence theorem to obtain a result by Geser ([Ges90, p. 77],[Klo92, Exe. 2.0.8.(19)]).

Definition 2.3.24. Let $\mathcal{I} =^{\mathrm{def}} \langle A, \{i,j\}, \rightarrow\rangle$ be an LRS.

1. The relation $\rightarrow_i$ *modulo* $\rightarrow_j$ on A, $\rightarrow_{i/j}$, is defined to be $\twoheadrightarrow_j \,;\, \rightarrow_i \,;\, \twoheadrightarrow_j$. If $\rightarrow_{i/j}$ is terminating, then $\rightarrow_i$ is said to be *relatively terminating* (with respect to $\rightarrow_j$).
2. The relation $\rightarrow_j$, is *non-splitting*, if $\leftarrow_j \,;\, \rightarrow_{\mathcal{I}} \subseteq \rightarrow_{\mathcal{I}}^{=} \,;\, \twoheadleftarrow_{\mathcal{I}}$.

Corollary 2.3.25. *Let $\mathcal{I} =^{\mathrm{def}} \langle A, \{i,j\}, \rightarrow\rangle$ be an LRS. If $\rightarrow_i$ is relatively terminating and locally confluent and if $\rightarrow_j$ is non-splitting, then $\mathcal{I}$ is confluent.*

Proof The idea of proof is the same as for the proof of Newman's lemma. The rewrite relations are split into smaller ones based on the 'weight' (with respect to $\rightarrow_{i/j}$) of the origin of a rewrite step. Let the LRS $\mathcal{J} =^{\mathrm{def}} \langle A, A, \rightsquigarrow\rangle$ be defined by $a \rightsquigarrow_c b$ if $c \twoheadrightarrow_j a \rightarrow_{\mathcal{I}} b$, for a, b and c in A. Note that we can translate a rewrite step in $\mathcal{I}$ to many different rewrite steps in $\mathcal{J}$, but nevertheless $\mathcal{I}$ and $\mathcal{J}$ are clearly step equivalent. The order $\succ =^{\mathrm{def}} \rightarrow_{i/j}^{+}$ is a well-founded partial order on A (the set of labels of $\mathcal{J}$) because $\rightarrow_i$ is relatively terminating by assumption. To obtain confluence of $\mathcal{I}$ it suffices to check that $\mathcal{J}$ is strongly confluent. So suppose we have a local divergence $b \leftarrow_e a \rightarrow_f c$ in $\mathcal{J}$. Then by definition there exist rewrites $e \twoheadrightarrow_j a \rightarrow_{\mathcal{I}} b$ and $f \twoheadrightarrow_j a \rightarrow_{\mathcal{I}} c$ in $\mathcal{I}$. Because in both of these rewrites the $\rightarrow_{\mathcal{I}}$-step can be either an $\rightarrow_i$- or a $\rightarrow_j$-step, there are four cases to consider.

If both are $\rightarrow_i$-steps, then we can construct the following diagrams.

$$\begin{tikzcd}
a \arrow[r, "i"'] \arrow[d, "i"] & c \arrow[d, dashed, two heads, "i"] \\
b \arrow[r, dashed, two heads, "i"'] & d
\end{tikzcd}
\quad\Longrightarrow\quad
\begin{tikzcd}
a \arrow[r, "f"'] \arrow[d, "e"] & c \arrow[d, dashed, two heads, "\sigma"] \\
b \arrow[r, dashed, two heads, "\tau"'] & d
\end{tikzcd}$$

The conclusion of the $\mathcal{I}$-diagram on the left is obtained by local confluence of $\rightarrow_i$. The conclusion of the $\mathcal{J}$-diagram on the right is obtained from the one on the left by translating each step $a' \rightarrow_{\mathcal{I}} b'$ in the conclusion to $a' \rightarrow_{a'} b'$. By choice of the order $\succ$, each of the labels in σ and τ are then below f and e, respectively. So the diagram is a strong confluence diagram.

If one of the steps is an $\rightarrow_i$-step and the other one an $\rightarrow_j$-step, we can construct the following diagrams.

$$\begin{tikzcd}
a \arrow[r, "i"'] \arrow[d, "j"] & c \arrow[d, dashed, two heads, "\mathcal{I}"] \\
b \arrow[r, dashed, "=", "\mathcal{I}"'] & d
\end{tikzcd}
\quad\Longrightarrow\quad
\begin{tikzcd}
a \arrow[r, "f"'] \arrow[d, "e"] & c \arrow[d, dashed, two heads, "\sigma"] \\
b \arrow[r, dashed, "=", "f"'] & d
\end{tikzcd}$$

$$\begin{tikzcd}
a \arrow[r, "j"'] \arrow[d, "i"] & c \arrow[d, dashed, "="', "\mathcal{I}"] \\
b \arrow[r, dashed, two heads, "\mathcal{I}"'] & d
\end{tikzcd}
\quad\Longrightarrow\quad
\begin{tikzcd}
a \arrow[r, "f"'] \arrow[d, "e"] & c \arrow[d, dashed, "="', "e"] \\
b \arrow[r, dashed, two heads, "\tau"'] & d
\end{tikzcd}$$

The diagrams on the left can be constructed because $\to_j$ is non-splitting. The diagrams on the right can be constructed analogously to the previous case.

Finally, if both steps are $\to_j$-steps, then we can construct the following diagrams.

$$\begin{tikzcd} a \arrow[r, "j"'] \arrow[dd, "j"] & c \arrow[dd, dashed, "\mathcal{I}"] \\ & \\ b \arrow[r, dashed, "\mathcal{I}"'] \arrow[ur, phantom, "="] & d \end{tikzcd} \quad \Longrightarrow \quad \begin{tikzcd} a \arrow[r, "f"'] \arrow[dd, "e"] & c \arrow[d, dashed, "f"] \\ & c' \arrow[d, dashed, "\sigma"] \\ b \arrow[r, dashed, "f"'] \arrow[ur, phantom, "="] & d \end{tikzcd}$$

The diagram on the left can be constructed again since $\to_j$ is non-splitting. The construction of the diagram on the right is somewhat problematic. The labels in the right-hand side of the conclusion cannot be chosen below f until the first (if any) $\to_i$-step occurs. Happily, we can choose them to be equal to f and still obtain a strong confluence diagram. $\odot$

Remark 2.3.26. The proof shows that local confluence of $\to_i$ is an unnecessarily restrictive condition. In the corresponding diagram, the i-labels in the conclusion can be replaced by $\mathcal{I}$ without affecting the proof.

2.3.4. Is DCR equivalent to CR?

In the preceding subsections we have proven many ARSs confluent by proving them to be decreasing Church-Rosser. An interesting question is whether this holds in general, that is, whether $CR \iff DCR$ holds ([DJK93, Problem 56]). A negative as well as a positive answer to this question would be interesting. A positive one because it would show that proving confluence can always be reduced to proving a form of local confluence. A drawback is that one can claim that our main Theorem 2.3.20 then merely is a 'shift in viewpoint'. But indeed a useful one! A negative answer could also be interesting, if it would provide an effective classification of confluent ARSs. Alas, we have only been able to come up with a partial result.

Definition 2.3.27. Let $\to$ be an ARS on A. Let $a \in A$, $b \in A$. Define the *rewrite-distance* from a to b by $\mathsf{rd}(a,b) =^{\mathrm{def}} \boldsymbol{\mu} m \in \mathbb{N}.\, a \to^m b$. The rewrite-distance is extended to sets of objects, by taking the minimum over all the objects occurring in them.

The rewrite-distance measures how many rewrite steps are needed to reach one object from another. In the proofs of the following rewrite steps are labelled by the rewrite-distance to the 'normal form'.

Proposition 2.3.28. *CP* $\Longrightarrow$ *DCR*.

PROOF Let $\rightarrow =^{\text{def}} \langle A, \longrightarrow \rangle$ be an ARS having the cofinality property in the formulation of Remark 2.2.11. It is clear that an arbitrary union of (object-) disjoint ARSs which are *DCR* is again *DCR*, so we may assume without loss of generality that $\rightarrow = \langle a \leftrightarrow^* \rangle$ for an arbitrary $a \in A$. Let d be the rewrite which is cofinal in $\langle A, \longrightarrow \rangle$. Define an LRS $\mathcal{I} =^{\text{def}} \langle A, \mathbb{N}, \longrightarrow \rangle$ as follows. Let $b \rightarrow c$, then

1. $b \rightarrow_0 c$, if $\exists m \in \mathbb{N}. d(m) = b \rightarrow c$.
2. $b \rightarrow_{m+1} c$, where $m =^{\text{def}} \mathsf{rd}(c,d)$, otherwise.

CLAIM If we take the natural order $>$ on $\mathbb{N}$, then every local divergence in $\mathcal{I}$ can be completed to form a decreasing diagram.

PROOF OF CLAIM Suppose $b \leftarrow_m d \rightarrow_{m'} c$ is a local divergence in $\mathcal{I}$. We consider two cases:

1. If m and m' are both 0, then both steps are part of the cofinal rewrite d. By Proposition 2.2.9 we may assume that this rewrite does not contain cycles, so we must have $b = c$.
2. Suppose one of m and m' is not equal to 0. Consider b. By cofinality of d and by definition of rewrite-distance, we can find a rewrite $b_{m \dot{-} 1} \rightarrow_{m \dot{-} 1} \ldots \rightarrow_1 b_0$, where $b_{m \dot{-} 1} =^{\text{def}} b$, such that b_0 is on d. All the labels in this rewrite are smaller than m. The same can be done for c, giving a rewrite ending in c_0. Now to join b_0 and c_0, only steps on d need to be taken, which all have labels 0. The resulting local confluence diagram is easily seen to be decreasing.

In both cases we have constructed a decreasing diagram. $\odot$

$\odot$

REMARK 2.3.29. Independently, K. Mano, proved the same result [Man93]. He also pointed out that *DCR* does not imply *CP* by the very example J. W. Klop used to show that *CR* does not imply *CP* ([Klo80, Rem. I.5.15.(i)]).

COROLLARY 2.3.30. (*WN* or *CNT*) & *CR* $\iff$ *DCR*.

PROOF $\implies$ By Proposition 2.2.10 and Proposition 2.3.28.

$\impliedby$ By Theorem 2.3.20.

$\odot$

Summing up the relationship between *CR*, *DCR* and *CP*, we have *CP* $\implies$ *DCR* $\implies$ *CR*, but neither of *DCR* and *CR* implies *CP*. The question remains whether *CR* implies *DCR*.

CONJECTURE 2.3.31. *CR does not imply DCR.*

It is not all that bad, that we can prove equivalence only for countable systems, since the rewriting systems in practice are countable. For example, the Higher-Order Rewriting Systems of the next chapter are countable, so *DCR* and *CR* coincide for them.

2.4. Confluence by Orthogonality

The result of the preceding section is based on an analysis of the interaction between rewrite steps. In particular it shows that one has confluence in the case that the duplication of a label in a local confluence diagram is really a splitting of the label. That is, a label in the hypothesis, may have several 'children' in the conclusion, but then the weight of these must be smaller than the weight of the original one. So, if we want to prove confluence for a particular system, we devise the labels such that this condition is satisfied. We put the ordering on the labels, so to speak.

Another approach is to view an object as a combination of resources. The rewrite steps then consume certain resources and possibly produce new ones, while leaving other resources unchanged. This can be captured by introducing a 'tracing' or 'tracking' function which keeps track of what happens to resources along rewrite steps. We formalise these intuitive notions in the following way.

Definition 2.4.1. A *descendant rewriting system* (DRS for short) is a quintuple $\langle A, I, \rightarrow, \mathsf{Pos}, \lfloor \cdot \rfloor \rangle$, such that $\langle A, I, \rightarrow \rangle$ is an LRS (its underlying LRS), $\mathsf{Pos} : A \to \mathfrak{P}(\mathsf{Pos})$ is a function mapping objects to sets of *positions* (*properties*), and $\lfloor \cdot \rfloor$ is the *trace* function mapping each rewrite step $a \to_i b$ to a *descendant* relation $\lfloor a \to_i b \rfloor$ from the set $\mathsf{Pos}(a)$ of a-positions to the set $\mathsf{Pos}(b)$ of b-positions. If $\forall a \in A.\mathsf{Stp}(a) \subseteq \mathsf{Pos}(a)$, that is, descendants are defined for rewrite steps, then we call the DRS a *residual rewriting system* (RRS). We use $\mathcal{D}$, $\mathcal{E}$, $\mathcal{F}$, $\mathcal{G}$ and $\mathcal{H}$ to range over DRSs and ϕ, ψ, χ, ω to range over positions. Let $\mathcal{D}$ be a DRS.

1. The trace function is straightforwardly extended from rewrite steps to conversions:

   (a) $\lfloor 0 \rfloor$ is the identity.

   (b) $\lfloor (u{,}0)d \rfloor =^{\mathrm{def}} \lfloor u \rfloor \,;\, \lfloor d \rfloor$.

   (c) $\lfloor (u{,}1)d \rfloor =^{\mathrm{def}} \lfloor u^{-1} \rfloor^{-1} \,;\, \lfloor d \rfloor$.

   Note that the relation compositions are always defined and that we have $\lfloor d \rfloor^{-1} = \lfloor d^{-1} \rfloor$ and $\lfloor d \rfloor \,;\, \lfloor e \rfloor = \lfloor d \,;\, e \rfloor$.

2. Let d be a conversion and let ϕ and ψ be positions in $\mathcal{D}$. If $\phi \lfloor d \rfloor \psi$, then ψ is a *descendant of* ϕ *by* d and ϕ is an *origin* or *ancestor of* ψ *by* d. Descendants of rewrite steps which are rewrite steps again are also called *residuals*. Positions which have 0, 1, > 1 descendant(s) by d are said to be *d-erased*, *d-linear* and *d-duplicated*, respectively. Positions which have have 0, 1, > 1 origin(s) by d are said to be *d-created*, *d-linear* and *d-shared*, respectively.

REMARK 2.4.2. 1. Our definitions are copied from [GLM92], where *abstract reduction systems* are multi-graphs and come equipped with a set of residual relations on the edges, indexed by the labelled edges. So, nothing new is presented in the above, but our terminology differs from the one employed there. In particular, instead of their *residual* and $[\![\cdot]\!]$, we use descendant and $\lfloor\cdot\rfloor$. We will use the word residual only in case of descendants of redexes which are redexes again, because in general we will trace not just redexes but any suitable property.

2. The *concurrent transition systems* of Stark, see e.g. [Sta90], can be viewed as DRSs where for each rewrite step, its trace relation is a partial function satisfying some symmetry constraints.

In the literature there are two ways for dealing with so-called *developments*, i.e. rewrites issuing from some object a in which only steps are performed which are residuals of steps in (a subset of) $\mathsf{Stp}(a)$. One way is via residual systems as defined above and another one via licensing systems. Licensing systems appear under various guises such as 'marked' or 'underlined' systems. In such system, the objects are endowed with sets of 'licenses' controlling which rewrite steps are allowed. Furthermore, there is an inheritance mechanism for licenses along rewrite steps. We mainly work with descendant systems, but sometimes (in proofs) it is convenient to have the extra structure on objects provided by licensing systems, explicitly available.

DEFINITION 2.4.3. 1. Let $\mathcal{D} =^{\mathrm{def}} \langle A, I, \rightarrow, \mathsf{Pos}, \lfloor\cdot\rfloor\rangle$ be an RRS. The *licensing* LRS $\lceil\mathcal{D}\rceil$ of $\mathcal{D}$ is defined by $\lceil\mathcal{D}\rceil =^{\mathrm{def}} \langle B, I, \rightsquigarrow\rangle$, where

(a) $B =^{\mathrm{def}} \{(a,\mathcal{V}) \in A \times \mathfrak{P}(\mathsf{Pos}).\mathcal{V} \subseteq \mathsf{Pos}(a)\}$. The possible licenses of an object are sets of positions,

(b) $(a,\Psi) \rightsquigarrow_i (b,\{\Psi \lfloor a \rightarrow_i b\rfloor\})$, if $a \rightarrow_i b \in \Psi$. A step can be performed if it is licensed, and the descendants of licensed positions are licensed.

Obviously, to each rewrite d from (a,Ψ) to (b,Υ) in $\lceil\mathcal{D}\rceil$, there corresponds a unique rewrite denoted by $\lfloor d\rfloor$ from a to b in $\mathcal{D}$, by forgetting the second component of the objects in it. $\lfloor d\rfloor$ is called a *development* (*of* Ψ). It is called *complete* if $\Upsilon = \emptyset$.

2. Let $\mathcal{J} =^{\mathrm{def}} \langle B, I, \rightsquigarrow\rangle$ be the licensing LRS of an RRS. Define $\mathcal{D} =^{\mathrm{def}} \langle A, I, \rightarrow, \mathsf{Pos}, \lfloor\cdot\rfloor\rangle$ by

(a) $A =^{\mathrm{def}} \pi_1(B)$,

(b) $\mathsf{Pos}(a) =^{\mathrm{def}} \bigcup b \in B \;\&\; \pi_1(b) = a.\pi_2(b)$, $\mathsf{Pos} =^{\mathrm{def}} \bigcup a \in A.\mathsf{Pos}(a)$,

(c) $a \rightarrow_i b$ iff $\exists\Psi.\exists\Upsilon.(a,\Psi) \rightsquigarrow_i (b,\Upsilon)$,

(d) For $u =^{\mathrm{def}} a \rightarrow_i b$, $\psi \in \mathsf{Pos}(a)$ and $\chi \in \mathsf{Pos}(b)$, $\psi\lfloor u\rfloor\chi$ if and only if $\exists\Upsilon.(a,\{u,\psi\}) \rightsquigarrow_i (b,\Upsilon) \;\&\; \chi \in \Upsilon$.

The DRS $\mathcal{D}$ is denoted by $\lfloor \mathcal{J} \rfloor$.

From the definition it follows that if d is a development of Φ and $\Phi \subseteq \Psi$, then d is also a development of Ψ. Hence, two coinitial developments of Φ and Ψ can both be viewed as developments of $\Phi \cup \Psi$. Furthermore, every sub-rewrite of a development is again a development, in particular, every suffix of a complete development is again a complete development.

REMARK 2.4.4. The idea of tracing properties of a term along a rewrite is omnipresent in the literature on rewriting. We mention some of its guises.

1. In the realm of λ-calculus, Church and Rosser ([CR36]) used *residuals* to keep track of the redex parts of a λ-term along reductions. To keep track also of other parts of a term, various devices were introduced, such as *underlining*, *marking* and *colouring* of symbols and subterms ([Bar84, Klo80]).

2. A *residual* relation (more abstract than the one of Church and Rosser) for tracing redexes was also employed by Gonthier, Lévy and Melliès in an axiomatic approach to the standardisation theorem ([GLM92]).

3. In the abstract setting of tree-manipulating systems, Rosen employed a *residue map* to trace positions in trees along rewrite steps ([Ros73]) in order to prove a confluence result for those systems.

4. In the world of interactive programming environments, it is useful (e.g. for debugging purposes) to keep track of subterms (e.g. identifiers) of the syntax tree of a program along its execution via term rewriting. Subterms of the initial term are related to subterms of an intermediate term via so-called *origins*. For the case of execution by conditional term rewriting these were defined by Van Deursen, Klint and Tip in [DKT93]. Recently, this has been extended to the case of higher-order term rewriting in [DD93].

5. In the algebraic approach to graph rewriting the descendant relation is known as the *track function*. See e.g. [CR93, Plu93b].

6. Finally, licensing systems as above were introduced by De Vrijer in his PhD thesis [Vri87]. The difference with descendant systems is that licenses put extra structure on the objects, whereas in descendant systems the structure which objects have is expressed by the rewrite steps they can perform.

A DRS and its licensing LRS convey the same information if the trace relation $\lfloor \cdot \rfloor$ satisfies the following condition.

$$\{u\}\lfloor u \rfloor\emptyset \tag{2.1}$$

Doing a step erases the step itself.

Proposition 2.4.5. *Let $\mathcal{D}$ be an RRS satisfying (2.1), such that* $\mathsf{Pos} = \bigcup a \in A.\mathsf{Pos}(a)$ *(every property is possessed by some object), then* $\lfloor\lceil\mathcal{D}\rceil\rfloor = \mathcal{D}$.

Proof Let $\mathcal{D} =^{\mathrm{def}} \langle A, I, \rightarrow, \mathsf{Pos}, \lfloor\underrightarrow{\cdot}\rfloor\rangle$ be a DRS, $\mathcal{J} =^{\mathrm{def}} \langle B, I, \rightsquigarrow\rangle =^{\mathrm{def}} \lceil\mathcal{D}\rceil$ and $\mathcal{D}' =^{\mathrm{def}} \langle A', I, \rightarrow', \mathsf{Pos}', \lfloor\underrightarrow{\cdot}\rfloor'\rangle =^{\mathrm{def}} \lfloor\mathcal{J}\rfloor$.

1. there is an obvious correspondence between objects $a \in A$ and $(a,\emptyset) \in B$.
2. $\mathsf{Pos}'(a) = \mathsf{Pos}(a)$ by straightforward calculation. $\mathsf{Pos} = \mathsf{Pos}'$ using the equality of Pos and Pos' and the assumption that every property is possessed by some object.
3. $a \rightarrow_i b \implies (a,\{a \rightarrow_i b\}) \rightsquigarrow_i (b,\emptyset) \implies \exists\Psi.\exists\Upsilon.(a,\mathcal{V}) \rightsquigarrow_i (b,\mathcal{W}) \implies a \rightarrow'_i b$, by definition and (2.1). The other direction is trivial.
4. Let $u =^{\mathrm{def}} a \rightarrow_i b$. $\psi\lfloor\underrightarrow{u}\rfloor\chi \implies (a,\{u,\psi\}) \rightsquigarrow_i (b,\{\{\psi\}\ \lfloor\underrightarrow{u}\rfloor\})\ \&\ \chi \in \{\{\psi\}\ \lfloor\underrightarrow{u}\rfloor\} \implies \psi\lfloor\underrightarrow{u}\rfloor'\chi$, by definition and (2.1). The other direction is just as easy.

$\odot$

Example 2.4.6. Can one always distinguish the descendant relation induced by 0 (i.e. the identity) from the one induced by another rewrite in a residual rewriting system? Of course, there may be several derivations leading from one object to another. Let $\mathcal{D} =^{\mathrm{def}} \langle\{a,b\}, \{i,j,k\}, \rightarrow, \mathsf{Pos}, \lfloor\underrightarrow{\cdot}\rfloor\rangle$ be a DRS, such that $u =^{\mathrm{def}} a \rightarrow_i b$ and $v =^{\mathrm{def}} b \rightarrow_j a$, having empty descendant relations. Then we have the derivations 0 and uv from a to b. Still, these derivations can be distinguished, because $\lfloor\underrightarrow{0}\rfloor$ is the identity, while $\lfloor\underrightarrow{uv}\rfloor$ is the empty relation on $\mathsf{Stp}(a)$. In general this is not the case, even for DRSs having finite developments. If we extend the rewrite relation by $w =^{\mathrm{def}} b \rightarrow_i a$ and the descendant relations by $u\lfloor\underrightarrow{u}\rfloor w$, $w\lfloor\underrightarrow{v}\rfloor u$ then $\lfloor\underrightarrow{0}\rfloor = \lfloor\underrightarrow{uv}\rfloor$. This problem is solved if (2.1) is assumed.

Parametricity of a descendant rewriting system roughly expresses that the 'names' of the symbols at positions which are traced does not matter. For example, term rewriting systems are parametric with respect to the positions of variables. As another example, lambda calculus is parametric with respect to 'symbols' see [Klo80, I.10.1.2].

Definition 2.4.7. A DRS is called *parametric*, if for every two rewrites d, e from a to normal form b, we have $\lfloor\underrightarrow{d}\rfloor = \lfloor\underrightarrow{e}\rfloor$.

Remark 2.4.8. In general one cannot expect to have $\lfloor\underrightarrow{d}\rfloor = \lfloor\underrightarrow{e}\rfloor$, for rewrites from a to b, for *any* b. Both rewrites might end in b due to a syntactical accident, see e.g. [HL91a, p. 402].

The main reason for introducing DRSs was to formulate the notion of orthogonality. Orthogonality consists of three parts. First, distinct actions consume distinct resources ('consistency'). Second, actions may interact as long as this interaction is finitary ('finite developments'). Finally, the order in which distinct actions are performed does not influence the effect on other resources ('parametricity'). In other words, no matter in what order these actions are performed the effect on their surroundings is always the same. These constraints are satisfied by the independent actions occurring in concurrency theory. The definition of orthogonality will be such that if we consider an orthogonal term rewriting system, combinatory reduction system or higher-order rewriting system in the usual sense of orthogonality, i.e. having left-linear and non-ambiguous rules (cf. [DJ90, Klo92, KOR93, Nip93]), and construct the RRS naturally associated to it, the RRS will be orthogonal in our sense.

DEFINITION 2.4.9. Let $\mathcal{D} =^{\mathrm{def}} \langle A, I, \rightarrow, \mathsf{Pos}, \lfloor\cdot\rfloor\rangle$ be an RRS, and let $\mathcal{J} =^{\mathrm{def}} \langle B, I, \rightsquigarrow\rangle$ be the licensing LRS of $\mathcal{D}$. Let Φ and Ψ be sets of a-positions.

1. The set Φ is *consistent*, if $\langle(a,\Phi) \rightsquigarrow^*\rangle$ is confluent. The RRS $\mathcal{D}$ is *consistent* if every set of positions is consistent, i.e. $CR(\lceil\mathcal{D}\rceil)$.

2. The RRS $\mathcal{D}$ *has* finite developments, if every development is finite, i.e. $SN(\lceil\mathcal{D}\rceil)$.

3. The RRS $\mathcal{D}$ *has* parametric developments, if $\lceil\mathcal{D}\rceil$ is parametric in the sense of Definition 2.4.7, i.e. $\lfloor d\rfloor = \lfloor e\rfloor$, for complete developments d and e.

The RRS $\mathcal{D}$ is *orthogonal*, if it is consistent and has finite, parametric developments.

REMARK 2.4.10. The definition of orthogonality as given above is based on Axiom 0 (Finite Developments) of [GLM92].

Not the whole standard machinery of orthogonality, which will be presented next, is needed to prove confluence. To this end, consistency suffices but can be weakened in several ways. In order to capture a notion of harmless syntactical coincident as in the case of lambda beta-eta calculus or of PCF with parallel or, we introduce weakly orthogonal RRSs and prove confluence for them.

DEFINITION 2.4.11. Let $\mathcal{D}$ be an RRS and $\mathcal{I}$ be its licensing LRS. The RRS $\mathcal{D}$ is *weakly orthogonal*, if there is a property P on objects of $\mathcal{I}$ satisfying:

1. if $P((a,\Phi))$ and $u =^{\mathrm{def}} a \rightarrow_i b$, then there exist $\Psi \supseteq \Phi$ and Ψ', such that $(a,\Psi) \rightsquigarrow^* (b,\Psi')$, $P((a,\Psi))$ and $P((b,\Psi'))$,

2. $P \implies COMP$, i.e. if an object satisfies P then it is complete.

In a diagram

$$\begin{tikzcd}
 & a \arrow[r, "\mathcal{D}"'] & b \\
(a,\Phi) \subseteq & (a,\Psi) \arrow[r, dashed, two heads] \arrow[d, dashed, two heads] & (b,\Psi') \arrow[d, dashed, two heads] \\
(c,\emptyset) \subseteq \arrow[from=2-1, to=3-1, two heads] & (c,\Psi') \arrow[r, dashed, two heads] & (d,\emptyset)
\end{tikzcd}$$

where the bottom square can be constructed because $P((a,\Psi))$ hence (a,Ψ) is complete.

Every orthogonal DRS is also weakly orthogonal, because we can take $P((a,\Phi)) =^{\text{def}} \Phi$ *is consistent* and $\Psi =^{\text{def}} \Phi \cup \{u\}$. Note that in a weakly orthogonal system a set consisting of a single rewrite step need not be orthogonal, but (by taking the empty set for Φ) it can always be 'simulated' by a development of some set of steps.

THEOREM 2.4.12. *Every weakly orthogonal RRS is confluent.*

PROOF By Lemma 2.2.5(2), stripping suffices for proving confluence. This is proved by the following diagram.

$$\begin{tikzcd}
 & a_0 \arrow[r, "j_0"'] & a_1 \arrow[r, "j_1"'] & a_2 \arrow[r, two heads] & a_m \\
a_0 \arrow[d, "i"] & a_0 \arrow[r, dashed, two heads] \arrow[d, dashed, two heads] \arrow[dr, phantom, "\Phi_0"] & a_1 \arrow[r, dashed, two heads] \arrow[d, dashed, two heads] \arrow[dr, phantom, "\Phi_1"] & a_2 \arrow[r, dashed, two heads] \arrow[d, dashed, two heads] & a_m \arrow[d, dashed, two heads] \\
b_0 & b_0 \arrow[r, dashed, two heads] & b_1 \arrow[r, dashed, two heads] & b_2 \arrow[r, dashed, two heads] & b_m
\end{tikzcd}$$

For each square the two ways of chasing it are developments of the set of positions in its centre. By weak orthogonality we can find each set Φ_n from the already constructed development on its left-hand side and the step $a_n \to_{j_n} a_{n+1}$. In particular, in the orthogonal case we can take $\Phi_0 =^{\text{def}} \{a_0 \to_i b_0, a_0 \to_{j_0} a_1\}$. $\odot$

The point of orthogonality is not just that confluence can be proved. Confluence diagrams can be constructed in a canonical way. That is, constructing the diagram by filling it with 'elementary diagrams' (such as in the consistent case in the proof above) always terminates and every way of doing this results in 'permutation equivalent' rewrites, i.e. rewrites which can be transformed into each other. Another way of looking at the same process is by viewing it as gradually transforming the initial divergence into a convergence. This gives rise to a so-called 'proof-ordering' relation. The two situations can be depicted as

$$
\begin{tikzcd}
& a \arrow[dl] \arrow[dr] & \\
b \arrow[dr] \arrow[rr, Leftrightarrow, "\text{permutation}"'] & & c \arrow[dl] \\
& d &
\end{tikzcd}
\qquad
\begin{tikzcd}
& a \arrow[dl] \arrow[dr] & \\
b \arrow[dr] \arrow[rr, phantom, "\text{proof} - \text{order}"] & & c \arrow[dl] \\
& d &
\end{tikzcd}
$$

The theory of permutation equivalence was developed for lambda calculus by Lévy ([Lév80]) and for TRSs by Huet and Lévy ([HL91a]) via an algebraic approach.

The permutation equivalence relation is indeed an equivalence relation on rewrites, so one can search for an ARS implementing or even deciding it. Similarly, one can look for an ARS implementing the proof-ordering relation. Such a 'diagrammatic' approach was taken by Klop ([Klo80]). In this approach, both transformations are generated by local transformations on conversions as follows.

Definition 2.4.13. Let $\mathcal{D} =^{\text{def}} \langle A, I, \longrightarrow, \mathsf{Pos}, \lfloor\cdot\rfloor_{\to}\rangle$ be a DRS.

1. Define relations $\mathsf{Prm}_{\mathcal{D}}$ and $\mathsf{Pfo}_{\mathcal{D}}$ on $\mathcal{D}$-conversions as follows. If $d =^{\text{def}} ue'$ and $e =^{\text{def}} vd'$ are rewrites such that they are both complete developments of $\{u,v\}$ and $\lfloor d\rfloor_{\to} = \lfloor e\rfloor_{\to}$, then
   (a) the quadruple $\langle u, v, e', d'\rangle$ is called an *elementary diagram*,
   (b) d *is elementary permutation equivalent to* e, denoted by $d\ \mathsf{Prm}_{\mathcal{D}}\ e$, and
   (c) $e'\,;d'^{-1}$ *is elementary lower than* $u^{-1}\,;v$, denoted by $u^{-1}\,;v\ \mathsf{Pfo}_{\mathcal{D}}\ e'\,;d'^{-1}$.
2. Let R be a relation on rewrites (between the same objects). The *proof step* ARS, $\Rightarrow_R$, on $\mathcal{D}$-conversions is defined by $e\,;d\,;f \Rightarrow_R e\,;d'\,;f =^{\text{def}} d\ R\ d'$. The relation is obtained from R by closing under 'contexts' consisting of conversions. The usual operations on ARSs apply.
   (a) The *permutation equivalence* relation $\simeq_{\mathcal{D}}$ is defined to be the restriction to rewrites of $\Rightarrow^{*}_{\mathsf{Prm}_{\mathcal{D}}}$.
   (b) The *proof order* relation $\rightarrowtail_{\mathcal{D}}$ is defined to be equal to $\Rightarrow^{+}_{\mathsf{Pfo}_{\mathcal{D}}}$.

Remark 2.4.14. The permutation equivalence relation is usually denoted by $\equiv$ (see e.g. [HL91a]).

By definition, permutation equivalence is an equivalence relation. Furthermore, permutation equivalent rewrites have the same effect on their surroundings, i.e. $d \simeq e \implies \lfloor d\rfloor_{\to} = \lfloor e\rfloor_{\to}$. We now define some special shapes for diagrams.

Definition 2.4.15. Let $\mathcal{D}$ be a DRS.

1. An elementary diagram $\langle u, u, 0, 0\rangle$ is called *trivial*.
2. An elementary diagram $\langle u, v, w, r\rangle$ such that $i = \mathsf{lab}(u) = \mathsf{lab}(w)$ and $j = \mathsf{lab}(v) = \mathsf{lab}(r)$ is called an *independence* diagram for i, j.
3. The DRS $\mathcal{D}$ *has elementary diagrams* if there exist parametric developments of every pair of coinitial steps u, v, that is, cofinal developments ud and ve of $\{u,v\}$ such that $\lfloor \underrightarrow{ud} \rfloor = \lfloor \underrightarrow{ve} \rfloor$.
4. The DRS $\mathcal{D}$ is called CR^+ if $\rightarrowtail\!\!\rightarrow_{\mathcal{D}}$ is well-founded.

The trace equivalences studies (see e.g. [Sta90]) in concurrency theory are permutation equivalences generated by sets of independence diagrams. The notion CR^+ was introduced by Klop [Klo80] and expresses that filling with elementary diagrams always terminates. (Actually he requires $\rightarrowtail\!\!\rightarrow_{\mathcal{D}}$ to be well-founded only on divergences, so one could call his property CON^+.) Because a $\rightarrowtail\!\!\rightarrow_{\mathcal{D}}$-normal form is a convergence this implies CR. It is a strengthening of CR because, e.g. weakly orthogonal systems may not be CR^+ although they are CR. Orthogonal systems however are CR^+.

If all developments are finite, then orthogonality can be localised to the property of having elementary diagrams. This is shown in the following proposition corresponding to Property (D) of Curry and Feys, [CF58, p. 114], Corollary 11.2.24 of [Bar84] and Axiom 5 (One-step confluence) of [Mel93].

Proposition 2.4.16. *Let $\mathcal{D}$ be an RRS. If $\mathcal{D}$ has finite developments and elementary diagrams, then $\mathcal{D}$ is orthogonal.*

Proof We have to prove that $\mathcal{D}$ is consistent and $\lfloor \underrightarrow{d} \rfloor = \lfloor \underrightarrow{e} \rfloor$ for complete developments d and e of some set of positions Φ. The proof is analogous to the usual proof of Newman's Lemma, proving the following predicate P on divergences $\langle d, e\rangle$ issuing from (a,Φ) in $\lceil \mathcal{D} \rceil$: *there exists a convergence $\langle e', d'\rangle$ such that $f =^{\mathrm{def}} \lfloor d\,; e' \rfloor$ and $g =^{\mathrm{def}} \lfloor e\,; d' \rfloor$ are complete developments of Φ and* $\lfloor \underrightarrow{f} \rfloor = \lfloor \underrightarrow{g} \rfloor$, by noetherian induction on the origin (a,Φ) of the divergence ordered by $\rightsquigarrow^+$. The proof is shown in the following picture.

$$
\begin{tikzcd}
(a,\Phi) \arrow[r, "v"] \arrow[d, "u"'] & (c,\Upsilon) \arrow[r, twoheadrightarrow] \arrow[d, dashed] & {} \arrow[d, dashed, twoheadrightarrow, "d''"'] \\
(b,\Psi) \arrow[r, dashed, twoheadrightarrow] \arrow[d] & {} \arrow[r, dashed, twoheadrightarrow] & (d,\emptyset) \arrow[d, dashed, "0"'] \\
{} \arrow[rr, dashed, twoheadrightarrow, "e'"] & & (d,\emptyset)
\end{tikzcd}
$$

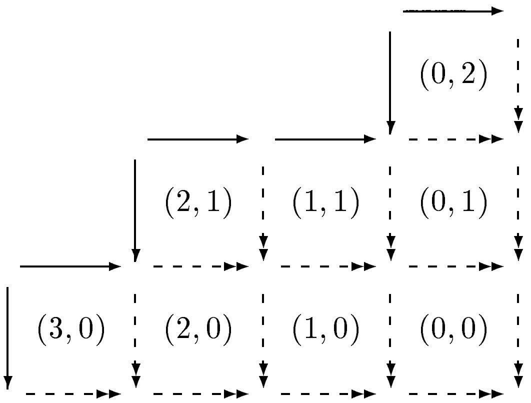
The arrows are labelled by developments. One first uses the assumptions to complete the steps u and v to complete developments f' and g' of $\{u,v\}$ such that $\lfloor f' \rfloor = \lfloor g' \rfloor$. These complete developments can be viewed as developments of Φ, so give rise to two cofinal rewrites issuing from (a,Φ) and hence to the upper-left square of the diagram. Then apply the induction hypothesis (twice) as indicated in the diagram, using that for any rewrites d and e, we have $\lfloor d \rfloor ; \lfloor e \rfloor = \lfloor d ; e \rfloor$. $\odot$

The next proposition states that trying to construct a confluence diagram by tiling with elementary diagrams always terminates. Apart from giving the confluence result, this also gives an induction principle for proving properties of particular diagram constructions.

PROPOSITION 2.4.17. *Let $\mathcal{D}$ be a DRS.*

1. *The elementary ordering is well-founded, if $\mathcal{D}$ has finite developments.*
2. *If $\mathcal{D}$ is orthogonal, then $CR^+(\mathcal{D})$.*

PROOF 1. Suppose we start with some conversion and repeatedly perform a proof-order step, as in the following diagram.



Intuitively, if a step is performed inside some square, we are constructing developments of the origin of the square. Since developments are finite by assumption, in each square only finitely many steps can be performed and because there are only finitely many squares, this process must end. Formally, one constructs a graph where the vertices are labelled by objects (a,Φ) of $\lceil \mathcal{D} \rceil$ and the edges are either 'inclusion' edges $(a,\Phi) \subseteq (a,\Psi)$ (where $\Phi \subseteq \Psi$) or 'step' edges $(a,\Phi) \rightsquigarrow_{(i,(m,n))} (b,\Psi)$ where $(a,\Phi) \rightsquigarrow_i (b,\Psi)$ is a step in $\lceil \mathcal{D} \rceil$ and (m,n) is a pair of natural numbers indicating in which 'square' the step takes place (as in the diagram above). The inclusion steps are needed to go from the border of one square to the border of the next one. The measure of a graph is the multiset consisting of pairs $((m,n),(a,\Phi))$ of the edges $(a,\Phi) \rightsquigarrow_{(i,(m,n))} (b,\Psi)$ in the 'lower border' of the graph. Graphs are ordered by ordering their measure by

$((> \times >) \times_{lex} \rightsquigarrow^{+})_{mul}$. Now the measure of the graph decreases when we perform either an inclusion step:

$$
\begin{tikzcd}
(a,\Phi) \arrow[r, "{(m,n)}"'] \arrow[d, phantom, sloped, "\subseteq"] & (b,\Psi) \arrow[d, phantom, sloped, "\subseteq"] \\
(a,\Phi') \arrow[r, dashed, "{(m,n \dot{-} 1)}"'] & (b,\Psi')
\end{tikzcd}
\qquad
\begin{tikzcd}
(a,\Phi) \arrow[r, phantom, "\subseteq"] \arrow[d, "{(m,n)}"] & (a,\Phi') \arrow[d, dashed, "{(m \dot{-} 1,n)}"] \\
(b,\Psi) \arrow[r, phantom, "\subseteq"] & (b,\Psi')
\end{tikzcd}
$$

or a proof-order step

$$
\begin{tikzcd}
(a,\Phi) \arrow[r, "{(m,n)}"'] \arrow[d, "{(m,n)}"] & (c,\Upsilon) \arrow[d, dashed, two heads, "{(m,n)}"] \\
(b,\Psi) \arrow[r, dashed, two heads, "{(m,n)}"'] & (d,\Xi)
\end{tikzcd}
$$

obtained from 'lifting' a proof-order step from $b \leftarrow a \rightarrow c$ in $\mathcal{D}$ first to $\lceil \mathcal{D} \rceil$ and subsequently adding the label (m,n). In both cases we add the 'dashed' step-edges to the graph and in the first case also the inclusion-edges.

2. By definition of orthogonality and 1.

$\odot$

The Derivations Space Orthogonal RRSs can be viewed as an abstraction from orthogonal term rewriting systems, such that the *Parallel Moves lemma* ([HL91a, Lem. 2.2]) holds. This can be made more precise by defining the notion corresponding to *multi-derivation* and the operations corresponding to $\backslash$ and $\sqcup$ of [HL91a] for orthogonal RRSs.

DEFINITION 2.4.18. Let $\mathcal{D} =^{\text{def}} \langle A, I, \longrightarrow, \mathsf{Stp}, \lfloor \cdot \rfloor \rangle$ be an orthogonal RRS. Define the *derivation* LRS $\mathcal{I} =^{\text{def}} \langle A, \mathfrak{P}(\mathsf{Stp}(\mathcal{D})), \rightsquigarrow \rangle$ and the operations $\backslash$ and $\sqcup$ by

1. if there exists a complete development of a set $\mathcal{U}$ of rewrite steps, from a to b, then $a \rightsquigarrow_{(i,\mathcal{U})} b$ and this step is called an *elementary multi-derivation contracting the set* $\mathcal{U}$.

2. Let u and v be coinitial elementary multi-derivations contracting $\mathcal{U}$ and $\mathcal{V}$, corresponding to complete developments d and e in $\mathcal{D}$. The *residual* $v \backslash u$ *of* v *by* u is the elementary derivation contracting the set $\{\mathcal{V} \lfloor d \rfloor\}$.

3. $u \sqcup v =^{\text{def}} u \,; (v \backslash u)$.

With this definition, Section 2 of [HL91a] on the derivations space of orthogonal term rewriting systems can be repeated for orthogonal residual rewriting systems without difficulty. For example:

Lemma 2.4.19. (parallel moves) *Let $\mathcal{I}$ be the derivation LRS of an orthogonal RRS $\mathcal{D}$. If u and v are coinitial elementary multi-derivations, then $u \sqcup v$ and $v \sqcup u$ are cofinal and for every other elementary multiderivation w, we have $w \backslash (u \sqcup v) = w \backslash (v \sqcup u)$.*

Proof If u, v correspond to complete developments d, e in $\mathcal{D}$ and contract $\mathcal{U}$ and $\mathcal{V}$. Let $v \backslash u$, $u \backslash v$ correspond to complete developments e' and d' contracting $\{\mathcal{U}\ \lfloor d \rfloor\}$ and $\{\mathcal{V}\ \lfloor e \rfloor\}$. Then $u \sqcup v$, $v \sqcup u$ correspond to complete developments $d\,;e'$ and $e\,;d'$ both contracting $\mathcal{U} \cup \mathcal{V}$. By orthogonality of $\mathcal{D}$, all complete developments end in the same term and induce the same descendant relation, so we are done. $\odot$

Standard Rewrites Because permutation equivalence is an equivalence relation, one can look for representatives of permutation equivalence classes satisfying some nice properties and, even better to give a procedure for constructing such representatives from given rewrites. Of course, this depends on the definition of 'nice property'.

For example, for λ-calculus every equivalence class contains a unique 'standard' rewrite which contracts redexes in a leftmost outermost fashion. In his thesis Klop showed that these can be constructed from a given rewrite sequence by repeatedly transforming 'anti-standard' pairs in the rewrite sequence into 'more standard' ones. The nice property standard reductions have is that they allow for a simple proof that 'normal' rewriting, i.e. repeatedly contracting the leftmost outermost redex, is a normalising strategy (see e.g. [Hin78b]).

For term rewriting the leftmost outermost strategy is not normalising in general, but outermost rewriting of needed redexes is. This is proved using standard rewrites which are defined now as outside-in rewrites contracting (somewhat arbitrarily) always the leftmost of the external redex occurrences ([HL91a]). Neededness of a redex is not decidable in general, so the strategy although normalising is not practical. For the class of sequential systems one can determine effectively a needed redex, but sequentiality itself is undecidable. Finally, for the subclass of strongly sequential systems everything is decidable ([HL91b]).

We think that descendant rewriting systems are at the right level to study these properties, cf. [GLM92, Klo80].

2.5. Confluence Modulo

In [Hue80], it is noted that a theory is usually defined by axioms of two forms: "structural axioms" such as associativity and commutativity of operators, and "simplification rules" such as "**if** true **then** x **else** $y \to x$". In order to prove properties, it is therefore convenient to partition the set of rules accordingly and study properties by studying the interaction between the "structural axioms" and the "simplification rules". Like in the case of having only "simplification rules", one really wants to localise properties. For example, to deduce confluence up to "structural equivalence" from the local interaction. In this section we give new proofs of two such lemmata of Huet [Hue80, Lem. 2.7,2.8], in the spirit of the confluence by diagrams method.

DEFINITION 2.5.1. Let A be a set. Let $\to$ be an ARS on A. Let $\vdash\!\dashv$ be a symmetrical relation on A, and $\sim =^{\text{def}} \vdash\!\dashv^*$ its transitive, reflexive closure. We define the following 'modulo'-relations.

1. the relation $\Uparrow_{\sim} =^{\text{def}} \twoheadleftarrow ; \sim ; \twoheadrightarrow$, is called $\to$-*meetability modulo* $\sim$,
2. the relation $\Downarrow_{\sim} =^{\text{def}} \twoheadrightarrow ; \sim ; \twoheadleftarrow$, is called $\to$-*joinability modulo* $\sim$.

To study confluence up to the equivalence relation $\sim$, apart from the interaction between $\to$-steps, also the interaction between $\to$ and $\vdash\!\dashv$ has to be taken into account. This gives rise to the following properties.

1. if $\uparrow \subseteq \Downarrow_{\sim}$, then $\to$ is *locally confluent modulo* $\sim$.

$$\begin{tikzcd}
& & a \arrow[dll] \arrow[drr] & & \\
b \arrow[dr, dashed, two heads] & & & & c \arrow[dl, dashed, two heads] \\
& d \arrow[rr, dash, dashed, "\sim"] & & d' &
\end{tikzcd}$$

2. if $\leftarrow ; \vdash\!\dashv \subseteq \Downarrow_{\sim}$, then $\to$ is *locally coherent with* $\vdash\!\dashv$.

$$\begin{tikzcd}
& & a \arrow[dll] \arrow[drr, dash] & & \\
b \arrow[dr, dashed, two heads] & & & & c \arrow[dl, dashed, two heads] \\
& d \arrow[rr, dash, dashed, "\sim"] & & d' &
\end{tikzcd}$$

3. if $\Uparrow_{\sim} \subseteq \Downarrow_{\sim}$, then $\to$ is *confluent modulo* $\sim$.

$$\begin{tikzcd}
& a \arrow[dl] \arrow[rr, dash, "\sim"] & & a' \arrow[dr] & \\
b \arrow[dr, dashed, two heads] & & & & c \arrow[dl, dashed, two heads] \\
& d \arrow[rr, dash, dashed, "\sim"] & & d' &
\end{tikzcd}$$

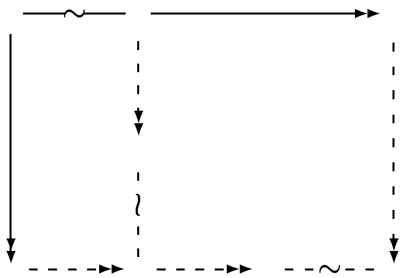
REMARK 2.5.2. The concepts above stem from [Hue80] and [JK86], but we have used the terminology of [MS91]. In particular the interaction between the rewrite relation $\rightarrow$ and the symmetrical relation $\vdash\!\dashv$ generating the equivalence $\sim$ is expressed by 'coherence'.

PROPOSITION 2.5.3. *Let $\vdash\!\dashv$ be a symmetrical relation on A and $\sim =^{\mathrm{def}} \vdash\!\dashv^*$. Let $\rightarrow$ be an ARS on A which is locally confluent modulo $\sim$.*

1. *If $SN(\rightarrow)$ and $\rightarrow$ is locally coherent with $\sim$, then $\rightarrow$ is confluent modulo $\sim$.*

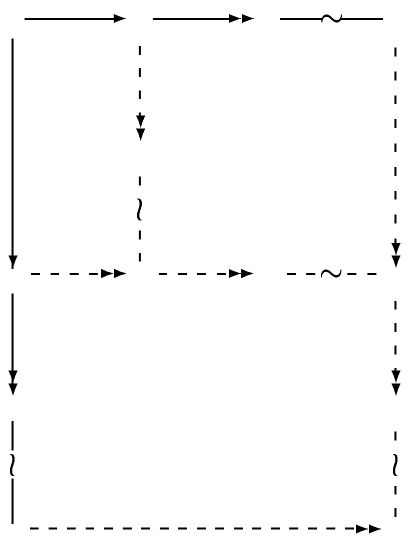
2. *If $SN(\rightarrow/\sim)$ and $\rightarrow$ is locally coherent with $\vdash\!\dashv$, then $\rightarrow$ is confluent modulo $\sim$.*

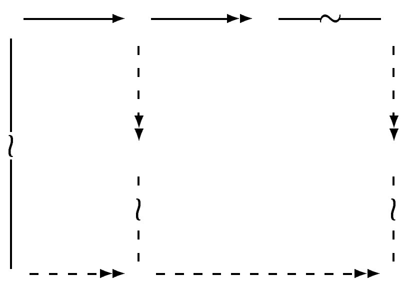
PROOF In both cases we prove $\Diamond(\twoheadrightarrow ; \sim)$ by well-founded induction and conclude by using this twice as in the following diagram:



The idea of the proof is that in a convergence $\twoheadrightarrow ; \sim ; \sim ; \twoheadleftarrow$ the two equivalence steps can be combined into one and then can be 'bicycled around the corner' in the diagram in order to make the diagrams fit.

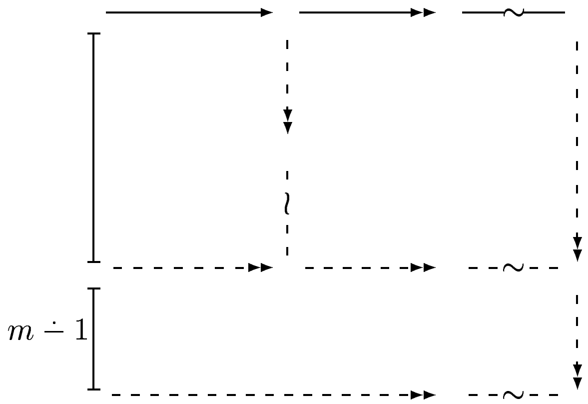
1. $\Diamond(\twoheadrightarrow ; \sim)$ is proved by well-founded induction on the origin a for a divergence $b \sim ; \twoheadleftarrow\; a \twoheadrightarrow ; \sim c$ ordered by $\rightarrow^+$. We distinguish four cases depending on whether the diverging rewrites start with a 'computation'- or with a 'modulo'-step. Three cases are shown in the following two diagrams (the diagram on the right can be mirrored).





The case $b \sim a \sim c$ is left to the reader. In the first diagram the assumption that $\rightarrow$ is locally confluent modulo $\sim$ and the induction hypothesis (twice) are made use of, whilst in the second diagram the assumption that $\rightarrow$ is locally coherent modulo $\sim$ and the induction hypothesis are used.

2. $\Diamond(\twoheadrightarrow\,;\sim)$ is proved by well-founded induction on the pair $(a/{\sim},m)$ for a divergence $b \sim ; \twoheadleftarrow\ a\ \twoheadrightarrow ; \sim c$, where $m =^{\mathrm{def}} 0$, if either $b \sim ; \leftarrow^{+}$ $a \rightarrow^{+} ; \sim c$ or $b \sim a \sim c$, otherwise we can write either $b \vdash\!\dashv^{m} a \rightarrow^{+} ; \sim c$ or $b \sim ; \leftarrow^{+} a \vdash\!\dashv^{m} c$. The pairs are ordered by $(\rightarrow/\sim)^{+} \times_{lex} >$. The diagram on the left in the first item can be copied without further ado. The second one has to be refined by splitting an equivalence step $\sim$ into a number of symmetric steps $\vdash\!\dashv^{m}$:



The diagram can be mirrored again, because the order is symmetric in the diverging rewrites.

$\odot$

REMARK 2.5.4. The above proposition is well known [Hue80, Lem. 2.7,2.8] (cf. also [MS91, Lem. 7.4] and [BO93, Lem. 1.2.4]). Nevertheless, we present it because the proof given is much simpler than the one in [Hue80] ([BO93]) and more elementary than the one in [MS91]. Moreover, in the case of $\vdash\!\dashv$ being the identity relation, the proofs specialise to the usual proof of Newman's Lemma.

3 Higher-Order Rewriting

In this chapter, which is based on joint work with Femke van Raamsdonk ([OR93, OR94]), we consider *Higher-Order Rewriting Systems* (HORSs) and in particular the class of orthogonal HORSs.

In the literature there is not a universally accepted notion of higher-order, let alone a notion of higher-order rewriting system. We will add another notion of higher-order rewriting system. It is an attempt to unify the theory for the existing higher-order rewriting systems by parametrising the *substitution calculus* used in these.

Let us explain by some examples what is meant by substitution calculus. Suppose we have a term rewriting system (TRS) specified by the rule $\mathtt{2/2} \to \mathtt{1}$. Then one can rewrite the term $\mathtt{2/2}$ to $\mathtt{1}$. Applying a rule is done by *matching* the left-hand side of the rule to the term, symbol by symbol. The maximum time this takes is obviously proportional to the size of the left-hand side.

TRS rules can be applied in any *context*, a term with a hole ($\Box$) in it. For example, one can rewrite the term $\mathtt{1/(2/2)}$ to $\mathtt{1/1}$. In this case, the context is $\mathtt{1/}\Box$. Still it is not difficult to see whether a rule can be applied to a term or not: just try to apply the rule to each subterm in turn.

This was only an extremely simple kind of TRS. The rules are *closed*, i.e. they do not contain *free variables*. In general rules may be *open*, i.e. contain free variables. For example, one can have rules like $x\mathtt{/}x \to \mathtt{1}$ and $x\mathtt{/1} \to x$. In order to show that such a rule can be applied to a term, one has to find a proper *valuation* of the variables. For example, $\mathtt{2/2}$ rewrites by the first rule to $\mathtt{1}$, using the valuation $x \mapsto \mathtt{2}$. By the second rule $\mathtt{2/1}$ rewrites to $\mathtt{2}$, using the same valuation. The matching to be performed can now be more complex; if a term variable occurs more than once in a left-hand side, as in the first rule, its complexity is no longer proportional to the size of the left-hand side, because the subterms at these positions must be tested for equality. Moreover, in order to obtain the result the valuation has to be applied to the right-hand side of the rule. This is still not terribly complex, but it is not obvious any more that performing a rewrite step consists of one 'elementary' step.

The situation is more complex if one considers the higher-order rewriting systems in the literature. These are systems which have a so-called *binding* term-forming operation. There one can write rules like the μ-recursion rule: $\mu\xi.x(\xi) \to x(\mu\xi.x(\xi))$. Using this rule, one can rewrite $s =^{\mathrm{def}} \mu\xi.\mathtt{F}(\xi)(\xi)$ to $\mathtt{F}(s)(s)$, using the valuation $x \mapsto (y \mapsto \mathtt{F}(y)(y))$. In general, people have to think a bit in order to see that a step is indeed a consequence of a higher-order rule. The same holds for machines. For some higher-order rewriting systems, e.g. the Higher-Order Term Rewriting Systems of Wolfram [Wol93] it is not even known whether matching is decidable or not.

In our opinion it is therefore justified to have the matching process be performed by steps in a, usually more elementary, calculus which we call *substitution calculus*. Rules will become closed expressions, and a rule can be applied to a term if the term is equal modulo the substitution calculus to one containing the left-hand side of the rule. In this way, the complexity of the matching is put inside the substitution calculus.

It is often convenient to have some λ-calculus (see Section 3.2) as the substitution calculus. In that case, the TRS rule $x/x \to \mathtt{1}$ can be written as $\xi.\xi/\xi \to \zeta.\mathtt{1}$. We can rewrite $\mathtt{2/2}$ to $\mathtt{1}$ as follows. For the first term we have the β-expansion $\mathtt{2/2} \leftarrow_\beta (\xi.\xi/\xi)\mathtt{2}$. In the *expanded* term $(\xi.\xi/\xi)\mathtt{2}$ we can 'see' the left-hand side $\xi.\xi/\xi$ of the rule, *replace* it by the right-hand side $\zeta.\mathtt{1}$ and *reduce* the result $(\zeta.\mathtt{1})\mathtt{2}$ in one β-step to $\mathtt{1}$.

This three phase process of expansion, replacement and reduction will be common to all these systems, but the substitution calculus is not fixed. For example, we could use Combinatory Logic (see Example 3.3.2) as a substitution calculus. Then, the same rule can be written (now in applicative notation) as $S(/)(I) \to K(\mathtt{1})$. We can rewrite $/(\mathtt{2})(\mathtt{2})$ to $\mathtt{1}$ as follows. We have the expansion $/(\mathtt{2})(\mathtt{2}) \leftarrow_{CL} /(\mathtt{2})(I(\mathtt{2})) \leftarrow_{CL} S(/)(I)(\mathtt{2})$. Replacing the left-hand side $S(/)(I)$ by the right-hand side $K(\mathtt{1})$, the result $K(\mathtt{1})(\mathtt{2})$ is obtained, which reduces in one CL-step to $\mathtt{1}$.

In both cases the rules are closed, but the left- and right-hand side are 'polluted' by symbols from the substitution calculus which indicate how substitution takes place.

By introducing HORSs we hope to achieve two goals.

1. To obtain a better understanding of the rôle of substitution in term rewriting. This is made possible by explicitly introducing a substitution calculus in the definition of a Higher-Order Rewriting System.

2. Obtaining a general framework for studying and proving properties of (higher-order) term rewriting systems, such as confluence and termination.

In this chapter, we find sufficient conditions on HORSs to prove a 'confluence by orthogonality' result in the sense of Section 2.4. We then show how this result can be used to prove orthogonal TRSs, CRSs and HRSs confluent.

3.1. Higher-Order Rewriting Systems

After having motivated Higher-Order Rewriting Systems (HORSs) in the preceding section, we now formalise them. A HORS is a triple consisting of an alphabet, a substitution calculus and a set of rewrite rules. The rewrite rules generate a rewrite relation on the set of terms, which are objects built from the symbols in the alphabet using the term building operations of application and abstraction.

DEFINITION 3.1.1. An *alphabet* $(a, b, c \in)$ $\mathcal{A}$ is a countable set consisting of the following symbols:

1. A binary term forming symbol $\cdot(\cdot)$ called *application*,

2. A binary term forming symbol $\cdot.\cdot$ called *abstraction*,

3. The following nullary symbols:

   (a) symbols F, G, H, ... for *substitution operators* in $\mathcal{O}_{\mathcal{SC}}$,

   (b) symbols ξ, ζ, ς, ... for *substitution variables* (or *bound variables*) in $\mathcal{B}\mathsf{var}$,

   (c) symbols $\mathtt{F}$, $\mathtt{G}$, $\mathtt{H}$, ... for *operators* in $\mathcal{O}_{\mathcal{R}}$,

   (d) symbols x, y, z, ... for *term variables* (or *free variables*) in $\mathcal{F}\mathsf{var}$. Among these symbols, we have distinguished symbols $\Box_1$, $\Box_2$, $\Box_3$, ... called *holes*.

   The substitution operators together with the bound variables form the *substitution alphabet* $\mathcal{A}_{\mathcal{SC}} =^{\mathrm{def}} \mathcal{O}_{\mathcal{SC}} \cup \mathcal{B}\mathsf{var}$. The operators together with the free variables form the *rewrite alphabet* $\mathcal{A}_{\mathcal{R}} =^{\mathrm{def}} \mathcal{O}_{\mathcal{R}} \cup \mathcal{F}\mathsf{var}$.

From the symbols in the alphabet, raw preterms are built in the following way. For any set Z of bound variables, the set (s, t, $r \in$) $\mathbb{RPT}(\mathcal{A}, Z)$ of *raw ($\mathcal{A}$-)preterms on* Z is defined by:

1. a bound variable ξ is a raw preterm on Z, if $\xi \in Z$. All other nullary symbols are raw preterms on Z,

2. if s, t are raw preterms on Z, then $s(t)$ is one too,

3. if s is a raw preterm on $Z \cup \{\xi\}$, then $\xi.s$ is a raw preterm on Z.

The set $\mathbb{RPT}(\mathcal{A}) =^{\mathrm{def}} \mathbb{RPT}(\mathcal{A}, \mathcal{B}\mathsf{var})$, is the set of *raw preterms*. Equality of raw preterms is denoted by $\equiv$.

An alphabet is partitioned into two sets of symbols: one set of symbols of the substitution calculus and another one consisting of 'parameters' of the substitution calculus. The intended usage of the substitution symbols is explained in the following example.

EXAMPLE 3.1.2. Let $\mathcal{A}$ be an alphabet for a HORS with substitution calculus $\mathcal{SC}$.

1. If $\mathcal{SC}$ is some λ-calculus, then the set of substitution operators is empty and the set of bound variables consists of the 'usual' bound variables. (In most modern texts on λ-calculus there is no notational distinction between free and bound variables, but one assumes Barendregt's Variable Convention (see Remark 3.4.4). Assuming this convention, it is no restriction to partition the set of variables, into sets of bound and free variables as we have done.)

2. If $\mathcal{SC}$ is Combinatory Logic, then $\{I, K, S\}$ is the set of substitution operators and the set of bound variables is empty.

REMARK 3.1.3. It may seem a bit strange to a priori separate the sets of bound and free variables. The point is that substitution for bound variables is dealt with by the substitution calculus of a HORS, while substitution for free variables is dealt with on a meta level. This difference will be of crucial importance when tracing of symbols along rewrite steps is defined. The bound variables cannot be traced, but the free variables can. The reason is that rewrite steps are performed modulo the substitution calculus. Furthermore, this separation is not uncommon in logic, see e.g. [Pra65], where bound variables are called *variables* and free variables are called *parameters*. His *pseudo-terms* correspond to our raw preterms.

On raw preterms over an alphabet we define the usual operations.

DEFINITION 3.1.4. Let $\mathcal{A}$ be an alphabet. Let $(\phi, \psi, \chi \in)$ $\mathsf{Pos} =^{\mathrm{def}} \{0,1\}^*$ be the set of *positions* or *occurrences*. The function $\mathsf{Pos} : \mathbb{RPT}(\mathcal{A}) \to \mathfrak{P}(\mathsf{Pos})$, mapping a raw preterm to its *set of positions*, the function $\mathsf{top} : \mathbb{RPT}(\mathcal{A}) \to \mathcal{A}$, mapping a raw preterm to its *top symbol*, and the partial function $\backslash : \mathbb{RPT}(\mathcal{A}) \times \mathsf{Pos} \to \mathbb{RPT}(\mathcal{A})$, mapping a pair s, ϕ to the *raw subpreterm at position* ϕ *in* s, are defined by:

1. if s is a nullary symbol, then

$$\begin{array}{rcl} \mathsf{Pos}(s) & =^{\mathrm{def}} & \{\varepsilon\} \\ \mathsf{top}(s) & =^{\mathrm{def}} & s \\ \varepsilon\backslash s & =^{\mathrm{def}} & s \end{array}$$

2. if $s =^{\mathrm{def}} s_0(s_1)$, then

$$\begin{array}{rcl} \mathsf{Pos}(s) & =^{\mathrm{def}} & \{\varepsilon\} \cup \{0\sigma.\sigma \in \mathsf{Pos}(s_0)\} \cup \{1\sigma.\sigma \in \mathsf{Pos}(s_1)\} \\ \mathsf{top}(s) & =^{\mathrm{def}} & \cdot(\cdot) \\ \varepsilon\backslash s & =^{\mathrm{def}} & s \\ 0\sigma\backslash s & =^{\mathrm{def}} & \sigma\backslash s_0 \\ 1\sigma\backslash s & =^{\mathrm{def}} & \sigma\backslash s_1 \end{array}$$

3. if $s =^{\mathrm{def}} \xi.s_0$, then

$$\begin{array}{rcl} \mathsf{Pos}(s) & =^{\mathrm{def}} & \{\varepsilon\} \cup \{0\sigma.\sigma \in \mathsf{Pos}(s_0)\} \\ \mathsf{top}(s) & =^{\mathrm{def}} & \cdot.\cdot \\ \varepsilon\backslash s & =^{\mathrm{def}} & s \\ 0\sigma\backslash s & =^{\mathrm{def}} & \sigma\backslash s_0 \end{array}$$

Furthermore, we define the partial function $s(\phi) =^{\mathrm{def}} \mathsf{top}(\phi\backslash s)$, mapping a pair s, ϕ to the *symbol at position* ϕ *in* s. A symbol $a \in \mathcal{A}$ *occurs* (*m times*) in a raw preterm if it is at some position (m distinct positions) in the raw preterm. Such a position is then called an a-position. For $A \subseteq \mathcal{A}$, we define $A\mathsf{Pos}(s) =^{\mathrm{def}} \{\phi \in \mathsf{Pos}(s).s(\phi) \in A\}$. The function Bvar (Fvar) maps a raw preterm to the set of bound (free) variables occurring in it.

The raw preterms are a bit too raw, we use the functions defined above to restrict attention to the classes of interest.

DEFINITION 3.1.5. Let $\mathcal{A}$ be an alphabet. A *preterm* is a raw $\mathcal{A}$-preterm on $\emptyset$. From now on, we restrict s, t, r to range over the set $\mathbb{PT}(\mathcal{A})$ of preterms. If $\mathsf{Fvar}(s) = \emptyset$, then the preterm s is called *closed*, otherwise *open*. A preterm is an *m-ary precontext*, if the holes occurring in it are among $\square_1,\ldots,\square_m$. We use C, D, E to range over precontexts. We denote m-ary precontexts by $C\boxed{m}$, $D\boxed{m}$, $E\boxed{m}$ and unary precontexts just by $C\square$, $D\square$, $E\square$. The result of replacing the occurrences of $\square_1,\ldots,\square_m$ by preterms $s_1,\ldots,s_m$ in an m-ary precontext $C\boxed{m}$ is denoted by $C\boxed{s_1,\ldots,s_m}$. An m-ary context is *linear* if every hole $\square_1,\ldots,\square_m$ occurs exactly once in it.

REMARK 3.1.6. Note that in a replacement $C\boxed{s}$, there is no binding between the 'context part' $C\square$ and the 'substitution part' s, because s is required to be a preterm, not just a raw preterm. For example, we cannot have a replacement $\xi.\boxed{\xi}$, because the substitution part ξ is a raw preterm on $\{\xi\}$, but not on $\emptyset$, i.e. ξ is not a preterm.

DEFINITION 3.1.7. A *Higher-Order Rewriting System* (HORS) is a structure $\langle \mathcal{A}, \mathcal{SC}, \mathcal{R}\rangle$ consisting of an alphabet $\mathcal{A}$, a *substitution calculus* $\mathcal{SC}$, and a set $\mathcal{R}$ of *rewrite rules*, which satisfy the following conditions.

1. $\mathcal{SC}$ is some kind of rewriting system to which an abstract rewriting system $\to_{\mathcal{SC}}$ on $\mathbb{PT}(\mathcal{A})$ is associated.

2. $\mathcal{R}$ is a set of rewrite rules. A *rewrite rule* is a pair (l,r) of closed preterms also denoted by $l \to r$. We use $\aleph$, $\beth$, $\gimel$ and $\daleth$ to range over rewrite rules. The first and second component of a rule $\aleph$ are denoted by $\mathsf{lhs}(\aleph)$ and $\mathsf{rhs}(\aleph)$ and are its *left-* and *right-hand side*, respectively.

We use $\mathcal{H}$, $\mathcal{I}$, $\mathcal{J}$ to range over HORSs. Let $\mathcal{H} =^{\mathrm{def}} \langle \mathcal{A}, \mathcal{SC}, \mathcal{R}\rangle$ be a HORS. We associate two abstract rewriting systems to a HORS. They are both defined on the set of preterms.

1. For a rule $\aleph =^{\mathrm{def}} l \to r$ of $\mathcal{H}$ and precontext $C\square$, we define the *replacement of* $\aleph$ *in* $C\square$ by $C\boxed{l} \to_{C\boxed{\aleph}} C\boxed{r}$. This is generalised to *replacement of* $\aleph$ in an arbitrary precontext, by defining $\to_{\aleph} =^{\mathrm{def}} \bigcup C\square.\!\to_{C\boxed{\aleph}}$ and to *replacement* of an arbitrary rule, by defining $\to_{\mathcal{R}} =^{\mathrm{def}} \bigcup \aleph \in \mathcal{R}.\!\to_{\aleph}$.

2. The actual *rewrite relation* is obtained by 'replacing modulo the substitution calculus': $\to_{\mathcal{H}} =^{\mathrm{def}} \leftrightarrow^*_{\mathcal{SC}} ; \to_{\mathcal{R}} ; \leftrightarrow^*_{\mathcal{SC}}$. Abstract rewriting notions are defined for $\mathcal{H}$ via its rewrite relation $\to_{\mathcal{H}}$.

The 'replacement' in a replacement step is literal replacement. No matters of binding are taken into account; they cannot be, because preterms are replaced by others in such a step (see Remark 3.1.6). It is the task of the substitution calculus to deal with matters of binding. This point of view is also taken by van de Pol in [Pol93], where a similar definition of the rewrite step relation is given, taking $\lambda_{\eta}^{\rightarrow}$-calculus as substitution calculus. It differs from the views taken by Klop [Klo80], Nipkow [Nip91], Felty [Fel91], Kahrs [Kah] and Wolfram [Wol93], in that they define the rewrite relation of their respective higher-order rewriting formalisms via 'contexts and substitutions', thereby introducing unnecessary noise in the definition of the system.

REMARK 3.1.8. A replacement step $C\boxed{l} \rightarrow_{C\boxed{\aleph}} C\boxed{r}$ can be viewed as the composition of an *erasure step* $C\boxed{l} \rightarrow_{C\Box} C\Box$ followed by a *creation step* $C\Box \leftarrow_{C\Box} C\boxed{r}$. This point of view will be adopted when defining the descendant relation induced by a replacement step. Alternatively, one can view the erasure step as an *abstraction step*, abstracting over the structure of the left-hand side l of the rule $\aleph$. The creation step is then an *specialisation step*, specialising to the right-hand side of the rule $\aleph$.

The HORSs introduced above can be seen as 'full' rewriting systems, since rewriting is performed on the set of all preterms. Often there are useful 'substructures' of a rewriting system where the (raw) preterm formation is restricted in some way. For example, simply typed λ-calculus is a substructure of λ-calculus, because terms are required to be simply typable. Next to 'full' HORSs, we now also admit all its substructures as HORSs. We will call such substructures *restricted* HORSs.

Note that restricting term formation also influences the substitution calculus involved. For example, in a HORS with β-reduction as substitution calculus rule, restricting term formation from polymorphically typable λ-terms to simply typable λ-terms, changes the substitution calculus from polymorphically typed λ-calculus into simply typed λ-calculus. The conditions which will be put on a HORS in the sequel are not automatically closed under restriction, so need to be proved again for the restriction. For example, to show completeness (condition A1) of simply typed λ-calculus from completeness of polymorphically typed λ-calculus requires a proof that simply typed λ-terms are closed under β-reduction (the *subject reduction property*).

Furthermore, if a left-hand side of a rule is present in a term, it should be replaceable by its right-hand side, resulting in a preterm. That is, in the substitution calculus there should be 'types' of variables expressing that such a replacement is always allowed (cf. the preceding remark). In the sequel, we assume that this requirement is met. For example, in the case of simply typed λ-calculus, left- and right-hand sides of rules should have the same simple type and there should be sufficiently many variables of these types.

Up till now, nothing is specified about the interaction between the substitution calculus and the replacement steps. The next example shows the intended way of interaction.

EXAMPLE 3.1.9. Take for $\mathcal{SC}$ the $\lambda_{\overline{\eta}}^{\rightarrow}$-calculus. Take as preterms the set of 'mathematical expressions' and as rules:

$$1 + 1 \quad \rightarrow \quad 2$$

$$
\begin{array}{rcl}
1+2 & \to & 3 \\
2+2 & \to & 4 \\
\xi.\xi/\xi & \to & \xi.1 \\
\xi.2\,\mathrm{x}\,\xi & \to & \xi.\xi+\xi
\end{array}
$$

The last two rules express that dividing a number by itself is equal to one and that multiplying a number by two is equal to adding the number to itself. In the rules we have employed the usual infix notation for the binary operators, e.g. $1+2$ is written instead of $+(1)(2)$. We now have the following computation

$$
\begin{array}{rl}
\multicolumn{2}{l}{2\,\mathrm{x}\,((1+2)/3+3/(1+2)) \leftarrow_\beta} \\
\leftarrow_\beta & (\boxed{\xi.2\mathrm{x}\xi})((1+2)/3+3/(1+2)) \\
\to & (\xi.\xi+\xi)((1+2)/3+3/(1+2)) \\
\leftarrow_\beta & (\xi.\xi+\xi)((\zeta.\zeta/3+3/\zeta)(\boxed{1+2})) \\
\to & (\xi.\xi+\xi)((\zeta.\zeta/3+3/\zeta)3) \\
\to_\beta & (\xi.\xi+\xi)(3/3+3/3) \\
\leftarrow_\beta & (\xi.\xi+\xi)((\zeta.\zeta+\zeta)(3/3)) \\
\leftarrow_\beta & (\xi.\xi+\xi)((\zeta.\zeta+\zeta)((\boxed{\varsigma.\varsigma/\varsigma})3)) \\
\to & (\xi.\xi+\xi)((\zeta.\zeta+\zeta)((\varsigma.1)3)) \\
\to_\beta & (\xi.\xi+\xi)((\zeta.\zeta+\zeta)1) \\
\to_\beta & (\xi.\xi+\xi)(\boxed{1+1}) \\
\to & (\xi.\xi+\xi)2 \\
\to_\beta & \boxed{2+2} \\
\to & 4
\end{array}
$$

In the case of a replacement step, the left-hand side of the rule is boxed in the computation. Remark that the number of replacement steps (5), which is the same as the number of rewrite steps, is quite optimal. This due to the fact that the two terms $3/3$ in $3/3+3/3$ arising in the middle of the computation are 'shared', although they do arise from initially different expressions. Of course, noting that this kind of sharing is possible is in general an expensive operation.

REMARK 3.1.10. So we have cheated a little bit in the example doing literal replacement. In the computation, the preterm $\varsigma.\varsigma/\varsigma$ is seen as literally the same as the preterm $\xi.\xi/\xi$. Of course, they are only the same up to the name of the bound variable, i.e. up to α-conversion (see [Bar84, Con. 2.1.13]). To be perfectly rigorous, $\mathcal{SC}$ should also contain the rule of α-conversion.

In the following we will put ever more conditions on HORSs, culminating in a list of conditions enabling us to derive orthogonality of HORS. If a condition concerns the substitution calculus, we illustrate it by briefly indicating how the condition is met by some λ-calculi. The syntax of these calculi, as well as detailed proofs how $\lambda_{\bar{\eta}}^{\rightarrow}$-calculus (simply typed λ-calculus with β- and $\bar{\eta}$-reduction) satisfies all the conditions required for a substitution calculus, can be found in Section 3.2.

Rewriting in a HORS is performed modulo the substitution calculus, or, stated differently, rewriting is performed on substitution calculus equivalence classes. It is usually easier to work with unique representatives of such classes than to work with the whole class. This brings us to the first condition on a HORS: the substitution calculus should implement finding a unique representative starting from any preterm in an equivalence class. These unique representatives will then be our objects of interest.

A 1. (completeness) *The HORS* $\langle \mathcal{A}, \mathcal{SC}, \mathcal{R} \rangle$ *satisfies* *COMP*$(\rightarrow_{\mathcal{SC}})$.

Definition 3.1.11. Let $\mathcal{H} =^{\mathrm{def}} \langle \mathcal{A}, \mathcal{SC}, \mathcal{R} \rangle$ be a HORS. A *context* is a precontext in $\mathcal{SC}$-normal form. A *term* is a preterm in $\mathcal{SC}$-normal form. The set of $\mathcal{A}$-terms is denoted by $\mathbb{T}(\mathcal{A})$. In general, for a notion on preterms, the corresponding notion on $\mathcal{SC}$-normal forms is obtained by omitting 'pre' from it. The rewrite relation $\rightarrow_{\mathcal{H}}$ is restricted to a relation on the set $\mathbb{T}(\mathcal{A})$ terms instead of on preterms in the following.

Three classes of 'terms' on an alphabet $\mathcal{A}$ were introduced: $\mathbb{RPT}(\mathcal{A}) \supseteq \mathbb{PT}(\mathcal{A}) \supseteq \mathbb{T}(\mathcal{A})$. The set $\mathbb{PT}(\mathcal{A})$ can be thought of as the set of raw preterms from $\mathbb{RPT}(\mathcal{A})$, which are semantically interesting. The set of terms $\mathbb{T}(\mathcal{A})$ is the set of representatives of the $\mathcal{SC}$-equivalence classes of preterms, i.e. $\mathbb{T}(\mathcal{A}) = \mathbb{PT}(\mathcal{A})/\leftrightarrow_{\mathcal{SC}}^{*}$.

Example 3.1.12. The 'term' ξ is a raw preterm, but not a preterm. Taking $\lambda_{\bar{\eta}}^{\rightarrow}$ as $\mathcal{SC}$, $(\xi.\xi)x$ is a preterm, but not a term, because it can be β-reduced to x. Note that terms may contain symbols from the substitution alphabet $\mathcal{A}_{\mathcal{SC}}$. In fact, as soon as a term contains a bound variable, it contains a symbol from the substitution alphabet.

Note that by completeness of the substitution calculus, any $\mathcal{H}$-rewrite (sequence of rewrite steps) between two terms in the previous sense (i.e. there might be steps between preterms in it) projects onto a unique $\mathcal{H}$-rewrite of the same length consisting of steps between terms only. For example a rewrite $s_1 \rightarrow_{\mathcal{H}} s_2 \rightarrow_{\mathcal{H}} s_3$, from term s_1 to term s_3 via the preterm s_2, can be transformed into a rewrite from s_1 to s_3 via the term s_2' as in the following diagram.

$$\begin{tikzcd}
t_1 \arrow[r] \arrow[d, leftrightarrow, "*" ', "\mathcal{SC}"] & r_1 \arrow[r, leftrightarrow, "*", "\mathcal{SC}" '] \arrow[dr, "\mathcal{SC}" '] & s_2 & t_2 \arrow[l, leftrightarrow, "*" ', "\mathcal{SC}"] \arrow[r] \arrow[dl, "\mathcal{SC}"] & r_2 \arrow[d, leftrightarrow, "*" ', "\mathcal{SC}"] \\
s_1 & & s_2' & & s_3
\end{tikzcd}$$

It is easy to see that the abstract rewriting properties for the relation $\rightarrow_{\mathcal{H}}$ modulo (cf. Section 2.5) the substitution calculus on preterms and the same relation restricted to terms coincide. Moreover, every rewrite step $s \to_{\mathcal{H}} t$ can be viewed as $s \twoheadleftarrow_{\mathcal{SC}} s' \to_{\mathcal{R}} t' \twoheadrightarrow_{\mathcal{SC}} t$, that is, as composed of an *expansion* in the substitution calculus, a *replacement* by applying some rule, and a *reduction* in the substitution calculus. Linking this three phase process to ordinary term rewriting, the expansion can be thought of as the *matching* phase and the reduction as the *substitution* phase.

REMARK 3.1.13. Restricting rewriting to rewriting on terms, is more convenient for studying the rewrite relation, because one has more 'grip' on terms than on preterms. However, from an efficiency point of view it is often better to rewrite on preterms. For example, transforming the computation in Example 3.1.9 to a computation consisting of rewrite steps between terms, leads to a significant overhead of substitution steps.

The same holds for requiring rewrite steps to be of the form expansion, replacement, reduction instead of conversion, replacement, conversion. For example, when $\mathcal{SC}$ is some λ-calculus, the theory of optimal *reduction* (cf. [Lan93]) is different from the theory of optimal *conversion*; sometimes it may be advantageous to perform an expansion to introduce sharing. For example, if some λ-term s has a long reduction to normal form. If we want to convert the term $\xi.\xi(s)(s)$ to normal form, it is better to share s by first β-expanding the term to $(\zeta.\xi.\xi(\zeta)(\zeta))s$ and only then start reducing s to normal form. In this case, the possibility to share is already present in the initial term. In general, it might turn up anywhere in a reduction. We do not know of work on this subject. (Note however a similar concern in term graph rewriting: collapsed tree rewriting of Plump and Hoffmann [Plu93a, Hof92].)

If the substitution calculus is not complete, unwanted phenomena appear if, nevertheless, the rewrite relation is restricted to terms, i.e. to normal forms with respect to the substitution calculus.

EXAMPLE 3.1.14.

1. Take for $\mathcal{SC}$, the non-confluent but terminating rewriting system defined by $B \to_{\mathcal{SC}} K$, $B \to_{\mathcal{SC}} K'$ and let $\mathcal{R} =^{\text{def}} \{\mathtt{A} \to B\}$. The term $\mathtt{A}$ rewrites to K as well as to K'. One does not expect a seemingly deterministic rule like $\mathtt{A} \to B$ to be non-confluent.

2. Take $\mathcal{SC}$ to be the non-terminating but confluent rewriting system, defined by $B \to_{\mathcal{SC}} B$ and let $\mathcal{R} =^{\text{def}} \{\mathtt{A} \to B\}$. Although it does look like one can apply this rule to the term $\mathtt{A}$, it cannot, because B has no normal form. One would expect that if a left-hand side is present in a term, it can be replaced by the corresponding right-hand side.

An expansion $s \twoheadleftarrow_{\mathcal{SC}} C\boxed{l}$ of a term splits the term s into a context part ($C\Box$) and a part which is to be replaced (l). The substitution calculus is used to glue these parts together. The next condition entails that only glue *on the borders* of both parts is needed, not inside either $C\Box$ or l, for the gluing.

A 2. (glue) *The HORS $\mathcal{H}$ satisfies:*

1. *rules of $\mathcal{H}$ consist of pairs of closed terms (not just closed preterms),*

2. $\rightarrow_{\mathcal{H}} = \rightarrow'_{\mathcal{H}}$, *where* $\rightarrow'_{\mathcal{H}}$ *is* $\rightarrow_{\mathcal{H}}$ *restricted to rewrite steps for which the precontext of its replacement step is a context.*

Every HORS can be transformed into a step-equivalent one satisfying A2 provided that its substitution calculus is context free.

Definition 3.1.15. Let $\mathcal{H} =^{\mathrm{def}} \langle \mathcal{A}, \mathcal{SC}, \mathcal{R} \rangle$ be a HORS. The substitution calculus $\mathcal{SC}$ is called *context free* if it satisfies the following three conditions:

1. *($\mathcal{SC}$ preserves closedness)* if l is closed and $l \rightarrow_{\mathcal{SC}} g$, then g is closed,

2. *(closure of conversion under contexts)* if $l \leftrightarrow^*_{\mathcal{SC}} g$ is a *closed* conversion, that is, all the preterms in it are closed, then $C\boxed{l} \leftrightarrow^*_{\mathcal{SC}} C\boxed{g}$, for any context $C\Box$,

3. *(closure of conversion under substitution)* if l is a closed term and $C\Box \leftrightarrow^*_{\mathcal{SC}} D\Box$, then $C\boxed{l} \leftrightarrow^*_{\mathcal{SC}} D\boxed{l}$,

If $\mathcal{SC}$ is not context free, then it is *context sensitive*.

Remark 3.1.16. For the next proposition to hold, we could weaken the above definition by replacing both occurrences of $\leftrightarrow^*_{\mathcal{SC}}$ in the hypotheses by $\twoheadrightarrow^!_{\mathcal{SC}}$.

Proposition 3.1.17. *Let $\mathcal{H} =^{\mathrm{def}} \langle \mathcal{A}, \mathcal{SC}, \mathcal{R} \rangle$ be a HORS satisfying A1 for which $\mathcal{SC}$ is context free, then there exists a HORS $\mathcal{H}' =^{\mathrm{def}} \langle \mathcal{A}, \mathcal{SC}, \mathcal{R}' \rangle$ satisfying A1–2 such that $\rightarrow_{\mathcal{H}} = \rightarrow_{\mathcal{H}'}$.*

Proof Define $\mathcal{R}'$ as follows: $\mathcal{R}' =^{\mathrm{def}} \{\aleph{\downarrow_{\mathcal{SC}}}.\aleph \in \mathcal{R}\}$, where $(l \rightarrow r){\downarrow_{\mathcal{SC}}} =^{\mathrm{def}} l{\downarrow_{\mathcal{SC}}} \rightarrow r{\downarrow_{\mathcal{SC}}}$. If $s \twoheadleftarrow_{\mathcal{SC}} C\boxed{l} \rightarrow_{C\boxed{\aleph}} C\boxed{r} \twoheadrightarrow_{\mathcal{SC}} t$, then one can construct the following diagram

$$\begin{tikzcd}
C\boxed{l} \arrow[rrr, "C\boxed{\aleph}"'] \arrow[dr, leftrightarrow, "*"', "\mathcal{SC}"] \arrow[dd, "\mathcal{SC}"] & & & C\boxed{r} \arrow[dd, "\mathcal{SC}"] \\
& C{\downarrow_{\mathcal{SC}}}\boxed{l{\downarrow_{\mathcal{SC}}}} \arrow[r, "C{\downarrow_{\mathcal{SC}}}\boxed{\aleph{\downarrow_{\mathcal{SC}}}}"'] \arrow[dl, "\mathcal{SC}"] & C{\downarrow_{\mathcal{SC}}}\boxed{r{\downarrow_{\mathcal{SC}}}} \arrow[ur, leftrightarrow, "*", "\mathcal{SC}"'] \arrow[dr, "\mathcal{SC}"'] & \\
s \arrow[rrr] & & & t
\end{tikzcd}$$

giving a rewrite step in the required format in $\mathcal{H}'$. $\odot$

Remark 3.1.18. Examples of substitution calculi in the realm of simply typed λ-calculi satisfying the conditions in the proposition are β-, $\beta\eta$- and $\beta\bar{\eta}$-reduction. Note that the rewrite *step* relation need not be closed under either contexts or substitutions. For the first two calculi, the rewrite step relation is indeed closed under substitution, i.e. if $C\Box \rightarrow_{\mathcal{SC}} D\Box$, then $C\boxed{l} \rightarrow_{\mathcal{SC}} D\boxed{l}$.

However, for $\beta\bar{\eta}$-reduction this property fails, as shown by $\Box \to_{\beta\bar{\eta}} \xi.\Box(\xi)$, but not $\zeta.s \to_{\beta\bar{\eta}} \xi.(\zeta.s)\xi$ (see e.g. [Aka93, Exa. 1.(1)]). Simlarly, for the first two calculi the rewrite step relation is closed under contexts, i.e. if $l \to_{\mathcal{SC}} l'$, then $C\boxed{l} \to_{\mathcal{SC}} C\boxed{l'}$. Again, this property fails for $\beta\bar{\eta}$-reduction, as shown by $s \to_{\beta\bar{\eta}} \xi.s(\xi)$, but not $s(t) \to_{\beta\bar{\eta}} (\xi.s(\xi))t$ (see e.g. [Aka93, Exa. 1.(1)]).

If the transformation process in the proof of the preceding proposition is applied to a context sensitive HORS $\mathcal{H}$, the resulting HORS $\mathcal{H}'$ may have a rewrite relation different from $\mathcal{H}$, as shown in the next example.

EXAMPLE 3.1.19. Let $\mathcal{H} =^{\mathrm{def}} \langle \mathcal{A}, \mathcal{SC}, \mathcal{R} \rangle$ be a HORS.

1. Suppose $A \to_{\mathcal{SC}} K$ and $\aleph =^{\mathrm{def}} A \to B$. Then $\mathtt{G}(\boxed{A}) \to_{\mathtt{G}(\boxed{\aleph})} \mathtt{G}(\boxed{B})$, but since $\aleph{\downarrow_{\mathcal{SC}}} =^{\mathrm{def}} K \to B$, the term $\mathtt{G}(A)$ cannot be rewritten using this rule. ($\mathtt{G}(A)$ is in $\mathcal{SC}$-normal form, because we didn't take the closure of $\to_{\mathcal{SC}}$ under contexts.)

2. Suppose $\mathcal{SC}$ is generated by the rules:

$$\begin{array}{rcl} G(K) & \to & A \\ G(x) & \to & x \text{ if } x \not\equiv K \end{array}$$

and $\aleph =^{\mathrm{def}} K \to \mathtt{B}$. Then $A \leftarrow_{\mathcal{SC}} G(\boxed{K}) \to_{G(\boxed{\aleph})} G(\boxed{\mathtt{B}}) \to_{\mathcal{SC}} \mathtt{B}$, but A cannot be expanded to a preterm $C\boxed{K}$ such that $C\Box$ is in $\mathcal{SC}$-normal form.

The idea of the substitution calculus is to extract a left-hand side l from a term s by expanding the term into a context $C\Box$ containing the left-hand side. But so far, we have not put any restrictions on the number of occurrences of the hole in $C\Box$ in a replacement step, that is, containment is specified very loosely. There might even be no occurrence of the hole at all in $C\Box$. This last fact makes that HORSs are *never* terminating. To make HORSs useful for deciding equational theories, one can restrict attention to a set of rewrite steps having the same transitive, reflexive closure but which is not intrinsically non-terminating.

A 3. (serialisation) *The HORS* $\mathcal{H} =^{\mathrm{def}} \langle \mathcal{A}, \mathcal{SC}, \mathcal{R} \rangle$ *satisfies* $\to_{\mathcal{H}} = \to_{\mathcal{H}}^{\prime +}$, *where* $\to'_{\mathcal{H}}$ *is* $\mathcal{H}$ *restricted to* serial *rewrite steps, that is, rewrite steps for which the context in its replacement step is linear (the hole occurs exactly once in the context of the replacement step).*

The first part of the next example shows how contexts in which no hole occurs can turn up in the transformation process above. The second part shows that conditions A1–2 are not sufficient to guarantee A3.

EXAMPLE 3.1.20. Let $\mathcal{H} =^{\mathrm{def}} \langle \mathcal{A}, \mathcal{SC}, \mathcal{R} \rangle$.

1. Let $\rightarrow_{\mathcal{SC}}$ be generated by the rule $G(x)(y) \rightarrow x$. Suppose we have any rule $\aleph =^{\mathrm{def}} l \rightarrow r$, then, for any term s, $s \leftarrow_{\mathcal{SC}} G(s)(\boxed{l}) \rightarrow_{G(s)(\boxed{\aleph})} G(s)(\boxed{r}) \rightarrow_{\mathcal{SC}} s$. That is, every term rewrites to itself in one step. The problem is that the occurrence of the hole in the precontext $G(s)(\Box)$ is a 'fake' one; it disappears if the precontext is reduced to $\mathcal{SC}$-normal form, resulting in a context without holes occurring in it.

2. Let $\rightarrow_{\mathcal{SC}}$ be generated by the term rewriting rules

$$\begin{array}{lcl} F(A)(B)(x) & \rightarrow & F(x)(x)(x) \\ F(x)(A)(B) & \rightarrow & F(x)(x)(x) \\ F(B)(x)(A) & \rightarrow & F(x)(x)(x) \\ F(B)(A)(x) & \rightarrow & F(x)(x)(x) \\ F(x)(B)(A) & \rightarrow & F(x)(x)(x) \\ F(A)(x)(B) & \rightarrow & F(x)(x)(x) \end{array}$$

   based on Gustave's TRS ([Ber78, p. 84]). By standard reasoning $\mathcal{SC}$ is seen to satisfy A1–2. Let $\mathcal{R} =^{\mathrm{def}} \{A \rightarrow B\}$. We have the step $F(A)(A)(A) \rightarrow_{\mathcal{H}} F(B)(B)(B)$, by taking the non-linear context $C\Box =^{\mathrm{def}} F(\Box)(\Box)(\Box)$. If one is just allowed to take linear contexts, one gets nowhere: $F(A)(A)(A) \rightarrow_{\mathcal{H}} F(A)(A)(A)$, by taking the linear context $D\Box =^{\mathrm{def}} F(\Box)(A)(A)$ (or one of its two permutations). Every other expansion of $F(A)(A)(A)$ cannot be written as $D\boxed{A}$ for $D\Box$ a linear context, because $D\Box$ is never in $\mathcal{SC}$-normal form. The point is that if we fill two of the holes of $C\Box$ by A, its $\mathcal{SC}$-normal form does not change when filling the third hole by either A or B. The occurrences of the hole are not independent of each other.

In the next definition, we state conditions sufficient for A3 to hold, the idea being that if these conditions hold, then it is possible to pinpoint some occurrence of the left-hand side, such that if this occurrence is replaced by the right-hand side, one gets closer to the term resulting from the parallel replacement of all the occurrences of the left-hand side by the right-hand side.

DEFINITION 3.1.21. A HORS $\mathcal{H} =^{\mathrm{def}} \langle \mathcal{A}, \mathcal{SC}, \mathcal{R} \rangle$ satisfying A1–2 is *serial* if it satisfies the following two conditions.

1. *(persistent hole)* Every context $C\Box$ in which $\Box$ occurs at least once can be written as a binary context $D\boxed{\ ,\ }$, such that both $D\boxed{l,\ }$ and $D\boxed{l,\ }{\downarrow_{\mathcal{SC}}}$ are linear in $\Box$, for any closed term l. The occurrence of the second hole of $D\boxed{\ ,\ }$ is called a *persistent* occurrence in $C\Box$.

2. *(closure under substitution of rewrites)* for every rewrite $d\Box : C\boxed{l,\ } \rightarrow^{+}_{\mathcal{SC}} D\Box$, where $C\Box$ is a context and l a closed term, filling the hole by a closed term g results in a rewrite again, that is, $d\boxed{g} : C\boxed{l,g} \rightarrow^{+}_{\mathcal{SC}} D\boxed{g}$

PROPOSITION 3.1.22. *A serial HORS satisfies A3.*

Proof Let $s \twoheadleftarrow_{\mathcal{SC}} C\boxed{l} \rightarrow_{C\boxed{\aleph}} C\boxed{r} \twoheadrightarrow_{\mathcal{SC}} t$ for some rule $\aleph =^{\mathrm{def}} l \rightarrow r$ and context $C\square$. We prove by noetherian induction on $(C\boxed{r},m)$, where m is the number of occurrences of $\square$ in $C\square$, ordered by $\rightarrow^{+}_{\mathcal{SC}} \times_{lex} >$, that such a rewrite step can be serialised, that is, simulated by a rewrite consisting of linear steps. By the assumption, we can construct the following diagram

$$
\begin{tikzcd}
C\boxed{l} \arrow[rrrrrr] \arrow[d, phantom, sloped, "\equiv"] & & & & & & C\boxed{r} \arrow[d, phantom, sloped, "\equiv"] \\
D\boxed{l,l} \arrow[dd, "\mathcal{SC}"'] \arrow[dr, "\mathcal{SC}"'] & & & D\boxed{l,r} \arrow[dl, "\mathcal{SC}"] \arrow[dr, "\mathcal{SC}"'] & & & D\boxed{r,r} \arrow[dl, "\mathcal{SC}"] \arrow[dd, "\mathcal{SC}"] \\
& D'\boxed{l} \arrow[r] \arrow[dl, "\mathcal{SC}"] & D'\boxed{r} \arrow[dr, "\mathcal{SC}"'] & & C'\boxed{l} \arrow[r] \arrow[dl, "\mathcal{SC}"] & C'\boxed{r} \arrow[dr, "\mathcal{SC}"'] & \\
s \arrow[rrr] & & & s' \arrow[rrr, dashed, twoheadrightarrow] & & & t
\end{tikzcd}
$$

where $D\boxed{\;,\;}$ is obtained from $C\square$ as in the first assumption, by replacing a persistent occurrence of $\square$ by an occurrence of $\square_2$ and the others by occurrences of $\square_1$, $D'\square$ is the $\mathcal{SC}$-normal form of $D\boxed{l,\;}$ and $C'\square$ is the $\mathcal{SC}$-normal form of $D\boxed{\;,r}$. Because $\square_2$ is persistent in $C\square$, $D'\square$ is linear in $\square$, hence the bottom left square gives a rewrite step in the desired format. The induction hypothesis can be applied to obtain the bottom right trapezium, because either $D\boxed{\;,r} \rightarrow^{+}_{\mathcal{SC}} C'\square$ hence $C\boxed{r} \rightarrow^{+}_{\mathcal{SC}} C'\boxed{r}$, or $D\boxed{\;,r} \equiv C'\square$ hence the number of holes in $C'\square$ is one less than in $C\square$. In the trivial case, when $C\square$ contains no $\square$, we have

$$s \equiv C\boxed{l} \equiv C\boxed{r} \equiv t$$

$\odot$

Remark 3.1.23. The first assumption is satisfied for β-, $\beta\eta$- and $\beta\bar{\eta}$-reduction in simply typed λ-calculus, because the leftmost occurrence of a hole always has a unique descendant along reductions by the standardisation theorem. The second assumption in the proposition is weaker than closure of $\mathcal{SC}$ under substitutions, so it is satisfied by both β- and $\beta\eta$-reduction in simply typed λ-calculus. Moreover, it is satisfied by $\beta\bar{\eta}$-reduction because $\bar{\eta}$-normal forms are closed under substitution and β-reduction.

The proposition can be restated as: for any number of occurrences of some left-hand side of a rule in a term, there exists a serial rewrite, replacing each of the occurrences by the right-hand side of the rule. It uses that there always *exists* an occurrence of the left-hand side which has exactly one *descendant* (in the sense of Section 2.4) when the other occurrences are replaced by their right-hand side and reduced to $\mathcal{SC}$-normal form, a so-called *development* of these. Later on, we put restrictions on left-hand sides such that *all* occurrences of the left-hand side have a unique descendant along a development of the other ones, so-called *linearity* of the left-hand side.

The conditions A1–3 can be considered as non-degeneracy conditions on a HORS. If a HORS $\mathcal{H}$ satisfies these conditions, then for proving confluence of $\langle \mathcal{A}, \mathcal{SC}, \mathcal{R} \rangle$ it suffices to prove confluence of *serial rewrite steps* $\twoheadleftarrow_{\mathcal{SC}} ; \rightarrow'_{\mathcal{R}} ; \twoheadrightarrow_{\mathcal{SC}}$, where $\rightarrow'_{\mathcal{R}}$ consists of the replacement steps of $\rightarrow_{\mathcal{R}}$ for which the precontext is a linear context.

The conditions A1–3 can be considered also as sound- and completeness conditions on a HORS. We will show in Section 3.3 that when viewing an ordinary term rewriting system as a HORS having (simply typed) λ-calculus for substitution calculus, the HORS satisfies A1–3 and the serial rewrite steps correspond exactly to the rewrite steps of the term rewriting system. The conditions then imply that there is no harm in doing for example two consecutive replacement steps starting from some preterm, instead of first reducing the intermediate result to a term (see Example 3.1.9).

From all this we conclude that any 'sensible' HORS should satisfy the conditions A1–3 and in the sequel we will assume that each HORS is sensible.

3.1.1. Associating an RRS to a HORS

Having presented the minimal conditions A1–3 a HORS should satisfy in the preceding subsection, we continue our quest for minimal assumptions assuring orthogonality of a HORS in this subsection. This is done by associating a residual rewriting system in the sense of Section 2.4 to a HORS in a natural manner.

How should one associate a residual rewriting system to a HORS? Our plan de campagne is as follows:

1. Residuals are descendants of rewrite steps along other rewrite steps. Remembering that a rewrite step looks like: $s \twoheadleftarrow_{\mathcal{SC}} C\boxed{l} \rightarrow_{C\boxed{\aleph}} C\boxed{r} \twoheadrightarrow_{\mathcal{SC}} t$, i.e. it consists of an expansion, a replacement and a reduction, it is natural to define the descendant relation induced by a rewrite step as the (relation) composition of the descendant relations induced by these three components.

2. Once we have established the descendant relation *induced by* a rewrite step, we should determine what the descendant *of* a rewrite step is. The rewrite step issuing from the term s captures part of the *dynamics* of the term. Although nothing prevents us from having the step itself as a position in a descendant rewriting system, it is much simpler if it is *statically* represented by some *occurrence of a symbol* in the term s. The residual of a rewrite step is then defined via the descendant of this symbol. One can think of such symbols as *representing* the rewrite steps which are possible from a term inside the term itself. In effect, we have reduced the problem of defining residuals of rewrite steps to the problem of defining descendants of occurrences of symbols.

3. The intuitive meaning of an expansion $s \twoheadleftarrow_{\mathcal{SC}} C\boxed{l}$ is as an *extraction* of l from s. Dually, there should be some remnants of l in s along the

reduction $C\boxed{l} \twoheadrightarrow_{\mathcal{SC}} s$. If some (occurrence of an) operator symbol in l has a remnant (i.e. descendant) in $C\boxed{l}{\downarrow_{\mathcal{SC}}}$ for any context $C\Box$, then such a symbol can be used as a representative of a rewrite step generated by the rule $\aleph =^{\text{def}} l \to \ldots$ (there might be several suitable symbols). We pick any such occurrence and call this the *head symbol* of the rule $\aleph =^{\text{def}} l \to r$.

4. Because both the expansion and the reduction take place in the substitution calculus, it is natural to require the substitution calculus to be a descendant rewriting system on occurrences of symbols in $\mathcal{A}_{\mathcal{R}}$.

5. For the replacement step, one has a lot of freedom in associating a descendant relation to it. We take the easiest way out, as indicated in Remark 3.1.8, viewing a replacement step $C\boxed{l} \to_{C\boxed{\aleph}} C\boxed{r}$ as $C\boxed{l} \to_{C\Box} C\Box \leftarrow_{C\Box} C\boxed{r}$. The descendant relation associated to such a step is the identity on all (occurrences of) symbols in $C\Box$, erases all symbols in l, and creates all symbols in r.

Let's start campaigning. First, according to the fourth item, the substitution calculus should be a DRS (see Definition 2.4.1). The objects of the DRS are the preterms, $\mathbb{PT}(\mathcal{A})$, over the alphabet of the HORS and for each preterm s, the positions $\mathcal{R}\mathsf{Pos}(s)$ of symbols in $\mathcal{A}_{\mathcal{R}}$ are traced.

A 4. ($\mathcal{SC}$ is a DRS) *The substitution calculus of the HORS* $\langle \mathcal{A}, \mathcal{SC}, \mathcal{R}\rangle$ *is a DRS:* $\mathcal{SC} =^{\text{def}} \langle \mathbb{PT}(\mathcal{A}), I, \longrightarrow, \mathcal{R}\mathsf{Pos}, \lfloor\cdot\rfloor\rangle$.

REMARK 3.1.24. Only occurrences of symbols in the rewrite alphabet $\mathcal{A}_{\mathcal{R}}$, not occurrences of symbols in the substitution alphabet $\mathcal{A}_{\mathcal{SC}}$, are traced. This makes sense, because rewriting in the HORS is performed modulo $\mathcal{SC}$. For example, if $\mathcal{SC}$ is $\lambda^{\to}_{\bar{\eta}}$, the positions $\mathcal{R}\mathsf{Pos}(s)$ *do not* include the positions of: the application or abstraction symbols, the bound variables, or the positions of β and $\bar{\eta}$-redexes. They *do* include: the positions of the free variables, the positions of holes and the positions of operator symbols. Typically, the positions $\mathcal{R}\mathsf{Pos}(s)$ are such that the 'name' of the symbol at that position does not matter for the substitution calculus, that is, $\mathcal{SC}$ is 'parametric' in these symbols.

The descendant relation of a rewrite step is now defined as sketched above.

DEFINITION 3.1.25. Let $\mathcal{H} =^{\text{def}} \langle \mathcal{A}, \mathcal{SC}, \mathcal{R}\rangle$ be a HORS satisfying A1–4. Let $\mathcal{SC} =^{\text{def}} \langle \mathbb{PT}(\mathcal{A}), I', \longrightarrow, \mathcal{R}\mathsf{Pos}, \lfloor\cdot\rfloor\rangle$.

1. The domain of the trace function $\lfloor\cdot\rfloor$ is extended from $\mathcal{SC}$- to $\mathcal{R}$-steps in the following way. For an $\mathcal{R}$-step $u =^{\text{def}} C\boxed{l} \to_{C\boxed{\aleph}} C\boxed{r}$, where $\aleph =^{\text{def}} l \to r$ and $C\Box$ is a *precontext*, we define $\phi\lfloor u\rfloor\phi =^{\text{def}} \phi \not\geq \Box\mathsf{Pos}(C\Box)$. In words, every position which is not below the position of the hole in the precontext, is related to itself. Stated differently, each position in the context part is left unchanged by a replacement step.

2. The descendant relation system associated to $\mathcal{H}$ is defined by $\mathcal{H} =^{\mathrm{def}} \langle \mathbb{T}(\mathcal{A}), I, \longrightarrow, \mathcal{R}\mathsf{Pos}, \lfloor\cdot\rfloor \rangle$. Steps are labelled by triples expansion, replacement, reduction in I. The descendant relation $\lfloor\cdot\rfloor$ is defined via the descendant relation $\lfloor\cdot\rfloor$ as defined for $\mathcal{SC}$- and $\mathcal{R}$-steps as follows. Let $\aleph =^{\mathrm{def}} l \to r$ be a rule of $\mathcal{H}$. Consider a rewrite step $s \to_{\mathcal{H}} t$ consisting of an expansion $e\colon s \twoheadleftarrow_{\mathcal{SC}} C\boxed{l}$, a replacement step $u =^{\mathrm{def}} C\boxed{l} \to_{C\boxed{\aleph}} C\boxed{r}$, and a reduction $d\colon C\boxed{r} \twoheadrightarrow_{\mathcal{SC}} t$. The descendant relation *induced* by this step is $\lfloor e\,;u\,;d \rfloor =^{\mathrm{def}} \lfloor e \rfloor\,;\lfloor u \rfloor\,;\lfloor d \rfloor$.

In the following, it will always be clear from the mathematical context whether the domain of the trace function $\lfloor\cdot\rfloor$ consists of substitution calculus steps, or replacement steps or rewrite steps.

Remark 3.1.26. In the case of $\lambda_{\overline{\eta}}^{\rightarrow}$ as substitution calculus, the above definition is similar to *origin tracking* for higher-order term rewriting systems as studied by Van Deursen and Dinesh in [DD93]. In origin tracking, *subterms* of a term are traced. Our setup via DRSs is somewhat more general, allowing to trace any property instead of just subterms, but our applications will be more specialised, tracing those subterms which constitute redexes, not any subterm.

The above descendant relation only relates occurrences of $\mathcal{A}_{\mathcal{R}}$-symbols in s to occurrences of $\mathcal{A}_{\mathcal{R}}$-symbols in t, for a rewrite step $s \to_{\mathcal{H}} t$. In order to prove a confluence by orthogonality result, also residuals in t of rewrite steps issuing from s should be defined. Traditionally, these are defined via the descendant of the *head* symbol of the left-hand side. Let us attempt a definition along this way.

Definition 3.1.27. Let $\mathcal{H} =^{\mathrm{def}} \langle \mathcal{A}, \mathcal{SC}, \mathcal{R} \rangle$ be a HORS having descendant relation $\lfloor\cdot\rfloor$. Then $\mathcal{H}$ is said to be *parametric*, if $\mathcal{SC}$ is parametric (in the sense of Definition 2.4.7). That is, for every two $\mathcal{SC}$-reductions d, d' from s to normal form t, we have $\lfloor d \rfloor = \lfloor d' \rfloor$.

Definition 3.1.28. Let $\mathcal{H} =^{\mathrm{def}} \langle \mathcal{A}, \mathcal{SC}, \mathcal{R} \rangle$ be a HORS, and let its associated DRS be $\langle \mathbb{T}(\mathcal{A}), I, \longrightarrow, \mathcal{R}\mathsf{Pos}, \lfloor\cdot\rfloor \rangle$. The HORS $\mathcal{H}$ is called *head-defined*, if there exists a function head mapping each left-hand side l (and rule), to some position $\phi \in \mathcal{R}\mathsf{Pos}(l)$ called its *head position*. This head position must determine a rewrite step in the following way:

1. Let $C\Box$ be a linear context with hole-position ψ (i.e. $\psi =^{\mathrm{def}} \Box\mathsf{Pos}(C\Box)$). Then $\psi\,;\phi$ has a unique descendant χ along any reduction from $C\boxed{l}$ to its normal form $C\boxed{l}{\downarrow_{\mathcal{SC}}}$.

2. For a position χ in a term s, there is at most one linear context $C\Box$ with hole position ψ, such that $\psi\,;\phi$ is an origin of χ along an expansion from s to $C\boxed{l}$. If this exists, the pair $(\chi,\aleph)$ is called a *redex* in s and any such expansion is called an *extraction of* $(\chi,\aleph)$ *from* s (*into* $C\Box$).

In the sequel we often confuse a redex with the rewrite step it induces.

A 5. (extraction) *The HORS is parametric and head-defined.*

REMARK 3.1.29. A rewrite rule can be viewed as giving an (operational) semantics to its head symbol, so its head symbol is also referred to in the literature as the *defined* symbol of the rule.

The conditions in the definition of a head-defined rule intuitively say that there is at most one way in which a left-hand side can be 'seen' at a certain position in a term and the result of rewriting a redex is unique. Later on, we will see that TRS rules (Section 3.3) are head-defined because their left-hand side is (usually) required to be distinct from a variable, CRSs rules (Section 3.4) are head-defined because their left-hand side is required to be of the form $\mathtt{F}(s_1)\dots(s_m)$, and HRSs rules (Section 3.5) are head-defined because their left-hand side is required to be a *pattern*.

EXAMPLE 3.1.30. Consider a HORS having some λ-calculus as substitution calculus. By the standardisation theorem, we can choose $\mathtt{A}$ as head-symbol of each of the terms: $\mathtt{A}$, $\xi.\mathtt{A}$, $\xi.\mathtt{A}(\xi)$, and $\xi.\mathtt{A}(\xi)(\xi)$. The term $\xi.\zeta.\xi(\zeta)$ does not have a head symbol.

Note that by parametricity of $\mathcal{SC}$ the descendant of the position $\psi\,;\phi$ in the definition of extraction does not depend on the actual $\mathcal{SC}$-reduction from $C\boxed{l}$ to $\mathcal{SC}$-normal form. This makes that we can define the residual relation system associated to a HORS in the following way (cf. Definition 3.1.25 where the descendant relation system associated to a HORS was defined),

DEFINITION 3.1.31. Let $\mathcal{H} =^{\mathrm{def}} \langle \mathcal{A}, \mathcal{SC}, \mathcal{R}\rangle$ and $\langle \mathbb{T}(\mathcal{A}), I', \rightarrow', \mathcal{R}\mathsf{Pos}', \lfloor\cdot\rfloor'\rangle$ be its associated DRS, satisfying A1–5. The residual rewriting system *associated to* $\mathcal{H}$ is defined to be $\langle \mathbb{T}(\mathcal{A}), I, \rightarrow, \mathcal{R}\mathsf{Pos}, \lfloor\cdot\rfloor\rangle$, in which

1. the set of rewrite step labels is $I =^{\mathrm{def}} \mathsf{Pos}\times\mathcal{R}$, that is, steps are labelled by redexes,
2. the function $\mathcal{R}\mathsf{Pos}$ maps a term s to the occurrences in s, just like $\mathcal{R}\mathsf{Pos}'$, but additionally to the set of redexes in it,
3. the rewrite step relation is defined by $u =^{\mathrm{def}} s \rightarrow_{(\phi,\aleph)} t$, if $u' =^{\mathrm{def}} s \rightarrow'_{e;v;d} t$ in the DRS, for some $e\colon s \twoheadleftarrow_{\mathcal{SC}} C\boxed{l}$, $v =^{\mathrm{def}} C\boxed{l} \rightarrow_{C\boxed{\aleph}} C\boxed{r}$, and $d\colon C\boxed{r} \twoheadrightarrow_{\mathcal{SC}} t$, we have the rewrite step $u' =^{\mathrm{def}} s \rightarrow'_{e;v;d} t$ in the DRS, and e extracts $(\phi,\aleph)$ from s into $C\Box$. We then define $\lfloor u\rfloor =^{\mathrm{def}} \lfloor u'\rfloor'$,
4. for u either a substitution step, or a replacement step or rewrite step, the descendant of a redex is defined via the position of its head-symbol: $(\phi,\aleph)\lfloor u\rfloor(\psi,\aleph) =^{\mathrm{def}} \phi\lfloor u\rfloor\psi$.

Note that an $\aleph$-redex can only have $\aleph$-redexes as descendants.

3.1.2. Orthogonal HORSs

In the preceding subsection we have managed to associate a residual rewriting system to every HORS satisfying A1–5. We assume all HORSs in this subsection to satisfy these conditions.

Via the RRS associated to a HORS, as defined in the preceding subsection, the notion of orthogonality is directly defined for it. The topic of this subsection is to find sufficient conditions to ensure orthogonality. For an RRS to be orthogonal, three conditions must be met: consistency, finiteness of developments, and parametricity (see page 48). We will consider these three conditions in turn.

Consistency of a set of redexes roughly requires that every development (see page 45) of the set of redexes ends in the same result. This is a kind of forward notion, i.e. one determines what can happen in the future to a set of redexes, comparable to 'interleaving concurrency'. We will require something stronger ensuring consistency: *simultaneity* of the redexes, that is, a set of redexes is simultaneous in a term if the whole set can be extracted from the term. This is a kind of backward notion, i.e. one determines which redexes could have originated from completely disjoint parts, comparable to 'true concurrency'. Moreover, we require that extracting a redex from a term neither duplicates nor erases the other parts of a term; so-called *linearity* of the redexes.

To get an idea of these two notions, we give an example. The terminology used to explain the reasons of (non-)simultaneity and (non-)linearity, is the common one used in (higher-order) term rewriting, see e.g. Section 3.3.

EXAMPLE 3.1.32. Let the substitution calculus be $\lambda_{\overline{\eta}}^{\rightarrow}$ with the usual descendant relation.

1. Let $\aleph =^{\mathrm{def}} \xi.\mathtt{F}(\xi)(\xi) \to \ldots$ be some rule, with head symbol $\mathtt{F}$ at head position 000. Consider the term $s =^{\mathrm{def}} \mathtt{F}(\mathtt{F}(\mathtt{A})(\mathtt{A}))(\mathtt{F}(\mathtt{A})(\mathtt{A}))$. There exist three extractions of $\aleph$ from s.

   (a) An extraction of $(00,\aleph)$ into the context $\Box(\mathtt{F}(\mathtt{A})(\mathtt{A}))$,

   (b) an extraction of $(0100,\aleph)$ into the context $\mathtt{F}(\Box(\mathtt{A}))(\mathtt{F}(\mathtt{A})(\mathtt{A}))$, and

   (c) an extraction of $(100,\aleph)$ into the context $\mathtt{F}(\mathtt{F}(\mathtt{A})(\mathtt{A}))(\Box(\mathtt{A}))$.

   The first extraction is not simultaneous with the other ones. By trying out some expansions the reader will be convinced of this. The reason of the failure is the *'non-linearity'* of the left-hand side $\xi.\mathtt{F}(\xi)(\xi)$, combined with the *'nesting'* of the redexes. The second and third extraction are simultaneous, because they can be extracted into the context $\mathtt{F}(\Box_1(\mathtt{A}))(\Box_2(\mathtt{A}))$.

2. Let $\aleph =^{\mathrm{def}} \mathtt{F}(\mathtt{A}) \to \ldots$ be a rule with head symbol $\mathtt{F}$ at head position 0, and $\beth =^{\mathrm{def}} \mathtt{A} \to \ldots$ be a rule with head symbol $\mathtt{A}$ at head position ε. Consider the term $s =^{\mathrm{def}} \mathtt{F}(\mathtt{A})$. From s there exist two (trivial) extractions.

(a) An extraction of $(0,\aleph)$ into the context $\square$, and

(b) an extraction of $(1,\beth)$ into the context $\mathtt{F}(\square)$.

There does not exists a simultaneous extraction of both redexes into some context $C\boxed{\ ,\ }$. This follows directly from the context being in normal form. The reason for this failure is the *'overlap'* of the left-hand sides $\mathtt{A}$ and $\mathtt{F}(\mathtt{A})$.

3. Let $\aleph =^{\mathrm{def}} \xi.\mathtt{F}(\xi) \to \dots$ with head symbol $\mathtt{F}$ at head position 00, and $\beth =^{\mathrm{def}} \mathtt{A} \to \dots$ with head symbol $\mathtt{A}$ at head position ε. Consider the term $s =^{\mathrm{def}} \mathtt{F}(\mathtt{A})$. From s there exist two extractions.

   (a) An extraction of $(0,\aleph)$ into the context $\square(\mathtt{A})$, and

   (b) a trivial extraction of $(1,\beth)$ into the context $\mathtt{F}(\square)$.

   There exist a simultaneous extraction of these redexes into the context $\square_1(\square_2)$. In fact, any number of $\aleph$- and $\beth$-redexes in a term can be extracted simultaneously. The reason for this is that their left-hand sides $\mathtt{A}$ and $\xi.\mathtt{F}(\xi)$ are *'non-overlapping, linear patterns'*.

Summing up, we have that simultaneity of redexes in a term captures that one can abstract over their left-hand sides in the term by replacing these by variables (thus hiding their structure), and linearity expresses that in the abstraction process other occurrences are not erased or duplicated. We first generalise the definition of extraction from redexes to sets of redexes.

DEFINITION 3.1.33. Let $\mathcal{H}$ be a HORS. Let $\mathcal{U} =^{\mathrm{def}} \{u_1, \dots, u_m\}$ be a set of (pairwise distinct) redexes in a term s, where for $u_n \in \mathcal{U}$, $u_n =^{\mathrm{def}} (\chi_n,\aleph_n)$. An expansion $e\colon s \twoheadleftarrow_{\mathcal{SC}} C\boxed{l_1,\dots,l_m}$ is called an *extraction of $\mathcal{U}$ from s (into $C\boxed{m}$)*, if $C\boxed{m}$ is a linear context and for every $u_n \in \mathcal{U}$, we have that $\square_n\mathsf{Pos}(C\boxed{m});\mathsf{head}(l_n)$ is an origin of χ_n. In words this reads, the concatenation of the position of the n-th hole in $C\boxed{m}$ with the head position of the n-th rule is an origin of the position of the n-th redex. If such an extraction exists for the set $\mathcal{U}$, then $\mathcal{U}$ is called a *simultaneous* set of redexes.

Note that the order of the redexes in the set $\mathcal{U}$ is not important in the above definition, because the order of the holes in a context is not important.

Next, we formalise the notion of linearity of a set of closed terms. This is done by looking what happens if some of these are put into a context and subsequently reduced to normal form (in the substitution calculus).

DEFINITION 3.1.34. Let $\mathcal{H} =^{\mathrm{def}} \langle \mathcal{A}, \mathcal{SC}, \mathcal{R} \rangle$ be a HORS having descendant relation $\lfloor\cdot\rfloor$.

1. An $\mathcal{SC}$-step u is called *linear*, if its descendant relation $\lfloor u \rfloor$ is a bijection. An $\mathcal{SC}$-conversion is *linear* if it is composed of linear $\mathcal{SC}$-steps.

2. A set S of closed terms is *linear*, if for every sequence $s_1, \ldots, s_m$ of terms in S and every m-ary context $C\boxed{m}$, any $\mathcal{SC}$-reduction starting from $C\boxed{s_1, \ldots, s_m}$ is linear.

3. A set of rules is *left-linear* (*right-linear*), if there exists a linear set of terms containing all its left-hand (right-hand) sides. The set is *linear*, if there exists a linear set of terms containing all its left- and right-hand sides. These notions carry over to the HORS $\mathcal{H}$ via its set of rules.

By definition, linear conversions can be concatenated to form linear conversions. The following proposition is proved by induction on the length of a conversion and expresses the idea of linearity: concatenating a linear rewrite with its inverse amounts to nothing.

PROPOSITION 3.1.35. *Let d be a linear conversion. Then* $\lfloor d\,;d^{-1}\rfloor = \lfloor 0 \rfloor$.

A 6. (left-linearity) *The HORS is left-linear.*

Linearity of a term expresses that it never has any duplicating effect on parts in its environment, even in combination with other linear terms. Usually left-linearity of rules is enforced by the condition that in the left-hand side of a rule every occurrence of a free variable is unique. In our framework this does not make sense, because we work with closed rules. In the sections on TRSs and HRSs, we will show that the standard notion of left-linearity for these systems implies the above one. The following example shows that it is not sufficient in general that each of the left-hand sides of the rewrite rules separately has no duplicating effect, in order to ensure that their combination has no duplicating effect.

EXAMPLE 3.1.36. Let $\rightarrow_{\mathcal{SC}}$ be generated by the rules

$$\begin{array}{rcl} F(G(x))(y) & \rightarrow & J(G(x)) \\ F(x)(H(y)) & \rightarrow & J(x) \\ F(G(x))(H(y)) & \rightarrow & F'(x)(x) \\ F'(x)(y) & \rightarrow & J(G(x)) \end{array}$$

By applying some standard rewriting theory $\mathcal{SC}$ is shown to be complete. Let $\mathcal{R} =^{\mathrm{def}} \{G(\mathtt{A}) \rightarrow \ldots, H(\mathtt{B}) \rightarrow \ldots\}$. One easily shows that for every unary context $C\Box$, the $\rightarrow_{\mathcal{SC}}$-rewrites issuing from $C\boxed{G(\mathtt{A})}$ or $C\boxed{H(\mathtt{B})}$ are linear, because filling a hole in a context by either $G(\mathtt{A})$ or $H(\mathtt{B})$ cannot create an $\mathcal{SC}$-redex for the third rule (if it could, the context would contain either a redex for the first or the second rule, which is not allowed) and $\mathcal{SC}$-reduction cannot create $\mathcal{SC}$-redexes. However, from the binary context $C\boxed{\ ,\ } =^{\mathrm{def}} F(\Box_1)(\Box_2)$, we have the duplicating step $F(G(\mathtt{A}))(H(\mathtt{B})) \rightarrow_{\mathcal{SC}} F'(\mathtt{A})(\mathtt{A})$.

In a HORS satisfying A6 every extraction must be linear. The point of the next condition is that it allows to view an extraction of $\mathcal{U}$, where $\mathcal{U} =^{\mathrm{def}} \mathcal{V} \cup \mathcal{W}$, as the composition of an extraction e of $\mathcal{V}$ and an extraction of (the origins of $\mathcal{W}$ along e) $\mathcal{W}$. For this it is needed that the descendant relation behaves 'naturally' when specialising a rewrite step $u\Box\colon C\Box \rightarrow_{\mathcal{SC}} D\Box$ into a rewrite step $u\boxed{l}\colon C\boxed{l} \rightarrow_{\mathcal{SC}} D\boxed{l}$.

DEFINITION 3.1.37. A HORS $\mathcal{H} =^{\mathrm{def}} \langle \mathcal{A}, \mathcal{SC}, \mathcal{R}\rangle$, for which $\lfloor \cdot \rfloor$ is the descendant relation of $\mathcal{SC}$, is said to be *naturally closed under substitutions*, if the following two conditions are satisfied.

1. $\mathcal{SC}$-rewrites of contexts are closed under substitutions (see Definition 3.1.21(2)).
2. let $u\Box\colon C\Box \rightarrow_{\mathcal{SC}} D\Box$ be an $\mathcal{SC}$-step. Then, by the first item, $u\boxed{l}\colon C\boxed{l} \rightarrow_{\mathcal{SC}} D\boxed{l}$ is an $\mathcal{SC}$-step too. The descendant relation induced by $u\boxed{l}$ relates ϕ to ψ, i.e. $\phi\lfloor u\boxed{l}\rfloor\psi$, if and only if
   (a) either $\phi\lfloor u\Box\rfloor\psi$ (positions in the context part are related via $u\Box$),
   (b) or $\phi = \phi';\chi$, $\psi = \psi';\chi$, $\phi' \in \Box\mathsf{Pos}(C\Box)$, $\psi' \in \Box\mathsf{Pos}(D\Box)$, $\chi \in \mathcal{R}\mathsf{Pos}(l)$, and $\phi'\lfloor u\Box\rfloor\psi'$ (positions in the substitution part are related via the position of the hole of the context).

The reason for calling $\mathcal{H}$ *naturally* closed under substitutions are the following two propositions. The first one is immediate from the definitions of naturality and linearity.

PROPOSITION 3.1.38. *Let $\mathcal{H}$ be naturally closed under substitutions. Let $d\Box$ be a linear $\mathcal{SC}$-conversion, then $d\boxed{l}$ is a linear $\mathcal{SC}$-conversion, i.e. substitutions preserve linearity of $\mathcal{SC}$-conversions.*

The next proposition expresses that for HORSs which are naturally closed under substitutions, the descendant relations of substitution steps and replacement steps interact nicely.

PROPOSITION 3.1.39. *Let $\mathcal{H}$ be naturally closed under substitutions. Let $\aleph =^{\mathrm{def}} l \rightarrow r$ be a $\mathcal{H}$-rule, and $u\Box\colon C\Box \rightarrow_{\mathcal{SC}} D\Box$ be an $\mathcal{SC}$-step. Let $d\colon C\boxed{\aleph}\,;\,u\boxed{r}$, $d'\colon u\boxed{l}\,;\,D\boxed{\aleph}$. Then $\lfloor d\rfloor = \lfloor d'\rfloor$ for any precontexts $C\Box$ and $D\Box$, that is, the two ways of chasing the diagram*

$$\begin{tikzcd} C\boxed{l} \arrow[r, "C\boxed{\aleph}", "\mathcal{SC}"'] \arrow[d, "u\boxed{l}"', "\mathcal{R}"] & C\boxed{r} \arrow[d, "u\boxed{r}"', "\mathcal{R}"] \\ D\boxed{l} \arrow[r, "D\boxed{\aleph}", "\mathcal{SC}"'] & D\boxed{r} \end{tikzcd}$$

induce the same descendant relation.

Proof Let $\phi \lfloor d \rfloor \psi$. By definition of $\lfloor C\boxed{\aleph} \rfloor$, we have $\phi \lfloor u\boxed{r} \rfloor \psi$ and $\phi \not\geq \Box\mathsf{Pos}(C\Box)$. Because $\mathcal{H}$ is naturally closed under substitutions, we have $\phi \lfloor u\Box \rfloor \psi$, $\phi \lfloor u\boxed{l} \rfloor \psi$, and $\psi \not\geq \Box\mathsf{Pos}(D\Box)$. By definition of $\lfloor D\boxed{\aleph} \rfloor$, we have $\phi \lfloor d' \rfloor \psi$. $\odot$

A 7. (naturality) *The HORS is naturally closed under substitution.*

We now can prove the key lemma needed for the Finite Developments theorem.

Lemma 3.1.40. *Let $\mathcal{H}$ be a HORS satisfying A1–7. Let $u =^{\mathrm{def}} s \to_{(\phi,\aleph)} t$ be a rewrite step. Hence there exists an extraction $e\colon s \twoheadleftarrow_{\mathcal{SC}} C\boxed{l}$ of $(\phi,\aleph)$ into context $C\Box$. Let $\mathcal{U} =^{\mathrm{def}} \mathcal{V} \cup \{u\}$ be a set of simultaneous redexes in s. Then the set $\mathcal{V}' =^{\mathrm{def}} \{\mathcal{V}\ \lfloor e \rfloor\}$ is a set of simultaneous redexes in $C\Box$. The situation is described in the following diagram*

$$\begin{tikzcd}
D\boxed{L,l} \arrow[dd, "g"'] \arrow[dr, "h\boxed{l}"'] & \\
& C\boxed{l} \arrow[dl, "e"] \\
s &
\end{tikzcd}$$

where L is the set of left-hand sides of $\mathcal{V}$.

Proof By $\mathcal{U}$ being simultaneous in s, there exists an extraction g of $\mathcal{U}$ from s into context $D\Box$, as in the diagram. Define $h\Box$ to be an $\mathcal{SC}$-expansion from $C'\Box =^{\mathrm{def}} D\boxed{L,\ }\downarrow_{\mathcal{SC}}$ to $D\boxed{L,\ }$.

Claim $C\Box \equiv C'\Box$.

Proof of claim By linearity of L, $h\Box$ must be a linear expansion. By natural closure under substitution, we can construct the linear expansion $h\boxed{l}$ from $C'\boxed{l}$ to $D\boxed{L,l}$. By completeness of $\mathcal{SC}$ and because $s \,{}^{!}\!\twoheadleftarrow_{\mathcal{SC}} D\boxed{L,l}$, there exists an expansion from e' from s to $C'\boxed{l}$, which is linear by linearity of l. By natural closure under substitution, the descendant relation induced by the linear expansion $h\boxed{l}$ must relate the head-position of l inside the context $C'\Box$ to the head-position, say ψ, of l in the context $D\boxed{L,\ }$. By parametricity, the linear expansions g and $e'\ ; h\boxed{l}$ must induce the same descendant relation, hence $\lfloor e' \rfloor$ must relate ϕ to ψ. We conclude that e' is an extraction of $(\phi,\aleph)$ from s into $C'\Box$. By head-definedness of the HORS, all extractions of a redex extract into the same context, that is, $C\Box \equiv D\Box$. $\odot$

Similarly, one shows that the head positions of the redexes in $\mathcal{V}'$ must be related by $\lfloor h\boxed{l} \rfloor$ to the head-positions of the set L inside the context $D\boxed{\ ,l}$. We conclude that $h\Box$ is an extraction of $\mathcal{V}'$ from $C\Box$ into the context $D\boxed{\ ,\ }$, i.e. the set $\mathcal{V}'$ is simultaneous in $C\Box$. $\odot$

From the lemma, it follows that the redexes in a simultaneous set of redexes can be extracted in any order. Since extraction of one redex uniquely determines the context into which it is extracted (by head-definedness), the context into which a set of redexes is extracted is unique too, by the lemma. The lemma is the key ingredient needed in our proof of the Finite Developments theorem.

REMARK 3.1.41. Actually, the context into which a set of simultaneous redexes is extracted is only unique up to some renaming of holes. For example, consider the rule $\aleph =^{\text{def}} \mathtt{A} \to \ldots$ and the term $\mathtt{F}(\mathtt{A})(\mathtt{A})$. The $\aleph$-redexes at positions 01 and 1 can be simultaneously extracted into the contexts $\mathtt{F}(\Box_1)(\Box_2)$ and $\mathtt{F}(\Box_2)(\Box_1)$. We will not be bothered by this imprecision.

Our strategy for proving FD consists of the following three parts. Let $\mathcal{U} =^{\text{def}} \mathcal{V} \cup \{u\}$ be some set of simultaneous redexes in a term, with sets L and R of left- and right-hand sides.

1. First, we prove that the rewrite step u can be simulated by a '$\mathcal{V}$-abstracted rewrite step', that is, a rewrite step in which we have abstracted over the redexes in $\mathcal{V}$, by replacing these by variables. This we call the *Envelop Lemma.*

2. Then, we give a measure on 'abstracted rewrite steps' and show that this measure decreases in some well-founded order along a development of $\mathcal{U}$. Hence, every development of $\mathcal{U}$ must be finite. This we call the *Develop Lemma.*

3. Finally, combining the Envelop lemma with the Develop Lemma, we show that every complete development of $\mathcal{U}$ from s to t can be simulated by a simultaneous extraction of $\mathcal{U}$ from s into some context $C\Box$, followed by a sequence of replacement steps from $C\boxed{L}$ to $C\boxed{R}$, followed by a reduction to t. This is the *Finite Developments theorem.*

LEMMA 3.1.42. (Envelop Lemma) *Let $\mathcal{H}$ satisfy A1–7. Let $\mathcal{U} =^{\text{def}} \mathcal{V} \cup \{u\}$ be some set of simultaneous redexes in a term, with sets L and R of left- and right-hand sides. Let $u =^{\text{def}} s \to_{(\phi,\aleph)} t$ Then u is* simulated *by the $\mathcal{V}$*-abstracted rewrite step *$u' =^{\text{def}} g\, ;\, D\boxed{L,\aleph}\, ;\, f$, where g is an extraction of $\mathcal{V}, \aleph$ from s into context $D\boxed{\ ,\ }$ and f is a reduction from $D\boxed{L,r}$ to t. By u being simulated by u', it is meant that $\lfloor u \rfloor = \lfloor u' \rfloor$.*

PROOF Let u be composed of the extraction $e\colon s \twoheadleftarrow_{SC} C\boxed{l}$, the replacement $C\boxed{\aleph} =^{\text{def}} C\boxed{l} \to C\boxed{r}$, and the reduction $d\colon C\boxed{r} \twoheadrightarrow_{SC} t$. The proof of the lemma is expressed by the following envelope

$$\begin{tikzcd}
D\boxed{L,l} \arrow[rrr, "D\boxed{L,\aleph}"'] \arrow[dr, "h\boxed{l}"'] \arrow[dd, "g"'] & & & D\boxed{L,r} \arrow[dl, "h\boxed{r}"] \arrow[dd, "f"] \\
\arrow[r, phantom, "1"] & C\boxed{l} \arrow[r, "C\boxed{\aleph}"'] \arrow[dl, "e"] \arrow[ur, phantom, "2"] & C\boxed{r} \arrow[dr, "d"'] \arrow[r, phantom, "3"] & \\
s \arrow[rrr, "\mathcal{H}"'] & & & t
\end{tikzcd}$$

By simultaneity of $\mathcal{U}$, we can construct the extraction g. By the key lemma, we can construct the linear expansion $h\boxed{l}: C\boxed{l} \twoheadleftarrow_{\mathcal{SC}} D\boxed{L,l}$, and the linear reduction $h\boxed{r}: D\boxed{L,r} \twoheadrightarrow_{\mathcal{SC}} C\boxed{r}$ (note that we define $h\boxed{l}$ to be an expansion, while $h\boxed{r}$ is defined to be a reduction). The only thing which remains to be shown is that the path on the outside of the diagram simulates the one on the inside, that is, $\lfloor g\,;D\boxed{L,\aleph}\,;f\rfloor = \lfloor e\,;C\boxed{\aleph}\,;d\rfloor$. This is shown in three steps, corresponding to the numbers in the diagram.

1. $\lfloor g\rfloor\,;\lfloor h\boxed{l}\rfloor^{-1} = \lfloor e\rfloor\,;\lfloor h\boxed{l}\rfloor\,;\lfloor h\boxed{l}\rfloor^{-1} = \lfloor e\rfloor$. This is true by parametricity and linearity of $h\boxed{l}$.
2. $\lfloor C\boxed{\aleph}\rfloor = \lfloor h\boxed{l}\rfloor;\lfloor h\boxed{l}\rfloor^{-1};\lfloor C\boxed{\aleph}\rfloor = \lfloor h\boxed{l}\rfloor;\lfloor D\boxed{L,\aleph}\rfloor;\lfloor h\boxed{r}\rfloor$. This is true by Proposition 3.1.39 and by linearity of $h\boxed{l}$.
3. From parametricity and linearity of $h\boxed{r}$, we have $\lfloor h\boxed{r}\rfloor^{-1}\,;\lfloor f\rfloor = \lfloor h\boxed{r}\rfloor^{-1}\,;\lfloor h\boxed{r}\rfloor\,;\lfloor d\rfloor = \lfloor d\rfloor$.

We can compose these three observations to obtain the desired result, using linearity of $h\boxed{l}$ and $h\boxed{r}$. $\odot$

The envelop lemma only expresses that for a single step one can abstract over the other redexes in a term. In order to conclude something about developments, i.e. sequences of rewrite steps, one should be able somehow to link these observations about single steps.

LEMMA 3.1.43. (Develop Lemma) *Let $\mathcal{H}$ satisfy A1–7, then every development of a simultaneous set of redexes is finite.*

PROOF Let $\mathcal{H} =^{\mathrm{def}} \langle\mathcal{A},\mathcal{SC},\mathcal{R}\rangle$ be a HORS with descendant relation $\lfloor\cdot\rfloor$. Let $\langle B, J, \rightsquigarrow\rangle$ be the licensing LRS associated to $\mathcal{H}$ (via its RRS). By definition of development (Definition 2.4.3), we have to prove that every rewrite in the licensing LRS is finite. Choose the variables as in the Envelop Lemma and let $(s,\mathcal{U}) \rightsquigarrow_u (s',\mathcal{U}')$ be a step in the LRS. We will construct an extraction $g': s' \twoheadleftarrow_{\mathcal{SC}} D'\boxed{L'}$ of $\mathcal{U}'$ from s', such that $(D\boxed{R,r},n) \rightarrow^{+}_{\mathcal{SC}} \times_{lex} > (D'\boxed{R'},n')$, where n, n' are the number of holes in $D\Box$ and $D'\Box$. The idea is the same as in Proposition 3.1.22, but the ordering is extended from unary to arbitrary contexts. The construction of g' is shown in the following diagram

$$\begin{tikzcd} & D\boxed{L,r} \arrow[dl, "h\boxed{r}"'] \arrow[dr, "h'\boxed{L}"] & \\ C\boxed{r} \arrow[dr, "d"'] & & E\boxed{L} \equiv D'\boxed{L'} \arrow[dl, "g'"] \\ & s' & \end{tikzcd}$$

Here $h'\Box$ is a reduction from $D\boxed{\ ,r}$ to its $\mathcal{SC}$-normal form $E\Box =^{\mathrm{def}} D\boxed{\ ,r}{\downarrow_{\mathcal{SC}}}$, reducing the redexes created by plugging in the right-hand side r in the context $D\boxed{\ ,\ }$. Because r might be non-linear (only left-hand sides were required to be linear), $E\Box$ might be a non-linear context. Now take $D'\Box$ to be the *linearisation* of $E\Box$, i.e. a linear context such that the positions of the holes in $E\Box$ and $D'\Box$ are the same, hence $E\boxed{L} \equiv D'\boxed{L'}$ for some appropriate L'. By closure of reduction under substitution, the reduction $h'\boxed{L}$ can be constructed and by completeness there exists an expansion g' from s' to $E\boxed{L} \equiv D'\boxed{L'}$.

Claim The expansion g' is an extraction of $\mathcal{U}'$ from s' into $D'\Box$.

Proof of claim 1. First, g' is linear, because the elements of L are elements of L, hence linear by assumption.

2. Next, by linearity of g' it is clear that for every head-position χ' of a left-hand side l' at position ω' in context $D'\Box$, $\omega' ; \chi'$ is an origin of some redex $(\psi',\beth)$ in s'.

3. Using natural closure of reductions under substitutions for $h'\Box$ we easily obtain that $\omega' ; \chi'$ must be the descendant of some redex in $\mathcal{U}$ along $g;D\boxed{L,\aleph};h'\boxed{L}$, moreover, every redex in $\mathcal{U}$ descends only to such positions. Hence we obtain from the Envelop Lemma, that the redex $(\psi',\beth)$ in the second item is a residual of a redex in $\mathcal{U}$, i.e. g' is a an extraction of a subset of the residuals of $\mathcal{U}$ in s'.

4. Finally, because every redex $(\psi',\beth)$ in $\mathcal{U}'$ descends from some redex $(\psi,\beth)$ in $\mathcal{U}$, we must have by the third item that it originates from some position $\omega' ; \chi'$ as above, that is, g' is an extraction of a superset of the residuals of $\mathcal{U}$ in s'.

The last two items together with linearity make that g' is indeed the desired extraction. $\odot$

Now, in order to prove that the extraction g' is smaller than the extraction g we remark that by closure of reduction under substitutions, we have the $\mathcal{SC}$-reduction $h'\boxed{R}$: $D\boxed{R,r} \twoheadrightarrow_{\mathcal{SC}} E\boxed{R} \equiv D'\boxed{R'}$. The extraction g' can only be not smaller than g if $h'\boxed{R}$ is an empty reduction, but then $D'\Box \equiv D\boxed{\ ,r}$ which has one hole less than $D\boxed{\ ,\ }$. $\odot$

The Finite Developments theorem (Theorem 3.1.45) will be based on the following proposition. It expresses that replacement steps at distinct positions in any precontext can be performed in any order (or even in parallel) as far as the descendant relation is concerned. In the Finite Developments theorem the 'independence' of redexes is reduced to this independence of replacement steps.

PROPOSITION 3.1.44. *Let* $C\boxed{m}$ *be a context and* $l_1 \to r_1, \ldots, l_m \to r_m$ *be a sequence of rewrite rules. Then every sequence of steps*

$$C\boxed{l_1,\ldots,l_m} \to_{C\boxed{\ldots,l_{n_1}\to r_{n_1},\ldots}} \cdots \to_{C\boxed{\ldots,l_{n_m}\to r_{n_m},\ldots}} t$$

where $n_1,\ldots,n_m$ *are pairwise distinct, ends in the preterm* $C\boxed{r_1,\ldots,r_m}$ *and induces the same descendant relation.*

PROOF By induction on m, showing that the induced descendant relation is the identity relation on positions in the context part $C\boxed{m}$. ⊙

THEOREM 3.1.45. (Finite Developments) *Let* $\mathcal{H}$ *be a HORS satisfying A1–7. Then every complete development of a set of simultaneous redexes ends in the same term and induces the same descendant relation.*

PROOF Using the Envelop and Develop lemmata (and the notations introduced in their proofs), we can construct the following diagram.

$$\begin{tikzcd}
D\boxed{L,l} \arrow[rrr, "D\boxed{L,\aleph}"'] \arrow[ddd, "g"'] \arrow[dr, "h\boxed{l}"] & & & D\boxed{L,r} \arrow[rrr, twoheadrightarrow, "D\boxed{L\to R,r}"'] \arrow[dl, "h\boxed{r}"] \arrow[dr, "h'\boxed{L}"] & & & D\boxed{R,r} \arrow[ddd, "f"] \arrow[dl, "h'\boxed{R}"'] \\
& C\boxed{l} \arrow[r, "C\boxed{\aleph}"'] \arrow[ddl, "e"] & C\boxed{r} \arrow[ddr, "d"'] & & E\boxed{L} \arrow[r, twoheadrightarrow, "E\boxed{L\to R}"'] \arrow[ddl, "g'"] & E\boxed{R} \arrow[ddr, "f'"'] & \\
& & & & & & \\
s \arrow[rrr, "\mathcal{H}"'] & & & s' \arrow[rrr, twoheadrightarrow, "\mathcal{H}"'] & & & t
\end{tikzcd}$$

CLAIM Every complete $\mathcal{U}$-development ends in the term t, and the descendant relation is the one induced by g ; $D\boxed{L\to R,l\to r}$; f, that is, the one induced by following the 'outside' of the diagram.

PROOF OF CLAIM The proof is by noetherian induction on $(D\boxed{R,r},n)$ ordered by $\to^{+}_{SC} \times_{lex} >$, where n is the number of holes in $D\Box$, obtained by extracting $\mathcal{U}$ from s.

1. If $D\boxed{n}$ is a linear context and $n = 0$, then $D\boxed{n}$ must be nullary hence equal to s and there is nothing to prove.

2. Otherwise, we know by the Envelop Lemma that $g'\colon s' \twoheadleftarrow_{SC} E\boxed{L} \equiv D'\boxed{L'}$ is an extraction of $\mathcal{U}'$ from s' and that we can apply the induction hypothesis, obtaining that every complete $\mathcal{U}'$-development of s' ends in t and its descendant relation is induced by the sequence g' ; $D'\boxed{L'\to R'}$; f' which

is the same as the one induced by the sequence $g'\,;E\boxed{L\to R}\,;f'$. By the Develop Lemma we know that the descendant relation induced by u is the same as the one induced by $g\,;D\boxed{L,l\to r}\,;h'\boxed{L}\,;g'$. Composing this sequence with the one supplied by the induction hypothesis and using linearity of g', it remains to be shown that $h'\boxed{L}\,;E\boxed{L\to R}$ and $D\boxed{L\to R,r}\,;h'\boxed{R}$ induce the same descendant relation. This follows from Proposition 3.1.39.

These are the only cases to be considered. $\odot$

To be rigorous, the term r and the induced descendant relation might still depend on the choice of the first rewrite step, but Proposition 3.1.44 shows that this is not so. $\odot$

In the proof essential use is made of termination of the substitution calculus in the construction of the well-founded order needed for the induction. So, the method can be viewed as reducing a proof of finite developments to a proof of termination of the substitution calculus.

A 8. (simultaneity) *The HORS is* simultaneous*, that is, every set of redexes is simultaneous.*

THEOREM 3.1.46. (Confluence by Orthogonality) *A HORS $\mathcal{H}$ satisfying*

A1 the substitution calculus is complete,

A2 the substitution calculus is only needed for gluing,

A3 parallel rewrite steps can be serialised,

A4 the substitution calculus is a descendant rewriting system,

A5 the substitution calculus is parametric and rules are head-defined,

A6 left-hand sides of rules are linear,

A7 the substitution calculus is naturally closed under substitution,

A8 every set of redexes is simultaneous,

is orthogonal, hence confluent.

PROOF Using Theorem 3.1.45, one easily verifies the conditions of orthogonality (Definition 2.4.9). In particular, we see that simultaneity of a set of redexes implies its consistency. By Theorem 2.4.12, remembering that orthogonal rewriting systems are weakly orthogonal, we conclude confluence. $\odot$

The proof shows that simultaneity implies consistency. One might wonder whether the inverse implication holds, that is, whether consistency implies simultaneity. The following example shows that this is not the case. The reason for the failure is that consistency of redexes can arise from a syntactical coincident whereas simultaneity requires them to be really independent.

EXAMPLE 3.1.47. Let $\mathcal{H}$ be a HORS, with as $\mathcal{SC}$ simply typed λ-calculus. Consider the two rules $\aleph$ and $\beth$:

$$\begin{array}{rcl} \mathtt{F}(\mathtt{A}) & \to & \mathtt{A} \\ \xi.\mathtt{F}(\xi) & \to & \mathtt{A} \end{array}$$

It is easy to check that the two redexes $(\varepsilon,\aleph)$ and $(\varepsilon,\beth)$ in the term $\mathtt{F}(\mathtt{A})$ are consistent, but not simultaneous.

The standard results in Section 2.4 allow to obtain a lot more information from orthogonality then just confluence.

The Finite Developments theorem holds for any set of simultaneous redexes in a term, but how can we guarantee a set of redexes to be simultaneous? In the next proposition we give a criterion on pairs of redexes ensuring simultaneity of any set of redexes.

DEFINITION 3.1.48. A HORS $\mathcal{H}$ is *pairwise simultaneous* if every pair of distinct redexes is simultaneous.

PROPOSITION 3.1.49. *A HORS is simultaneous if and only if it is pairwise simultaneous.*

PROOF The 'only if' part is trivial, so we show the 'if' part. The proof is by induction on the size of a set $\mathcal{U}$ of redexes in a term s. First remark that one cannot have two distinct redexes $u =^{\mathrm{def}} (\phi,\aleph)$ and $v =^{\mathrm{def}} (\phi,\beth)$ in $\mathcal{U}$. By assumption these would be simultaneous, which would require ϕ to have two distinct origins in some binary context $D\boxed{\,,\,}$ into which $\{u,v\}$ is extracted, which is not possible by linearity. Pick any redex $u =^{\mathrm{def}} (\phi,\aleph)$ from $\mathcal{U}$. By definition there exists an extraction $e\colon s \twoheadleftarrow_{\mathcal{SC}} C\boxed{l}$ of u into some context $C\Box$. By assumption we have that each redex $v =^{\mathrm{def}} (\psi,\beth)$ in $\mathcal{V} =^{\mathrm{def}} \mathcal{U} - \{u\}$ has an origin in $C\Box$. By linearity and the above remark these are all distinct and redex positions in $C\Box$. Now we can apply the induction hypothesis and obtain an extraction $h\Box$ of $\{\mathcal{V} \lfloor e \rfloor\}$ from $C\Box$ into some context $D\Box$. By natural closure of reduction under substitutions and by preservation of linearity under substitution and concatenation, we obtain that $e\,;h\boxed{l}$ is an extraction of $\mathcal{U}$ from s into $D\Box$. In a diagram (using some obvious notation):

$$\begin{tikzcd}
D\boxed{L,l} \arrow[d, dashed, "h\boxed{l}"] & \supseteq & D\boxed{L,} \arrow[d, "h\Box"] \\
C\boxed{l} \arrow[d, "e"] & \supseteq & C\Box \\
s & &
\end{tikzcd}$$

$\odot$

This proposition is the first step toward localising simultaneity of a HORS to absence of so-called *critical pairs* between its rewrite rules. In the next section in which we consider $\lambda_{\overline{\eta}}^{\rightarrow}$-calculus as substitution calculus, this property will be localised further.

3.1.3. Weakly Orthogonal HORSs

In this subsection we consider HORS which satisfy A1–7 and try to find conditions ensuring their weak orthogonality (see Definition 2.4.11). We search for conditions guaranteeing that the property P, defined by $P((a,\Phi)) =^{\text{def}} \Phi$ *is a set of simultaneous redexes in* a, satisfies the conditions in the definition of weak orthogonality. First remark that, with this definition $P \implies \mathit{COMP}$ by Theorem 3.1.45. Moreover, P is closed under reduction (in the LRS), because by the (Claim in the proof of the) Develop Lemma 3.1.43, the set of residuals of a set of simultaneous redexes is simultaneous again. By these two observations the following condition on P suffices for weak orthogonality:

if $P((s,\mathcal{U}))$ and u is a rewrite step from s to t, then either

1. there exists a $\mathcal{U}$-development from s to t, or
2. $\mathcal{U} \cup \{u\}$ is a set of simultaneous redexes.

The following proposition reduces this condition to a condition on triples of redexes similar to the condition on pairs of redexes in Proposition 3.1.49.

DEFINITION 3.1.50. A HORS is *cubic*, if every triple of pairwise simultaneous redexes is simultaneous.

Note that by Proposition 3.1.49 we have that every simultaneous HORS is cubic, but in general not every triple of pairwise simultaneous redexes is simultaneous as witnessed by the following example.

EXAMPLE 3.1.51. Consider the HORS with operator symbol $\mathtt{A}$, set of substitution operators $\{G, F, F_m, F_{m,n}.1 \leqslant m, n \leqslant 3\}$, and rewrite rule $\aleph =^{\text{def}} G(\mathtt{A}) \to \ldots$ with head symbol $\mathtt{A}$ at head position 1 and substitution rules:

$$
\begin{array}{rcl}
F_1(G(x))(y)(z) & \to & F(x)(y)(z) \\
F_2(x)(G(y))(z) & \to & F(x)(y)(z) \\
F_3(x)(y)(G(z)) & \to & F(x)(y)(z) \\
F_{1,2}(x)(G(y))(z) & \to & F_1(x)(y)(z) \\
F_{1,3}(x)(y)(G(z)) & \to & F_1(x)(y)(z) \\
F_{1,2}(G(x))(y)(z) & \to & F_2(x)(y)(z) \\
F_{2,3}(x)(y)(G(z)) & \to & F_2(x)(y)(z) \\
F_{1,3}(G(x))(y)(z) & \to & F_3(x)(y)(z) \\
F_{2,3}(x)(G(y))(z) & \to & F_3(x)(y)(z)
\end{array}
$$

with the ordinary descendant relation for term rewriting systems. This HORS satisfies A1–7, but the triple $\{(001,\aleph),(01,\aleph),(1,\aleph)\}$ of pairwise simultaneous redexes in $F(\mathtt{A})(\mathtt{A})(\mathtt{A})$ is not simultaneous. One can think of the symbol F as a resource needed for all three redexes, but being only available twice. Each extraction consumes one F, so at most two redexes can be extracted simultaneously.

PROPOSITION 3.1.52. *Let $\mathcal{H}$ be a HORS, such that*

> *if u and v are not simultaneous, then they are steps between the same two terms.*

If $\mathcal{H}$ is cubic, then it is weakly orthogonal.

PROOF We check the two conditions above which sufficed for proving weak orthogonality. Either $\{u,v\}$ is simultaneous for each $v \in \mathcal{U}$, or not. In the former case, one proves, by induction on the size of the set $\mathcal{V} =^{\text{def}} \mathcal{U} \cup \{u\}$ of simultaneous redexes that there exists an extraction of $\mathcal{V}$ from s, nach wievor (Proposition 3.1.49) but now using cubicity to ensure that origins of simultaneous redexes are simultaneous again. That is, one proves that every set of pairwise simultaneous redexes is simultaneous. In the latter case there is some $v \in \mathcal{U}$ doing exactly the same as u, hence there exists a $\mathcal{U}$-development from s to t. $\odot$

Intuitively, if for a weakly orthogonal system, occurrences of left-hand sides of two rules share some common resource, then replacing them by their respective right-hand sides produces exactly the same result.

3.1.4. Combinations of HORSs

In studying large rewriting systems, it is of obvious interest to partition such systems into smaller, more manageable subsystems. These subsystems should be more manageable in the sense that the properties one likes to prove for them are easier to show. Of course, this is only a sensible approach if the properties of the subsystems can be combined to imply the desired property of the original large system. A property which is preserved when combining systems is called *modular*. In recent years, the study of modular properties of rewriting systems has attracted much attention.

In this subsection we show that confluence is preserved when taking the union of two HORSs satisfying A1–7 which are simultaneous to each other, i.e. every pair of steps consisting of one step from each of the HORSs is simultaneous. The result can be strengthened a bit, by allowing that if the steps are not simultaneous, then they are steps between the same two terms. In the sections on TRSs, CRSs and HRSs we show that this result generalises many of the results in the literature.

Definition 3.1.53. Two HORSs $\mathcal{H} =^{\mathrm{def}} \langle \mathcal{A}, \mathcal{SC}, \mathcal{R}\rangle$ and $\mathcal{I} =^{\mathrm{def}} \langle \mathcal{A}, \mathcal{SC}, \mathcal{S}\rangle$, satisfying A1–7, are *simultaneous* to each other, if both are cubic and if each pair $\{u,v\}$ consisting of an $\mathcal{R}$-redex u, and an $\mathcal{S}$-redex v is simultaneous. They are *weakly simultaneous* to each other, if every such pair is either simultaneous or consists of steps between the same two terms.

This definition is the 'commutativity' version of Definition 3.1.48, i.e. a HORS is simultaneous if it is simultaneous to itself.

Theorem 3.1.54. *If $\mathcal{H} =^{\mathrm{def}} \langle \mathcal{A}, \mathcal{SC}, \mathcal{R}\rangle$ and $\mathcal{I} =^{\mathrm{def}} \langle \mathcal{A}, \mathcal{SC}, \mathcal{S}\rangle$ are confluent and weakly simultaneous to each other, then $\mathcal{H} \cup \mathcal{I} =^{\mathrm{def}} \langle \mathcal{A}, \mathcal{SC}, \mathcal{R} \cup \mathcal{S}\rangle$ is confluent.*

Proof Because $\mathcal{H}$ and $\mathcal{I}$ are confluent by assumption, by the Lemma of Hindley-Rosen it suffices to show that $\mathcal{H}$ and $\mathcal{I}$ commute, that is:

$$\begin{tikzcd}
\cdot \arrow[r, twoheadrightarrow, "\mathcal{I}"] \arrow[d, twoheadrightarrow, "\mathcal{H}"'] & \cdot \arrow[d, twoheadrightarrow, dashed, "\mathcal{H}"] \\
\cdot \arrow[r, twoheadrightarrow, dashed, "\mathcal{I}"'] & \cdot
\end{tikzcd}$$

Using the 'commutativity' variant of the Strip Lemma 2.2.5(2), it suffices to show that for any set $\mathcal{U}$ of simultaneous $\mathcal{R}$-redexes we can construct the following diagram.

$$\begin{tikzcd}
s \arrow[r, "v"] \arrow[d, "\mathcal{U}"'] & r \arrow[d, dashed, "\mathcal{U}'"] \\
t \arrow[r, twoheadrightarrow, dashed, "\mathcal{V}"'] & p
\end{tikzcd}$$

where v is an $\mathcal{I}$-step, $\mathcal{U}'$ is a set of simultaneous $\mathcal{R}$-redexes, $s \to_{\mathcal{U}} t$ is a complete development of $\mathcal{U}$ and $r \to_{\mathcal{U}'} p$ is a complete development of $\mathcal{U}'$. There are two cases to consider:

1. If v is simultaneous with each step in $\mathcal{U}$, one shows by reasoning analogous to Proposition 3.1.52, that first extracting v from s by some extraction e gives a set of pairwise simultaneous $\mathcal{R}$-redexes $\{\mathcal{U}\ \lfloor e \rfloor\}$, which is simultaneous by cubicity of $\mathcal{H}$, and hence the set $\mathcal{U} \cup \{v\}$ is simultaneous. By (the proof of) the Develop Lemma 3.1.43 we know that performing the step v preserves simultaneity, so we can take $\mathcal{U}' =^{\mathrm{def}} \{\mathcal{U}\ \lfloor v \rfloor\}$.

2. If v is not simultaneous with some step $u \in \mathcal{U}$, then we can take $\mathcal{U}' =^{\mathrm{def}} \{\mathcal{U}\ \lfloor u \rfloor\}$.

Finally, the diagram can be completed by an application of the Finite Developments theorem 3.1.45. To start the induction in the Strip Lemma, we observe that if $\mathcal{U}$ consists of just one step, it is simultaneous. $\odot$

3.2. Simply Typed Lambda Calculus

In this section we will show that simply typed λ-calculus satisfies all the criteria required for the substitution calculus of a Higher-Order Rewriting System (see Theorem 3.1.46). We present hardly any original material, but just fix the notation and cite the results we need from the literature. The literature on λ-calculus is extensive, so we refer the reader to the standard references for (untyped) λ-calculus: [Bar84] and [HS86]. A more recent survey can be found in [Bar92].

We first note that there is not just one λ-calculus. The only thing all λ-calculi have in common are the rules for term formation, the 'renaming' rule $\boldsymbol{\alpha}$ and the 'evaluation' rule $\boldsymbol{\beta}$. One then varies on this theme, by e.g. restricting term formation or adding rules. The convention we use to denote a restriction on term formation is by superscripting λ. The convention we use to denote the extra rules is by subscripting λ. So, $\lambda^{\rightarrow}_{\eta}$ denotes λ-calculus restricted to terms which have a so-called simple type (built from $\rightarrow$) and having $\boldsymbol{\beta}$ and $\boldsymbol{\eta}$ as rewrite rules.

We will be interested mainly in $\lambda^{\rightarrow}_{\overline{\eta}}$-calculus, i.e. simply typed λ-calculus with restricted η-expansion, because in the next sections we will 'encode' TRSs, CRSs and HRSs, respectively, as HORSs having $\lambda^{\rightarrow}_{\overline{\eta}}$ as substitution calculus.

Let us first present the definition of the set of simple types, used to restrict term formation.

DEFINITION 3.2.1. *Simple types* are inductively defined as follows:

1. the *base* type o is a simple type,
2. if σ and τ are simple types, then the *function* type $\sigma \rightarrow \tau$ is a simple type.

The variables σ, τ, υ range over the set $\mathcal{T}$ of simple types.

As usual we use association to the right for the function type constructor, e.g. $\sigma \rightarrow \tau \rightarrow \upsilon$ denotes the type $\sigma \rightarrow (\tau \rightarrow \upsilon)$. To measure the complexity of a type two functions are useful. The arity of a type expresses how many arguments a term of such a type needs. The order of a type indicates how 'active' the arguments of a term of that type are.

DEFINITION 3.2.2. On the set of simple types the *arity* and *order* functions, $\mathsf{arity}, \mathsf{order} : \mathcal{T} \rightarrow \mathbb{N}$ are inductively defined by:

1. $\mathsf{arity}(o) =^{\mathrm{def}} \mathsf{order}(o) =^{\mathrm{def}} 0$,
2. $\mathsf{arity}(\sigma \rightarrow \tau) =^{\mathrm{def}} \mathsf{arity}(\tau) + 1$, $\mathsf{order}(\sigma \rightarrow \tau) =^{\mathrm{def}} (\mathsf{order}(\sigma) + 1) \vee (\mathsf{order}(\tau))$.

The formation of raw preterms on an alphabet as per Definition 3.1.1 is now restricted by requiring that a simple type can be assigned to every raw preterm, the idea being that only to ra preterms which represent the mathematician's usual set-theoretic notion of function a type can be assigned.

DEFINITION 3.2.3. Let $\mathcal{A}$ be an alphabet and let a unique simple type be assigned to every nullary symbol from $\mathcal{A}$, such that we have for each type σ, infinitely many free and bound variables of type σ. The set of raw *simply typed preterms* is the union over all simple types σ, of sets of raw preterms *of type* σ, inductively defined by:

1. a nullary symbol of type σ is a preterm of type σ,
2. if s is a raw preterm of type $\sigma \to \tau$ and t is a raw preterm of type σ, then $s(t)$ is a raw preterm of type τ,
3. if s is a raw preterm of type τ, and ξ is a bound variable of type σ, then $\xi.s$ is a raw preterm of type $\sigma \to \tau$

Note that every raw simply typed preterm has a unique simple type, so it is sound to define the arity, $\mathsf{arity}(s)$, of a raw simply typed preterm s as the arity of the type it has. Because the raw simply typed preterms form a subset of the set of all raw preterms, the operations on and notions defined for these pertain. We denote by $\Lambda^{\to}$ the set of all simply typed preterms.

In the all important rule $\boldsymbol{\beta}$ of λ-calculus, one needs to switch from a preterm $\xi.s$ to its raw subpreterm s. If s contains the bound variable ξ, then it is not a preterm, but it can be easily made so, by replacing all occurrences of the bound variable ξ by a free variable x. This process can also be applied the other way around.

DEFINITION 3.2.4. Let s be a simply typed preterm and x some free variable. Then $\xi.s'$ is called an *x-closure of* s if ξ does not occur in s and s' is obtained from s by replacing every occurrence of x by ξ. The set of all x-closures of s is denoted by $\mathsf{clos}_x(s)$. For $\sigma =^{\mathrm{def}} x_1,\ldots,x_m$ a sequence of free variables, the *σ-closure of* s is the x_1-closure of $\ldots$ of the x_m-closure of s.

NOTATION 3.2.5. In the rest of this subsection the variables s, t, r range over $\Lambda^{\to}$.

For example, $\xi.\xi$ and $\zeta.\zeta$ are both x-closures of x, the term $\xi.x$ is a y-closure of x, but the term $\xi.\xi.\xi$ is not an x-closure of $\xi.x$. In the next definition, the usual rules $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\eta}$ are defined on simply typed λ-terms.

DEFINITION 3.2.6. The following reduction rules are defined on $\Lambda^{\to}$:

1. $\boldsymbol{\alpha} =^{\mathrm{def}} \{(s_0,s_1).s_0, s_1 \in \mathsf{clos}_x(s)\}$. This rule allows to rename bound variables preserving their distinctness.
2. $\boldsymbol{\beta} =^{\mathrm{def}} \{(s'(t),[x \mapsto t]s).s' \in \mathsf{clos}_x(s)\}$. This rule evaluates the *function* represented by s' on its *argument* t, by first 'x-opening' it to its *body* s and then *replacing* every occurrence of x in s by t.
3. $\boldsymbol{\eta} =^{\mathrm{def}} \{(\xi.s(\xi),s)\}$. This rule replaces the representation of a function by a smaller one.

To any rule $\aleph$ from the above, we associate the $\aleph$-rewrite relation $\rightarrow_{\aleph} =^{\mathrm{def}} \bigcup \phi \in \mathsf{Pos}.\rightarrow_{(\phi,\aleph)}$ on $\Lambda^{\rightarrow}$ called the *compatible closure* of $\aleph$, which is generated by the following inference system:

$$\frac{s \rightarrow_{(\phi,\aleph)} t}{s(r) \rightarrow_{(0\phi,\aleph)} t(r)}\,(\mathsf{appl}) \qquad \frac{(s,t) \in \aleph}{s \rightarrow_{(\varepsilon,\aleph)} t}\,(\aleph)$$

$$\frac{s \rightarrow_{(\phi,\aleph)} t}{r(s) \rightarrow_{(1\phi,\aleph)} r(t)}\,(\mathsf{appr}) \qquad \frac{s \rightarrow_{(\phi,\aleph)} t}{s' \rightarrow_{(0\phi,\aleph)} t'}\,(\mathsf{abs})$$

where $s' \in \mathsf{clos}_x(s)$ and $t' \in \mathsf{clos}_x(t)$ for some free variable x. We define $\rightarrow_{(\phi,\bar{\eta})} =^{\mathrm{def}} \leftarrow_{(\phi,\eta)} - \leftarrow_{\beta}$ and $\rightarrow_{\bar{\eta}}$ is the union over all positions ϕ of $\rightarrow_{(\phi,\bar{\eta})}$. This rule is called *restricted* η-expansion, because it allows η-expansion steps only in case it does not create a β-redex which can 'undo' the step. (see [Aka93, Cor. 16] for a justification of this definition).

In the definition, the first component of a pair $(\phi,\aleph)$ indicates at which position in the preterm the rewrite step 'takes place'. A step $s \rightarrow_{(\phi,\aleph)} t$ is said to *contract* the $\aleph$-*redex* at *depth* ϕ. It is *below* ψ (thinking of preterms as upside down trees), if $\phi \succeq \psi$. To indicate that d from s to t is a sequence of $\aleph$-steps below ψ, we write $d\colon s \twoheadrightarrow_{(\succeq\psi,\aleph)} t$.

We now show that $\lambda_{\bar{\eta}}^{\rightarrow}$-calculus has the properties needed for being a substitution calculus of an orthogonal HORS. The first condition on HORSs requires $\lambda_{\bar{\eta}}^{\rightarrow}$-calculus to be complete.

PROPOSITION 3.2.7. *(A1) COMP*$(\lambda_{\bar{\eta}}^{\rightarrow})$.

PROOF This is the basic fact about typed λ-calculi, hence can be found at several places, see e.g. [Wol93, Thm. 2.35 and 2.38]. $\odot$

REMARK 3.2.8. In fact, $\lambda_{\bar{\eta}}^{\rightarrow}$-calculus is only confluent up to α-conversion, i.e. modulo $\leftrightarrow_{\alpha}^{*}$. To remedy this, a confluent and terminating α-rewrite relation is required. This can be done, for example, by defining a reduction relation which chooses a unique representative of a preterm via De Bruijn's *nameless terms* (see [Bar84, App. C]).

The second condition on HORSs requires that the substitution calculus is only needed for gluing contexts and left-hand sides together. By Proposition 3.1.17 it suffices to prove that $\lambda_{\bar{\eta}}^{\rightarrow}$-calculus is context free.

PROPOSITION 3.2.9. *(A2)* $\lambda_{\bar{\eta}}^{\rightarrow}$*-calculus is context free.*

PROOF The reduction relation $\rightarrow_{\beta\bar{\eta}}$ preserves closedness of preterms by definition. The reduction relation $\rightarrow_{\bar{\eta}}$ is context sensitive ([Aka93, Exa. 1]), but one easily shows $\leftrightarrow_{\beta\bar{\eta}}^{*} = \leftrightarrow_{\beta\eta}^{*}$, and both $\leftrightarrow_{\beta}^{*}$ and $\leftrightarrow_{\eta}^{*}$ are easily seen to be context free (cf. [Bar84, pp. 54,55], but note that the 'variable-naming' problems there, do not appear here). $\odot$

The third condition on HORSs requires that rewrite steps can be serialised. By Proposition 3.1.22 it is sufficient to prove that the HORS is serial. In the proof, the following technical proposition is useful.

Proposition 3.2.10. *$\bar{\eta}$-normal forms are closed under β-reduction and under substitution of $\bar{\eta}$-normal forms.*

Proof A preterm s in $\bar{\eta}$-normal form reflects the structure of its type in the following sense. Each occurrence in s of any symbol a (be it operator or variable) of type σ, has precisely $\mathsf{arity}(\sigma)$ arguments, i.e. it occurs as $a(s_1)\ldots(s_{\mathsf{arity}(\sigma)})$. Then consider how an $\bar{\eta}$-step could arise in either case. Alternatively, see [Aka93, p. 12] and [Nip93, p. 309]. $\odot$

From this we immediately deduce that if $C\boxed{s} \twoheadrightarrow_{\beta\bar{\eta}} t$, where $C\Box$ is a context and s is a term, then the reduction is in fact a β-reduction $C\boxed{s} \twoheadrightarrow_{\beta} t$.

Proposition 3.2.11. *(A3) $\lambda^{\rightarrow}_{\bar{\eta}}$-calculus is serial.*

Proof

1. Let $C\Box$ be a context having at least one hole. We show that the leftmost occurrence of $\Box$ is persistent. By the preceding proposition we only need to show that the leftmost occurrence of the hole is persistent with respect to β-reduction after substitution of terms for the other occurrences of the hole. Remark that every redex created by the substitution is to the right of the persistent hole. The only way in which the persistent hole could be erased or duplicated is when a β-reduction above it takes place. Initially there are no such reductions possible and one easily shows that it is in fact not possible to create such a β-reduction step. Alternatively, one can apply the standardisation theorem for λ-calculus (see e.g. [Hin78b, Thm. 1], [Klo80, Thm. I.9.7], [GLM92]).

2. Suppose $d\Box\colon C\boxed{l,\ } \twoheadrightarrow_{\beta\bar{\eta}} D\Box$, where $C\Box$ is a context and l is a term. By the preceding proposition, we know that this is in fact a β-reduction. Since β-reduction is closed under substitution, we obtain that $d\boxed{g}\colon C\boxed{l,g} \twoheadrightarrow_{\beta\bar{\eta}} D\boxed{g}$ is a β-reduction again, as required.

$\odot$

The fourth condition on HORSs requires the substitution calculus to be a DRS. We can define the DRS associated to $\lambda^{\rightarrow}_{\bar{\eta}}$ any way we like, but we should keep in mind the conditions, such as natural closure under substitutions, which will be required later on.

Definition 3.2.12. The DRS associated to $\lambda^{\rightarrow}_{\bar{\eta}}$, which is also denoted by $\lambda^{\rightarrow}_{\bar{\eta}}$, is $\langle \Lambda^{\rightarrow}, \mathsf{Pos}\times\{\beta,\bar{\eta}\}, \rightarrow, \mathcal{R}\mathsf{Pos}, \lfloor\cdot\rfloor\rangle$, where the descendant relation $\lfloor\cdot\rfloor$ for each of the rules α, β, η and $\bar{\eta}$ is defined as follows: Let $\phi \in \mathcal{R}\mathsf{Pos}(s)$, and $\psi \in \mathcal{R}\mathsf{Pos}(t)$.

1. if $u =^{\mathrm{def}} s \rightarrow_{(\varepsilon,\alpha)} t$, then $\phi\lfloor u\rfloor\phi$,

2. if $u =^{\text{def}} s'(t) \to_{(\varepsilon,\beta)} [x \mapsto t]s$, then $00\phi\lfloor u\rfloor\phi$, if $s(\phi) \not\equiv x$ and $1\psi\lfloor u\rfloor\phi\,;\psi$, if $s(\phi) \equiv x$,

3. if $u =^{\text{def}} \xi.s(\xi) \to_{(\varepsilon,\eta)} s$, then $00\phi\lfloor u\rfloor\phi$,

4. if $u =^{\text{def}} s \to_{(\varepsilon,\bar{\eta})} t$, then $\lfloor u\rfloor =^{\text{def}} \lfloor t \to_{(\varepsilon,\eta)} s\rfloor^{-1}$.

This is extended to rewrite steps which take place at non-root positions via the above inference rules. Let in the following ϕ, ψ and χ be arbitrary $\mathcal{R}$-positions in s, t and r respectively. Consider a rewrite step v which is inferred from the hypothesis u of one of the inference rules.

1. if the inference rule is appl, then $v =^{\text{def}} s(r) \to_{(0\phi',\aleph)} t(r)$. We define $0\phi\lfloor v\rfloor 0\psi =^{\text{def}} \phi\lfloor u\rfloor\psi$, and $1\chi\lfloor v\rfloor 1\chi$.

2. if the inference rule is appr, then $v =^{\text{def}} r(s) \to_{(1\phi',\aleph)} r(t)$. We define $1\phi\lfloor v\rfloor 1\psi =^{\text{def}} \phi\lfloor u\rfloor\psi$, and $0\chi\lfloor v\rfloor 0\chi$.

3. if the inference rule is abs, then $s \to_{(0\phi',\aleph)} t$. We define $\phi\lfloor v\rfloor\psi =^{\text{def}} 0\backslash\phi\lfloor u\rfloor 0\backslash\psi$.

Note that if $s' \to_{(\phi',\aleph)} t'$ is the hypothesis in the last item, $s'(\chi) \equiv x$ and $\chi\lfloor u\rfloor\chi'$, then we do not have $0\chi\lfloor v\rfloor 0\chi'$, because the free variable x is turned into a bound variable and positions of these are not traced.

REMARK 3.2.13. The descendant relation is different from the origins function defined in [DD93]. Basically, we are only interested in tracing what happens to head-positions of redexes.

The fifth condition on HORSs consists of two parts. First, the DRS associated to $\lambda_{\bar{\eta}}^{\to}$ should be parametric. Second, the rewrite rules of a HORS should all be head-defined. We first check the former condition and then provide some sufficient conditions on terms to be head-defined. Terms satisfying these are called *patterns* using the terminology introduced by Nipkow [Nip91, Def. 3.1].

PROPOSITION 3.2.14. *(A5) $\lambda_{\bar{\eta}}^{\to}$ is parametric.*

PROOF To prove: if $d, e\colon s \twoheadrightarrow^{!}_{\beta\bar{\eta}} t$, then $\lfloor d\rfloor = \lfloor e\rfloor$. To see this, first check that if $\Phi =^{\text{def}} a\mathsf{Pos}(s)$ for some symbol $a \in \mathcal{A}_{\mathcal{R}}$ and $u =^{\text{def}} s \to_{\beta\bar{\eta}} s'$, then $a\mathsf{Pos}(s') = \{\Phi\ \lfloor u\rfloor\}$, because neither β- nor $\bar{\eta}$-steps create $\mathcal{A}_{\mathcal{R}}$-symbols. Next, observe that the 'name' of a symbol a in $\mathcal{A}_{\mathcal{R}}$ does not matter for rewrites. If in s some a-occurrence at ϕ is replaced by a fresh symbol b obtaining s', then the rewrites d and e are transformed into rewrites d' to t_1 and e' to t_2. Both t_1 and t_2 are normal forms so, by confluence of $\lambda_{\bar{\eta}}^{\to}$, $t_1 \equiv t_2$. Combining these two observations, we have $\phi\lfloor d\rfloor\psi \iff \phi\lfloor d'\rfloor\psi \iff t_1(\psi) \equiv b \iff t_2(\psi) \equiv b \iff \phi\lfloor e'\rfloor\psi \iff \phi\lfloor e\rfloor\psi$. $\odot$

REMARK 3.2.15. Alternatively, one can invoke the theory of permutation equivalence of reductions in λ-calculus (orthogonal expression reduction systems), see e.g. [Lév80], [Klo80, Rem. I.10.1.3] and [Kha92]. However, these papers only deal with orthogonal systems, so do not apply to $\lambda^{\rightarrow}_{\bar{\eta}}$-calculus. There are two solutions to this problem. The first solution is a kind of work-around and based on the observation that at all the places where parametricity was used (in the lemmata needed for confluence by orthogonality) the initial preterm is in fact in $\bar{\eta}$-normal form, so we can use the above proposition and Proposition 3.2.10 to get the desired result. The other solution is to prove the result directly. We sketch the proof method, which is analogous to the proof of Proposition 2.4.16. First show that for every local divergence (u,v), there exists a convergence (e',d'), such that $\lfloor \underrightarrow{u ; e'} \rfloor = \lfloor \underrightarrow{v ; d'} \rfloor$. Then show by noetherian induction on the initial preterm of a divergence (d,e) ordered by $\rightarrow^{+}_{\beta\bar{\eta}}$, that there exists a convergence (e',d') such that $\lfloor \underrightarrow{d ; e'} \rfloor = \lfloor \underrightarrow{e ; d'} \rfloor$. The first part is shown by case analysis, the only interesting part being the interaction between β- and $\bar{\eta}$-steps. Here it is essential that only $\mathcal{A}_{\mathcal{R}}$-positions are traced. The second part is shown as in Newman's Lemma, using that $\lfloor \underrightarrow{d ; e} \rfloor = \lfloor \underrightarrow{d} \rfloor ; \lfloor \underrightarrow{e} \rfloor$.

To present the set of patterns it is handy to introduce some terminology (see [Wol93]). A term (i.e. a $\beta\bar{\eta}$-normal form) s' of arity m can always be written as $\xi_1.\ldots\xi_m.a(s'_1)\ldots(s'_k)$, where $s'_1,\ldots,s'_k$ are in this form again. That is, s' is the $x_1,\ldots,x_m$-closure of some term $s =^{\mathrm{def}} a(s_1)\ldots(s_k)$. The parts of the expression to the left and to the right of the dot, are called the *binder* and *matrix* of s'. The symbol a is the *head* and $s'_1,\ldots,s'_k$ are the *arguments* of s (in general arguments are raw terms). Note that any position of a symbol of arity k in s', always has the form $\phi ; 0^k$, where 0^k denotes a string of k zeroes as usual.

DEFINITION 3.2.16. Let l' of arity m be the $x_1,\ldots,x_m$-closure of l. Then l' is called a *pattern*, if it satisfies the following conditions:

1. The head of l is an operator symbol.
2. Let x of arity k be among $x_1,\ldots,x_m$.
   (a) There is at least one occurrence of x in l,
   (b) For every occurrence $\phi ; 0^n$ of x in l, we have that if $l \twoheadrightarrow^{!}_{(\succeq\phi,\eta)} g$, then $\phi\backslash g \equiv x(\xi_1)\ldots(\xi_k)$, where $\xi_1,\ldots,\xi_k$ is a list of pairwise distinct bound variables.

The set of all patterns is denoted by $\mathcal{P}\mathsf{at}$.

The last condition on patterns expresses that for each occurrence of a variable which is in the binder of l' has as arguments only ($\bar{\eta}$-normal forms of) pairwise distinct bound variables not in the binder (cf. [Nip91, Nip93]).

Next we will show that for a HORS having $\lambda^{\rightarrow}_{\bar{\eta}}$ as substitution calculus to be head-defined, it suffices that the left-hand sides are patterns.

Proposition 3.2.17. *If all left-hand sides of the rules of a HORS are patterns, then it is head-defined.*

Proof We have to check the conditions stated in Definition 3.1.28. Let l' be as in Definition 3.2.16 and let its head $\mathtt{F}$ have arity k. Define $\mathsf{head}(l') =^{\mathrm{def}} 0^{m+n}$, that is, $\mathsf{head}(l')$ is the position of the head $\mathtt{F}$ in l'. According to Definition 3.1.28, we must verify two conditions for head-positions.

1. Let $\phi =^{\mathrm{def}} 0^{m+k}$. Then the position $\psi\,;0^m\,;\phi$ should have a unique descendant χ along any reduction from $C\boxed{l'}$ to its normal form $s =^{\mathrm{def}} C\boxed{l'}{\downarrow_{\beta\bar\eta}}$, where $C\Box$ is a linear context with hole-position $\psi\,;0^m$. First, note that the position of the hole must indeed be of the form $\psi\,;0^m$ because $C\Box$ is in $\bar\eta$-normal form, that is, $\psi\backslash C\Box \equiv \Box(t_1)\dots(t_m)$. Next, note that the reduction to normal form must in fact be a β-reduction as remarked above. Now, $\psi\backslash C\boxed{l'} \equiv \boxed{\xi_1.\dots\xi_m.\mathtt{F}(l'_1)\dots(l'_k)}(t_1)\dots(t_m)$. It is easy to check that the reduction of $C\boxed{l'}$ must be below ψ and that the symbol $\mathtt{F}$ always must have exactly one descendant using the standardisation theorem for λ-calculus. Doing a leftmost-outermost reduction gives that $\psi\backslash t \equiv \mathtt{F}(s'_1)\dots(s'_k)$, so the descendant of $\psi\,;0^k\,;\phi$ is $\psi\,;0^n$, obtained by 'erasing the $2\times m$ zeroes in the middle'. By parametricity of $\lambda_{\bar\eta}^{\rightarrow}$, this does not depend on the chosen reduction.

2. Let $C_1\Box$ with hole position $\psi_1\,;0^m$ and $C_2\Box$ with hole position $\psi_2\,;0^m$, be linear contexts. Let $e_1\colon s \twoheadleftarrow_{\beta\bar\eta} C_1\boxed{l'}$ and $e_2\colon s \twoheadleftarrow_{\beta\bar\eta} C_2\boxed{l'}$ be expansions from a term s. Let χ be a position in s such that $\chi\lfloor\underrightarrow{e_1}\rfloor\psi_1\,;0^m\,;\phi$ and $\chi\lfloor\underrightarrow{e_2}\rfloor\psi_2\,;0^m\,;\phi$. To show: $C_1\Box \equiv C_2\Box$.

   (a) First remark that, by the first item, we must have that $\psi_1\,;0^n = \chi = \psi_2\,;0^n$, hence $\psi_1 = \psi_2$. Call this position ψ. Again according to the first item, both expansions must be below ψ, so the contexts $C_1\Box$ and $C_2\Box$ are the same for positions not below ψ.

   (b) It remains to show that $C_1\Box$ and $C_2\Box$ are also the same for positions below ψ, i.e. each pair t_k, t'_k of corresponding arguments of $\Box$ in both of them must be equal. Let l' be the $x_1,\dots,x_m$-closure of l. By the definition of pattern, we can η-normalise l to g such that each x occurs as $x(\xi_1)\dots(\xi_m)$ in g, where $\xi_1,\dots,\xi_m$ is a list of distinct bound variables. Let g' be the $x_1,\dots,x_m$-closure of g. Consider the preterms $C_1\boxed{g'}$ and $C_2\boxed{g'}$. Their β-reduction to normal form ends in an $\bar\eta$-normal form. By confluence of $\lambda_{\beta\bar\eta}^{\rightarrow}$ and because $\leftrightarrow_{\beta\bar\eta}^{*} = \leftrightarrow_{\beta\eta}^{*}$ these normal forms must be equal to s. So we have shown: $C_1\boxed{l'} \twoheadrightarrow_\eta C_1\boxed{g'} \twoheadrightarrow_\beta s$ and $C_2\boxed{l'} \twoheadrightarrow_\eta C_2\boxed{g'} \twoheadrightarrow_\beta s$. Suppose we take innermost reductions for β in both of these reductions. Consider the first step in the first reduction. It reduces the redex $\boxed{g'}(t_1)\dots(t_m)$. Because x_1, say of arity i, is required to occur in l by the second condition on patterns, the next i steps 'plug the bound variables into t_1', starting from $t_1(\xi_1)\dots(\xi_i)$. But since all the variables $\xi_1,\dots,\xi_i$

were required to be distinct, this is just a renaming of t_1. The same happens in the second reduction, resulting in the same renaming of t'_1. These renamings must be the same term in s, but then t_1 and t'_1 must have been equal in the first place. Similar reasoning can be used to obtain that $t_k \equiv t'_k$ for each k.

$\odot$

The next example shows that the conditions in the definition of patterns cannot be relaxed easily, without losing the above uniqueness property.

EXAMPLE 3.2.18. 1. The term $\xi.\xi$ has no operator symbol, so certainly no head symbol. Consequently, it has no descendants in the term $\mathtt{A} =^{\mathrm{def}} \boxed{\xi.\xi}(\mathtt{A})\downarrow_{\beta\bar{\eta}}$. So, it would be senseless to speak of a rewrite step by a rule $\xi.\xi \to \ldots$ being represented by some symbol in $\mathtt{A}$.

2. We have $\boxed{\xi.\mathtt{A}}(s) \twoheadrightarrow_{\beta\bar{\eta}} \mathtt{A} \twoheadleftarrow_{\beta\bar{\eta}} \boxed{\xi.\mathtt{A}}(t)$, but $\Box(s) \not\equiv \Box(t)$. The term $\xi.\mathtt{A}$ violates the second condition on patterns because the variable ξ does not occur in $\mathtt{A}$. The problem is, that no restriction is put on variables that do not occur in the matrix.

3. Let $l =^{\mathrm{def}} \xi.\mathtt{G}(\zeta.\xi(\zeta)(\zeta))$, $C\Box =^{\mathrm{def}} \Box(\varsigma.\gamma.\varsigma)$, $D\Box =^{\mathrm{def}} \Box(\varsigma.\gamma.\gamma)$, and $s =^{\mathrm{def}} \mathtt{G}(\xi.\xi)$. We have $C\boxed{l} \twoheadrightarrow_{\beta\bar{\eta}} s \twoheadleftarrow_{\beta\bar{\eta}} D\boxed{l}$, but $C\Box \not\equiv D\Box$. The term l violates the third condition on patterns because the bound variable ζ occurs twice as argument. The problem caused by this is that the context can 'choose' which of these occurrences to use to establish the binding in the result. In the example there is only one binding in s, $C\Box$ chooses the first occurrence of ζ to establish it and $D\Box$ the second one.

4. Let $l =^{\mathrm{def}} \xi.\zeta.\mathtt{G}(\xi(\zeta))$, $C\Box =^{\mathrm{def}} \Box(\xi.\xi(\xi.\xi)(\mathtt{A}))(\bar{m})$, where $\bar{m}$ is any Church-numeral. We have $C\boxed{l} \twoheadrightarrow_{\beta\bar{\eta}} \mathtt{G}(\mathtt{A})$, for every such context $C\Box$ and all these contexts are pairwise different. The term l violates the third condition, because the bound variable ζ occurs both as argument of a (bound) variable in the binder and in the binder itself.

5. Let $l =^{\mathrm{def}} \xi.\mathtt{G}(\xi(\mathtt{A}))$, $C\Box =^{\mathrm{def}} \Box(\zeta.\zeta)$, $D\Box =^{\mathrm{def}} \Box(\zeta.\mathtt{A})$, and $s =^{\mathrm{def}} \mathtt{G}(\mathtt{A})$. We have $C\boxed{l} \twoheadrightarrow_{\beta\bar{\eta}} s \twoheadleftarrow_{SC} D\boxed{l}$, but $C\Box \not\equiv D\Box$. The term l violates the third condition because ξ has the operator $\mathtt{A}$ as argument. As a consequence both the projection context C and the imitation context D can be used to obtain the term s. The third condition ensures that only projections are possible. Imitation would require variables which are bound inside l to be imitated, which is not possible. For example, let $g =^{\mathrm{def}} \xi.\mathtt{G}(\zeta.\xi(\zeta))$ and $t =^{\mathrm{def}} \mathtt{G}(\zeta.\zeta)$. We have $C\boxed{g} \leftrightarrow^*_{\beta\bar{\eta}} t$ by the projection context, but an imitation context similar to D cannot be found.

In each case in the example, the problem is that different 'substitutions' for a term s which is not a pattern, might result in the same term. If s is the left-hand side of a rule, then this rule can be non-deterministic in the sense that applying it at some position might lead to different results. Consider the rule ([Nip91, p. 345]):

$$\xi.\mathtt{G}(\zeta.\xi(\zeta)(\zeta)) \to \xi.\mathtt{G}(\zeta.\varsigma.\xi(\zeta)(\varsigma))$$

There are four ways of applying this rule to the term $\mathtt{G}(\zeta.\mathtt{H}(\zeta)(\zeta))$, depending on whether $\Box(\zeta.\varsigma.\mathtt{H}(\zeta)(\zeta))$, $\Box(\zeta.\varsigma.\mathtt{H}(\zeta)(\varsigma))$, $\Box(\zeta.\varsigma.\mathtt{H}(\varsigma)(\zeta))$, or the context $\Box(\zeta.\varsigma.\mathtt{H}(\varsigma)(\varsigma))$ is chosen, each application resulting in a different term. Although this system has no critical pairs, it is certainly not confluent. A possible solution would be to require that the 'substitution' is used 'in the same way' in the right-hand side as in the left-hand side of a rule. For example, if ξ would occur always as a term of the form $\xi(s)(s)$, then the rule is deterministic again.

REMARK 3.2.19. Actually, the restriction in the third condition that the arguments should be $\bar{\eta}$-forms of variables, could be relaxed somewhat, without jeopardising uniqueness of extractions. For example, for the term $l =^{\mathrm{def}} \xi.\mathtt{G}(\zeta.\xi(\mathtt{H}(\zeta)))$ Proposition 3.2.17 still holds. The property needed is that the argument $\mathtt{H}(\zeta)$ of ξ is 'rigid' and contains a variable. In general, the arguments of an occurrence of a variable from the binder, should be distinct 'patterns' again.

The sixth condition on HORS requires its left-hand sides to be linear, that is, contained in some set of terms which is linear according to Definition 3.1.34.

DEFINITION 3.2.20. The set $\mathcal{LP}\mathsf{at}$ is the subset of the set $\mathcal{P}\mathsf{at}$ of patterns obtained by restricting the second condition in Definition 3.2.16 to:

There is *precisely* one occurrence of x in l.

PROPOSITION 3.2.21. *(A6) The set $\mathcal{LP}\mathsf{at}$ is linear.*

PROOF Easy, noting that the only redexes which can be created in a reduction starting from $C\boxed{l_1,\ldots,l_m}$ are of the following two forms:

1. $s(\xi_1{\downarrow_{\bar{\eta}}})\ldots(\xi_n{\downarrow_{\bar{\eta}}})$, reducing this redex might duplicate the η-expanded bound variable $\xi_1{\downarrow_{\bar{\eta}}}$, but this term does not contain any symbol from $\mathcal{A}_{\mathcal{R}}$ so can be duplicated without violating linearity,

2. $s'(s_1)\ldots(s_n)$, where s' is the x-closure of s in which x occurs precisely once (this is the case if s is of the form $\xi{\downarrow_{\bar{\eta}}}$). Reducing this redex does not duplicate s_1.

$\odot$

REMARK 3.2.22. The term l of Remark 3.2.19 is not linear, because the operator $\mathtt{H}$ can be duplicated as witnessed by $\boxed{l}(\varsigma.\mathtt{F}(\varsigma)(\varsigma)) \to_{\beta\bar{\eta}} \mathtt{G}(\zeta.\mathtt{F}(\mathtt{H}(\zeta))(\mathtt{H}(\zeta)))$. Nevertheless, a HORS having such a term as left-hand side is orthogonal because the symbol $\mathtt{H}$ does not occur as head symbol. For orthogonality one only needs to trace symbols in $\mathcal{A}_{\mathcal{R}}$ which represent redexes, i.e. which do occur as head symbols of rules. In the example, only $\mathtt{G}$, not $\mathtt{H}$ or $\mathtt{F}$, needs to be traced. This amounts to the usual partitioning of the set of operator symbols into a set of *defined* or *destructor* symbols ($\mathtt{G}$) and a set of *confined* or *constructor* symbols ($\mathtt{H}, \mathtt{F}$).

The seventh condition on HORS requires the substitution calculus to be naturally closed under substitution. This proposition is the reason why we have defined the descendant rewriting system associated to $\lambda^{\rightarrow}_{\overline{\eta}}$ in the way we have done. The proofs of the following propositions are sketchy, because we have refrained from introducing a formal definition of replacement (cf. [Hue80, p. 807]) which would be needed for a precise treatment.

PROPOSITION 3.2.23. *(A7)* $\lambda^{\rightarrow}_{\overline{\eta}}$ *is naturally closed under substitution.*

PROOF We must check the two conditions in Definition 3.1.37 for $\lambda^{\rightarrow}_{\overline{\eta}}$.

1. A rewrite starting from $C\boxed{l,}$, for $C\boxed{\;,\;}$ a context and l a left-hand side, should be closed under substitution of another left-hand side for the hole. This is direct from Proposition 3.2.10.

2. By a case analysis for a step $u\Box =^{\text{def}} C\Box \rightarrow_{(\varepsilon,\aleph)} D\Box$ at root position. For steps at non-root positions we use induction on the inference of the step.

$\odot$

We call a HORS having $\lambda^{\rightarrow}_{\overline{\eta}}$ as substitution calculus such that each left-hand side is a pattern, a *pattern* HORS.

The last condition on HORS requires that every set of redexes is simultaneous, or equivalently (see Proposition 3.1.49) that every pair of distinct redexes is simultaneous. This is, of course, not enforced by left-hand sides being patterns, so it is useful to have some characterisation of simultaneity in terms of the rules.

DEFINITION 3.2.24. Let $u =^{\text{def}} (\phi\,;0^m,\aleph)$, $v =^{\text{def}} (\psi\,;0^n,\beth)$ be distinct redexes in a term s, where m (n) is the arity of the head symbol of $\aleph$ ($\beth$).

1. The redexes are said to be *disjoint* if ϕ and ψ are disjoint (see Definition 1.3.2).

2. Otherwise we may assume without loss of generality that $\psi \succeq \phi$. We then have so called *prefix*-redexes. Let the left-hand side l' of $\aleph$ be the $x_1,\ldots,x_k$-closure of l. Two cases are distinguished: (cf. [Hue80, Lem. 3.1])

   (a) the redex u is said to *nest* v, if $\phi\,;\chi\,;\omega = \psi$, where $\chi\,;0^i$ is the position of some variable x of arity i in l, and x is among $x_1,\ldots,x_k$.

   (b) otherwise u is said to *overlap* with v.

A rule $\aleph$ *overlaps* with another rule $\beth$, if some $\aleph$-redex overlaps with some $\beth$-redex. A HORS *has overlap* if some rule overlaps with another one, it is *non-overlapping* otherwise. If in each case that a redex overlaps with another one, the rewrite steps end in the same term, then the HORS is said to have *trivial* overlaps.

The following proposition corresponds to [Nip91, Lem. 4.2] and [Nip93, Lem. 4.7] for the higher-order case and [Hue80, Lem. 3.1] for the first-order case.

PROPOSITION 3.2.25. *A left-linear pattern HORS is simultaneous if and only if it is non-overlapping.*

PROOF Let u, v be a pair of distinct redexes as above.

> Suppose $\mathcal{H}$ is simultaneous. Let ψ' be the origin of $\psi\,;0^n$ along the extraction $s \twoheadleftarrow_\beta C\boxed{l'}$ of u from s. By simultaneity of u and v and natural closure of reductions under substitutions, this position ψ' can only be a position in the context part $C\Box$ and one can show by induction on the length of the reduction that either u must be disjoint from v, or one of the redexes nests the other.
>
> Suppose $\mathcal{H}$ has no overlapping redexes. Let $e\colon s \twoheadleftarrow_{(\succeq\phi,\beta)} C\boxed{l'}$ be an extraction of u from s and $g\colon s \twoheadleftarrow_{(\succeq\psi,\beta)} D\boxed{g'}$ be an extraction of v from s. There are two cases to consider:
>
> 1. If u is disjoint from v, then ϕ is disjoint from ψ and we can construct the extraction $D\Box \twoheadleftarrow_{(\succeq\phi,\beta)} E\boxed{,l'}$, because the expansions take place below the disjoint positions ϕ and ψ, so do not influence each other.
> 2. If u nests v, then we can construct an extraction as in the first item, but for a different reason. One can show that the expansion e never expands a β-redex below a (descendant of) $\phi\,;\chi$. Because $\psi \succeq \phi\,;\chi$, it is safe to do the extraction g (which is below ψ) first.

$\odot$

THEOREM 3.2.26. *Every non-overlapping left-linear pattern HORS is orthogonal, hence confluent.*

PROOF By Theorem 3.1.46, using Proposition 3.1.49 and Proposition 3.2.25. $\odot$

In the preceding proposition simultaneity of a left-linear pattern HORS is reduced to the absence of overlaps between rules of the HORS. This can be further specialised to the absence of *critical pairs* between rules, where a critical pair is a 'most general instance' of overlap of one rule with another. In general, whether such a most general instance exists depends on the substitution calculus involved and on the restrictions put on the left-hand sides, but in the case of pattern HORSs there exist most general instances and they are computable as well. For the moment we are satisfied with the characterisation of simultaneity via non-overlapping rules because it allows for a simple proof that each left-linear pattern HORS as above is cubic.

PROPOSITION 3.2.27. *Every left-linear pattern HORS is cubic.*

PROOF Consider a triple $\mathcal{U} =^{\text{def}} \{u_1, u_2, u_3\}$ of pairwise simultaneous redexes, which are extracted by e_1, e_2 and e_3, respectively. If two redexes are simultaneous and not disjoint, then by Proposition 3.2.25 one redex must nest the other, so without loss of generality, there are four cases to consider:

1. All three redexes are disjoint,
2. u_1 nests u_2, which in turn nests u_3,
3. u_1 nests both u_2 and u_3 which are disjoint,
4. u_1 nests u_2 and both are disjoint from u_3.

In each of these cases, first performing the β-steps in e_3, then the ones in e_2 and finally the steps in e_3 gives and extraction of $\mathcal{U}$, by reasoning analogous to Proposition 3.2.25(2). $\odot$

From this we immediately (or may be better: finally) obtain the following two corollaries.

COROLLARY 3.2.28. *A left-linear pattern HORS having trivial overlaps is weakly orthogonal (cf. Definition 2.4.11).*

PROOF By Proposition 3.2.27, Proposition 3.1.52 and Proposition 3.2.25. $\odot$

COROLLARY 3.2.29. *If $\mathcal{H} =^{\text{def}} \langle \mathcal{A}, \mathcal{SC}, \mathcal{R} \rangle$ and $\mathcal{I} =^{\text{def}} \langle \mathcal{A}, \mathcal{SC}, \mathcal{S} \rangle$ are confluent left-linear pattern HORS and all the overlaps between $\mathcal{H}$-rules and $\mathcal{I}$-rules are trivial, then $\mathcal{H} \cup \mathcal{I}$ is confluent.*

PROOF We can apply Theorem 3.1.54, because checking that $\mathcal{H}$ and $\mathcal{I}$ are weakly simultaneous to each other boils down to checking that all overlaps are trivial by Proposition 3.2.27 and Proposition 3.2.25. $\odot$

Up till now we have been after a negative result, that is, to have some decision procedure for guaranteeing the absence of overlap. But to show that for each overlap of a step u with another step v, the steps are between the same terms, we need to localise the property of overlap to so-called critical pairs. For the first-order case critical pairs were introduced by Huet ([Hue80]) and for the higher-order case by Nipkow ([Nip91, Nip93]). We just present Nipkow's result in our terminology.

As a first step to localising overlap, it is useful to observe that attention can be restricted to overlaps as above such that $\phi = \varepsilon$. If two overlapping steps are performed inside a common context, then they can be performed without it and vice versa.

PROPOSITION 3.2.30. *The relation* $\rightarrow_\beta$ *is* naturally closed under contexts*:*

1. *Let* $\psi \succeq \phi$ *be some positions in* s*, then* $u{:}\, s \rightarrow_{(\psi,\beta)} t$ *if and only if* $\phi\backslash u{:}\, \phi\backslash s \rightarrow_{(\phi\backslash\psi,\beta)} \phi\backslash t$*, and*
2. $\phi\backslash\omega\lfloor\phi\backslash u\rfloor\phi\backslash\omega' \iff \omega\lfloor u\rfloor\omega'$.

PROOF By induction on the inference of the rewrite step. $\odot$

REMARK 3.2.31. The statement is not entirely correct, because $\phi\backslash s$ usually is a raw preterm and β-reduction is not defined for these. This can easily overcome by introducing a new operator '$\backslash\backslash$' which behaves like '$\backslash$', but replaces bound variables by (fresh) free variables when taking the subpreterm of an abstraction preterm.

By the proposition we have for $\psi \succeq \phi$, that $u =^{\mathrm{def}} s \rightarrow_{(\phi;0^m,\aleph)} t$ overlaps with $v =^{\mathrm{def}} s \rightarrow_{(\psi;0^n,\beth)} t'$ if and only if $\phi\backslash u =^{\mathrm{def}} \phi\backslash s \rightarrow_{(0^m,\aleph)} \phi\backslash t$ overlaps with $\phi\backslash v =^{\mathrm{def}} \phi\backslash s \rightarrow_{(\psi\backslash\phi;0^n,\aleph)} \phi\backslash t'$. Intuitively, one can 'forget' about the common context (the part not below ϕ) of both steps.

A redex u in s of the form $(0^k,\aleph)$ where k is the arity of the head symbol of $\aleph$, is called a *root* redex. Obviously, this root redex must extract into a context of the form $\Box(s_1)\ldots(s_m)$, where m is the arity of the left-hand side of $\aleph$. Contexts of this form will play the rôle of meta level substitutions. The next observation one can make is that if such a root redex overlaps with a redex $v =^{\mathrm{def}} (\psi,\beth)$ in s, then the position ψ must be a descendant in s of the (unique) position $0^{2\times m};\psi$. If it would descend from a position in some s_i, then u would nest v. In this case we say that u overlaps with v *at position* ψ in s.

DEFINITION 3.2.32. By a *closed* m-ary context, a context which contains no free variables other than $\Box_1, \ldots, \Box_m$ is meant. A *substitution context* is a unary linear closed context of the form $\xi_1.\ldots\xi_m.\Box(s_1)\ldots(s_n)$. A *unification pair for* $l \rightarrow r$*,* $g \rightarrow d$ *at* ψ is a pair of closed unary linear contexts $(C\Box,D\Box)$, say both of arity m, such that

1. $C\boxed{l}\downarrow_{\beta\bar\eta} \equiv D\boxed{g}\downarrow_{\beta\bar\eta}$ (the components unify l and g to form a *critical peak*),
2. a root redex u in s extracts into $C'\Box$ and overlaps with the redex v in s at position ψ, which extracts into $D'\Box$, if and only if $C\Box(s_1)\ldots(s_m) \twoheadrightarrow^!_{\beta\bar\eta} C'\Box$ and $D\Box(s_1)\ldots(s_m) \twoheadrightarrow^!_{\beta\bar\eta} D'\Box$, for some $s_1,\ldots,s_m$ (every overlap at ψ is an instance and vice versa), and
3. for every other pair $(C'\Box,D'\Box)$ satisfying the first two items, there is a substitution context $E\Box$ such that $E\boxed{C\Box}\downarrow_{\beta\bar\eta} \equiv C'\Box$ and $E\boxed{D\Box}\downarrow_{\beta\bar\eta} \equiv D'\Box$ (the pair is most general).

If v is a root redex, then we speak of a *root* unification pair. The pair $(C\boxed{r}\downarrow_{\beta\bar\eta},D\boxed{d}\downarrow_{\beta\bar\eta})$ is called a *critical pair*, it is an *overlay* if it stems from a root unification pair. It is called *trivial* if both components are equal.

REMARK 3.2.33. Note that the order of the components of a unification pair and hence the order of a critical pair is important, for example, (□,F(□)) is a unification pair for $\mathtt{F(A)} \to \ldots$, $\mathtt{A} \to \ldots$ at position 1, but (F(□),□) cannot be a unification pair since F(□) is not a substitution context, which is required for the first component of a unification pair. In the literature the order of critical pairs seems not to be fixed. For example, Dershowitz and Jouannaud ([DJ90, p. 286]) and Nipkow ([Nip93, Def. 3.7]) employ the order as above, while Huet ([Hue80, p. 809]) and Klop ([Klo92, Def. 2.4.9]) employ the reverse order.

The first condition on unification pairs expresses that when both left-hand sides are put into the holes of the unification pair, we obtain a 'unifier' for them. The second condition states that every overlap is an instance of the critical peak and also the other way around: every instance of the critical peak gives rise to an overlap. The third condition states that this unifier is a most general one: every overlap is an instance of it. By the second condition, one can take fresh free variables $x_1, \ldots, x_m$ for $s_1, \ldots, s_m$ in order to show that a unification pair is unique up to some renaming.

THEOREM 3.2.34. *A left-linear pattern HORS having only trivial critical pairs is confluent.*

PROOF By Proposition 3.2.30, it suffices to check that for each pair of steps u, v as in Definition 3.2.32, we have that they end in the same term. From the definition of critical pair it follows that u ends in $C\boxed{r}(s_1)\ldots(s_m)\downarrow_{\beta\bar{\eta}}$, and v ends in $D\boxed{d}(s_1)\ldots(s_m)\downarrow_{\beta\bar{\eta}}$, for some $s_1, \ldots, s_m$. By assumption every critical pair is trivial, so $C\boxed{r}\downarrow_{\beta\bar{\eta}} \equiv C\boxed{d}\downarrow_{\beta\bar{\eta}}$ and we are done. ⊙

Expansion and Substitution In the above we have made use of $\lambda^{\to}_{\bar{\eta}}$ as substitution calculus. For the matching phase (see page 67) β-expansion is employed and for the substitution phase β-reduction is used. Let us comment briefly on this.

First the substitution process. Since β-reduction is complete, any strategy will reach the normal form, i.e. will execute the substitution correctly. This does not mean that all these strategies are *optimal* in the sense of number of β-steps performed. Furthermore, from a computation point of view a β-reduction step is not a single step; the argument of a redex is copied to each of the occurrences of the bound variable in the body. To remedy this, β-steps can be decomposed into more elementary steps in several ways. We name some of the literature dealing with these two questions.

1. Copying is inefficient if the argument is large. It is better to *share* the argument between all these occurrences in order to avoid doing double work. The optimality problem then turns into: how to achieve optimal sharing in a reduction. This question was posed by Lévy (see [Lév80]) and solved by Lamping (see the bibliography of [Lan93] for papers on the subject), roughly speaking, by introducing extra operators (so-called

fans) allowing to share contexts and subterms. (In fact this does not provide a completely satisfactory answer. It shifts the problem to the optimal management of the extra operators.)

2. In a β-step the argument of a redex is substituted for all occurrences of the bound variable in the body. It is more realistic to allow substitution for each of these occurrences in its turn. This leads to so-called *local substitution* (see [Ned92]).

3. Apart from substituting only for one bound variable at the time, the β-reduction process can also be split into smaller steps gradually moving the argument toward the place where the bound variables appears in the body. This is realised by calculi for *explicit* substitution (see [Har93, Ned92]). It would be interesting to see what happens if such a fine-grained calculus is taken as substitution calculus of a HORS.

Next, we consider the matching process. The β-expansion needed to perform a HORS-step, can be viewed as a search process for redexes. To make this process efficient, one needs to restrict the enormous amount of possible β-expansions to those expansions which might result in finding a redex. Unfortunately, β-expansion is not as well-studied as β-reduction. For one, it is not listed among the main subjects within pure lambda calculus in [Bar84, p. xiv]. We do have the following important result of Miller [Mil91] (see also [Nip91, App. A]):

THEOREM 3.2.35. *It is decidable whether two patterns are unifiable; if they are unifiable, a most general unifier can be computed.*

This result entails that matching is decidable for patterns. Moreover, it has linear space and time complexity as was shown by Qian [Qia92]. However, since matching forms a bottle-neck in most implementations of rewrite systems, it is useful to have really efficient matching processes (see e.g. [Lip93] for the first-order case). We do not know of much literature on this subject.

REMARK 3.2.36. It requires some careful checking that our definition of critical pair for pattern HORSs coincides with the one of Nipkow ([Nip93, Def. 3.7]) for HRSs, the only difference being that we work with closed terms instead of open ones.

As explained above, the calculi for explicit substitution can be viewed as implementations of β-reduction. It would be interesting to see how the various matching processes in the literature can be related to β-expansion, and 'calculi for explicit matching'.

As noted above, β-expansion has not been the subject of intensive study, so there are a lot of properties to be investigated. For example, is β-expansion confluent?

EXAMPLE 3.2.37. ([Bar84, Exe. 3.5.11.(vii)]). The following two preterms, due to Plotkin: $(\xi.\xi(\xi))b(c)$ and $(\zeta.b(\zeta)(b(c)))c$ show that β-expansion is not locally confluent in the untyped λ-calculus.

This example does not work directly in various subcalculi such as simply typed and linear λ-calculus. We conjecture however that these do not enjoy (local) confluence either.

CONJECTURE 3.2.38. *Consider a term $F(G(H(A)))$ and two β-expansions of this term: $(\xi.F(G(\xi)))H(A)$ and $F((\zeta.G(H(\zeta)))A)$, obtained by 'splitting' the term in two different ways, do not have a common β-expansion. This example is directly translatable in typed and linear λ-calculus.*

In the next sections we show how ever more general classes of rewriting systems can be viewed as pattern HORSs and how the above confluence by orthogonality result specialises to these systems. We show for these systems how the usual notion of orthogonality for them implies their orthogonality in our sense in a natural way. The known 'confluence by orthogonality' proofs for these systems can then be reduced to one general confluence proof for pattern HORSs. Each time, we first present the system in the way it is usually presented and then show how it can be rendered as a HORS. The systems are presented roughly in increasing order of expressivity.

3.3. Term Rewriting Systems

In this section we present Term Rewriting Systems (TRSs) as higher-order rewriting systems. For the presentation of TRSs we closely follow the presentation of Klop in [Klo92].

DEFINITION 3.3.1. A *Term Rewriting System (TRS)* is a tuple $\langle \Sigma, \mathcal{R} \rangle$ of a *signature* Σ and a set of *rewrite rules* $\mathcal{R}$. The signature Σ consists of:

1. A countably infinite set of *variables* x_1, x_2, x_3, ... also denoted as x, y, z, x', y', ...,
2. a non-empty set of *operator* or *function symbols* each equipped with an *arity* (a natural number), i.e. the number of 'arguments' it is supposed to have.

The set of *terms over* Σ is $\mathsf{Ter}(\Sigma)$ and is defined inductively:

1. each variable in Σ is a term over Σ,
2. if $\mathtt{F}$ is an m-ary operator symbol and $s_1, \ldots, s_m \in \mathsf{Ter}(\Sigma)$ $(m \geqslant 0)$, then $\mathtt{F}(s_1, \ldots, s_m) \in \mathsf{Ter}(\Sigma)$. The s_n $(n = 1, \ldots, m)$ are the *arguments* of the last term.

Contexts are 'terms' containing one occurrence of a special symbol [ ], denoting an empty place. A context is generally denoted by $C[\,]$. If $s \in \mathsf{Ter}(\Sigma)$ and s is substituted for [ ], the result is $C[s] \in \mathsf{Ter}(\Sigma)$. A *rewrite rule* is a pair (l,r) of terms in $\mathsf{Ter}(\Sigma)$. It will be written as $l \to r$. Two conditions are imposed:

1. The *left-hand side* l is not a variable,
2. the variables in the *right-hand side* r already occur in l.

Let $\mathcal{H} =^{\mathrm{def}} \langle \Sigma, \mathcal{R} \rangle$ be a TRS. It determines a set $\longrightarrow_{\mathcal{H}}$ of rewrite steps

$$\{(C[l^\theta],C[r^\theta]).l \to r \in \mathcal{R}\}$$

where θ is a *substitution*, i.e. a map from $\mathsf{Ter}(\Sigma)$ to $\mathsf{Ter}(\Sigma)$ which satisfies $\mathtt{F}(s_1,\ldots,s_m)^\theta \equiv \mathtt{F}({s_1}^\theta,\ldots,{s_m}^\theta)$ for every m-ary operator symbol $\mathtt{F}$ (here $m \geqslant 0$). For a TRS $\mathcal{H} =^{\mathrm{def}} \langle \Sigma, \mathcal{R} \rangle$ there is a corresponding ARS, namely $\to_{\mathcal{H}} =^{\mathrm{def}} \langle \mathsf{Ter}(\Sigma), \longrightarrow_{\mathcal{H}} \rangle$. The relation $\to_{\mathcal{H}}$ can equivalently be presented by the following inference rules.

$$\frac{l \to r \in \mathcal{R}}{l^\theta \to_{\mathcal{H}} r^\theta}(\mathsf{sub}) \qquad \frac{s_n \to_{\mathcal{H}} t_n}{F(s_1,\ldots,s_n,\ldots,s_m) \to_{\mathcal{H}} F(s_1,\ldots,t_n,\ldots,s_m)}(\mathsf{ctx})$$

Example 3.3.2. The prime example of a TRS is Curry's Combinatory Logic ([Cur30a, p. 513]). Its signature consists of one binary operator symbol app and three nullary operator symbols S, K and I. The rules of Combinatory Logic are:

$$\begin{array}{rcl} \mathtt{app}(\mathtt{I})(x) & \to & x \\ \mathtt{app}(\mathtt{app}(\mathtt{K})(x))(y) & \to & x \\ \mathtt{app}(\mathtt{app}(\mathtt{app}(\mathtt{S})(x))(y))(z) & \to & \mathtt{app}(\mathtt{app}(x)(z))(\mathtt{app}(y)(z)) \end{array}$$

By first using infix notation for the operator symbol app, then using association to the left to remove as many parentheses as possible and finally omitting the operator symbol app altogether, these rules are also written as

$$\begin{array}{rcl} \mathtt{I}x & \to & x \\ \mathtt{K}xy & \to & x \\ \mathtt{S}xyz & \to & xz(yz) \end{array}$$

3.3.1. From TRS to HORS

We now show that every TRS $\mathcal{H}$ can be mapped onto a pattern HORS $\overline{\mathcal{H}}$, called the *closure* of $\mathcal{H}$, such that $\to_{\mathcal{H}}$ is isomorphic to $\to_{\overline{\mathcal{H}}}$. In fact, the transformation preserves much more structure, but for (finitary) rewriting purposes this correspondence suffices. The main (only) point of the transformation is taking the closure of the rules, changing free variables into bound variables:

$$\frac{l \to r}{\xi_1.\ldots\xi_m.l' \to \xi_1.\ldots\xi_m.r'}$$

where $\xi_1 \ldots \xi_m.l'$ and $\xi_1 \ldots \xi_m.r'$ are closed terms denoting the closures of l and r with respect to the free variables in l. Instead of dealing with substitution for the free variables at a meta level, they are changed into object variables, and substitution for them is now dealt with by the substitution calculus.

DEFINITION 3.3.3. Let Σ be a signature. The alphabet $\mathcal{A} =^{\text{def}} \mathcal{I}(\Sigma)$ associated to Σ is defined by:

1. if x is a variable in Σ, then $\mathcal{I}(x)$ is a free variable in $\mathcal{A}$ of type o,
2. if $\mathtt{F}$ is an m-ary operator symbol in Σ, then $\mathcal{I}(\mathtt{F})$ is an operator symbol in $\mathcal{A}$ of type $o \to \ldots \to o \to o$ of arity m.

If s is a term in $\mathsf{Ter}(\Sigma)$, then the term $\mathcal{I}(s)$ in $\mathbb{T}(\mathcal{A})$, associated to it is defined by

1. if s is a variable x, then $\mathcal{I}(s) =^{\text{def}} \mathcal{I}(x)$,
2. if s is $\mathtt{F}(s_1, \ldots, s_m)$, then $\mathcal{I}(s) =^{\text{def}} \mathcal{I}(\mathtt{F})(\mathcal{I}(s_1)) \ldots (\mathcal{I}(s_m))$.

The translation $\mathcal{I}$ is required to be injective on symbols and one easily checks that the translation of terms is then well-defined, injective and yields terms, not just preterms. The translation of terms is naturally extended to rules and sets of rules. The *closure* of a TRS $\mathcal{H} =^{\text{def}} \langle \Sigma, \mathcal{R} \rangle$, is the full sub-HORS $\mathcal{I}(\mathcal{H})$ of $\langle \mathcal{A}, \lambda_{\overline{\eta}}^{\rightarrow}, \overline{\mathcal{I}(\mathcal{R})} \rangle$, obtained by restricting $\mathbb{T}(\mathcal{A})$ to the set $\mathcal{I}(\mathsf{Ter}(\Sigma))$ of translated terms. Here $\overline{\mathcal{I}(\mathcal{R})}$ is the closure of $\mathcal{I}(\mathcal{R})$ obtained by closing each rule with respect to the sequence of free variables in its left-hand side (see Definition 3.2.4) from left to right.

NOTATION 3.3.4. In the sequel we confuse a symbol with its translation, e.g. a with $\mathcal{I}(a)$.

There are three things going on in this definition. First, functional terms are translated into applicative ones. The set of translated terms $S =^{\text{def}} \mathcal{I}(\mathsf{Ter}(\Sigma))$ is the set of *first-order* terms over $\mathcal{A}$, and can also be described by the inference rules

$$\frac{s_1 \in S \quad \ldots \quad s_m \in S}{a(s_1) \ldots (s_m) \in S}$$

for every m-ary symbol a in Σ (note that variables have arity zero). In the following we will not bother too much about this change in format.

Second, each rule is closed with respect to the variables in its left-hand side. By the condition on TRSs that the variables in the right-hand side of a rule are contained in the ones of the left-hand side, this procedure produces closed HORS rules. It is immediate from the form of first-order terms that the closure of a non-variable terms gives a pattern in the sense of Definition 3.2.16. Hence the translation of a term rewriting system is a pattern HORS (see page 99), because a left-hand sides of a TRS rule is required to be non-variable. Furthermore, this change from free into bound variables amounts to transforming the meta level substitutions in the TRS $\mathcal{H}$ into object-level substitutions ($\beta\eta$-expansions) in the pattern HORS $\mathcal{I}(\mathcal{H})$, as shown in the following proposition.

PROPOSITION 3.3.5. *Let $\mathcal{H}$ be a TRS, let s be a term in $\mathcal{H}$ and let θ be a substitution. Then $\mathcal{I}(s^\theta) \equiv \overline{\mathcal{I}(s)}(\mathcal{I}(x_1{}^\theta))\dots(\mathcal{I}(x_m{}^\theta))\downarrow_{\beta\bar{\eta}}$, where $\overline{\mathcal{I}(s)}$ is the x_1, $\dots$, x_m-closure of $\mathcal{I}(s)$, and $\mathsf{Fvar}(s) \subseteq \{x_1,\dots,x_m\}$.*

PROOF One shows $\mathcal{I}(s^\theta) \leftrightarrow^*_{\beta\bar{\eta}} \overline{\mathcal{I}(s)}(\mathcal{I}(x_1{}^\theta))\dots(\mathcal{I}(x_m{}^\theta))$ by induction on the structure of the term s. The result then follows by confluence of $\beta\bar{\eta}$, because $\mathcal{I}(s^\theta)$ is in $\beta\bar{\eta}$-normal form. $\odot$

Last, but not least, rewriting is restricted to first-order terms. If one wants to deduce confluence of a sub-HORS $\mathcal{I}$ of some HORS $\mathcal{H}$, from the confluence of $\mathcal{H}$, then the so-called *subject reduction* property has to be proven for $\mathcal{I}$, that is, one must show that the restricted set of terms is closed under rewriting (cf. [Bar92, Prop. 3.1.11]). Here, we must check that first-order terms are closed under rewriting.

PROPOSITION 3.3.6. *Let s be a first-order term in $\mathcal{I}(\mathcal{H})$. If u is an m-ary $\aleph$-redex in s, then the context $C\Box$ into which u is extracted does not contain λ-abstractions and the hole must occur as $\Box(\mathcal{I}(s_1))\dots(\mathcal{I}(s_m))$.*

PROOF First, we show that $C\Box$ cannot contain λ-abstractions. We give two proofs, the first one from an expansion point of view, the second one from a reduction point of view. First note that β-expansion never erases a subterm and that the only way in which it can erase a β-redex is by abstraction over a variable which has a higher type than the bound variable of the β-redex. Now, if $C\Box$ would contain a λ-abstraction, which is not part of a β-redex (hence it stems from an erased β-redex) then the preterm $C\boxed{l}$, where l is the left-hand side of $\aleph$ must contain a λ-abstraction of higher type. The term l cannot be part of this redex, because all the abstractions in it are of base type. So, for each λ-abstraction in $C\Box$, it would have to contain a λ-abstraction of higher type, which is obviously not possible.

From a reduction point of view, we know that $C\boxed{l}$ contains at most one β-redex, say $(\xi.s)t$. The variable ξ is of base type, so a β-redex can only be created if s is of the form $\xi'.s'$, where ξ' is again of base type. Such a β-reduction terminates and leaves all the λ-abstractions in $C\Box$ unaffected. Because s does not contain λ-abstractions, we conclude that $C\Box$ does not contain them either.

Because of this we know that the only λ-abstractions in $C\boxed{l'}$, where l' is a left-hand side of a $\mathcal{I}(\mathcal{H})$-rule, must be the ones in the binder of l'. One easily shows that the β-reduction is deterministic and 'plugs' the arguments in the body of the left-hand side. By the construction of $\mathcal{I}(\mathcal{H})$ each argument occurs at least once as subterm of s, so must be of the specified form. $\odot$

From this proposition it easily follows that if $s \twoheadleftarrow_{\beta\bar{\eta}} C\boxed{\overline{\mathcal{I}(l)}} \to C\boxed{\overline{\mathcal{I}(r)}} \twoheadrightarrow_{\beta\bar{\eta}} t$ is a rewrite step in $\mathcal{I}(\mathcal{H})$ starting from the first-order term s, then t is a first-order term again.

After these observations, the following lemma, which expresses that a TRS is isomorphic to its closure should not come as a surprise.

LEMMA 3.3.7. *Let $\mathcal{H}$ be a TRS, then $s \to_{\mathcal{H}} t \iff \mathcal{I}(s) \to_{\mathcal{I}(\mathcal{H})} \mathcal{I}(t)$.*

PROOF For the only if direction, one shows, for every context $C[\ ]$, term s and substitution θ, $\mathcal{I}(C[s^\theta]) \leftrightarrow^*_{\beta\bar{\eta}} \mathcal{I}(C)[\overline{\mathcal{I}(s)}(\mathcal{I}(x_1{}^\theta))\dots(\mathcal{I}(x_m{}^\theta))]$, where $\overline{\mathcal{I}(s)}$ is the $x_1, \dots, x_m$-closure of $\mathcal{I}(s)$, and $\mathsf{Fvar}(s) \subseteq \{x_1, \dots, x_m\}$. This follows from Proposition 3.3.5 and an easy induction proof of $\mathcal{I}(C[s]) \equiv \mathcal{I}(C)[\mathcal{I}(s)]$.

For the if direction, one shows using Proposition 3.3.6 that if $\mathcal{I}(s) \twoheadleftarrow_{\beta\bar{\eta}} C'\boxed{l'}$ is an extraction of a rule $\aleph'$ in $\overline{\mathcal{I}(\mathcal{R})}$, from $\mathcal{I}(s)$ into $C'\Box$, then $C'\boxed{l'} \equiv \mathcal{I}(C)[\boxed{l'}(\mathcal{I}(s_1))\dots(\mathcal{I}(s_m))]$. Define $x_n{}^\theta =^{\text{def}} s_n$ for each n. Using the first item and injectivity of $\mathcal{I}$, we conclude that $s \equiv C[l^\theta]$, from which the result follows easily. $\odot$

There is a lot of freedom in associating a pattern HORS to a TRS. We have chosen for the one above because it is the most natural one and seems to require the least effort. We sketch some other possibilities to render a TRS as a HORS.

A disadvantage of the procedure above is that it translates each 'functional' term $\mathtt{F}(s_1, \dots, s_m)$ into an 'applicative' one $\mathtt{F}(s_1')\dots(s_m')$. To circumvent this a term product operator ',' of type $o \to o \to o$ could be added to the alphabet. (In order to remain really functional, one would have to consider also adding a type product constructor '$\times$' to $\lambda_{\bar{\eta}}^{\to}$.) Then, e.g. $\mathtt{F}(s, t, r)$ can be translated to $\mathtt{F}(,(s')(,(t')(r')))$, instead of to $\mathtt{F}(s')(t')(r')$. Using infix notation and association to the right for ',' the former can we written as $\mathtt{F}(s', t', r')$. The advantage of such a translation would be mainly from an implementation point of view. Products have more structure than functions, so can be implemented in a faster way. More operationally, the head symbol $\mathtt{F}$ is easier to access, because only one application node has to be traversed from the root to access it.

In the translation above, rules are closed with respect to the sequence of free variables in its left-hand side, from left to right. In fact, closing under any sequence of variables containing the free variables in the rule, would do, i.e. would generate the same rewrite relation.

CLAIM Let $\mathcal{H}$ be a pattern HORS. Let $C\Box$ be a context such that $\aleph =^{\text{def}} l \to r$, where $l =^{\text{def}} \xi_1.\dots\xi_m.s$ and $r =^{\text{def}} \xi_1.\dots\xi_m.t$. If we have one of the situations:

1. $\aleph' =^{\text{def}} \xi_{\pi(1)}.\dots\xi_{\pi(m)}.s \to \xi_{\pi(1)}.\dots\xi_{\pi(m)}.t$ for some permutation π on m, or

2. $\aleph' =^{\text{def}} \xi_2.\dots\xi_m.s \to \xi_2.\dots\xi_m.t$ is a rule again, i.e. ξ_1 does occur in neither s nor t,

then there exists a context $D\Box$, such that $C\boxed{l} \leftrightarrow^*_{\beta\bar{\eta}} D\boxed{l'}$ and $C\boxed{r} \leftrightarrow^*_{\beta\bar{\eta}} D\boxed{r'}$.

PROOF OF CLAIM It is easy to check that we can take for $D\Box$ the normal forms of

1. $C\boxed{(\xi_1.\ldots\xi_m.\Box)\xi_{\pi(1)}\ldots\xi_{\pi(m)}}$, and
2. $C\boxed{\xi_1.\Box}$,

respectively, where the holes must be chosen of the appropriate type. Both normal forms indeed have one hole, that is, are linear contexts. $\odot$

By the first case, we can permute the variables in the binder of a rule in such a way that the variables which do not occur in either the left- or right-hand side come first, followed by the rest in order of appearance in the left- and right-hand side. By the second case, we can remove all the former ones. We can conclude that any way of closing a TRS rule produces the same rewrite relation, so there is no harm in closing it in the way we have chosen.

EXAMPLE 3.3.8. The rules of the closure of the TRS Combinatory Logic of Example 3.3.2 are:

$$\begin{array}{rcl} \xi.\mathtt{app}(\mathtt{I})(\xi) & \to & \xi.\xi \\ \xi.\zeta.\mathtt{app}(\mathtt{app}(\mathtt{K})(\xi))(\zeta) & \to & \xi.\zeta.\xi \\ \xi.\zeta.\varsigma.\mathtt{app}(\mathtt{app}(\mathtt{app}(\mathtt{S})(\xi))(\zeta))(\varsigma) & \to & \xi.\zeta.\varsigma.\mathtt{app}(\mathtt{app}(\xi)(\varsigma))(\mathtt{app}(\zeta)(\varsigma)) \end{array}$$

where $\mathtt{app}: 0 \to 0 \to 0$ and $\mathtt{S}, \mathtt{K}, \mathtt{I}: 0$.

NOTATION 3.3.9. The rules for CL look horrible, so henceforth we will write ordinary TRS rules, in order to denote the corresponding pattern HORS rules.

REMARK 3.3.10. In this section we have translated a TRS into a pattern HORS, i.e. we have employed $\lambda^{\to}_{\bar{\eta}}$ as substitution calculus. As stated in the introduction of this chapter, this is not forced upon us. Any substitution calculus could be used, for example typed Combinatory Logic ($CL^{\to}$) (see [HS86, Ch. 14]). The rules of a TRS are then closed by applying some 'abstraction algorithm' $[x]$ to both sides of each rule, for every variable x occurring in the left-hand side of the rule. One such algorithm is inductively defined by the following rules: (see e.g. [HS86, Def. 9.20])

$$\begin{array}{rcl} [x]x & =^{\mathrm{def}} & I \\ [x]s(t) & =^{\mathrm{def}} & S([x]s)([x]t) \\ [x]s & =^{\mathrm{def}} & K(s), \text{ otherwise} \end{array}$$

where $I: \alpha \to \alpha$, $K: \alpha \to (\alpha' \to \alpha)$ and $S: (\alpha \to \alpha' \to \alpha'') \to (\alpha \to \alpha') \to (\alpha \to \alpha'')$ for type variables α, α' and α''. As an amusing/confusing example, one might encode CL as a HORS having $CL^{\to}$ for substitution calculus. The rule for $\mathtt{I}$ translates as:

$$S(S(K(\mathtt{app}))(K(\mathtt{I})))(I) \to I$$

3.3.2. Orthogonal TRSs

In this section we show that for a TRS which is orthogonal in the standard sense ([DJ90, Klo92]), its closure is a pattern HORS which is orthogonal in our sense.

As remarked in the preceding subsection, the closure of the TRS is a pattern HORS, so we only need to show that orthogonality of a TRS implies orthogonality of its closure. The two ingredients of the usual definition of orthogonality for TRSs are: *left-linearity* and *non-ambiguity*. First, we give the standard definition of linearity of a set of terms and show that taking its closure yields a set of linear patterns (Definition 3.2.20). Then, we give the standard definition of non-ambiguity of a set of terms and show that combined with left-linearity of the set, this implies simultaneity of its closure.

DEFINITION 3.3.11. Let $\mathcal{H}$ be a TRS. A $\mathcal{H}$-term is *linear* if variable occurrences are unique. A $\mathcal{H}$-rule is *left-linear* (*right-linear*) if its left-hand side (right-hand side) is linear. It is linear if it is both left- and right-linear. The TRS $\mathcal{H}$ is (left-/right-)linear if all its rules are so.

PROPOSITION 3.3.12. *If $\mathcal{H}$ is a left-linear TRS, then $\mathcal{I}(\mathcal{H})$ is a left-linear pattern HORS.*

PROOF Trivial by comparing both definitions of left-linearity (Definition 3.3.11 and Definition 3.2.20). $\odot$

Following Klop [Klo92], we define a non-overlapping (non-ambiguous) TRS as a TRS without critical pairs. Although we will present some examples of non-ambiguous and ambiguous TRSs, we assume the reader to be familiar with the concept of critical pair and just state its definition. For an excellent intuitive explanation of the idea of (absence of) critical pair see Section 3.1 of [Klo92].

DEFINITION 3.3.13. Let $\mathcal{H}$ be a TRS. Let $\aleph =^{\mathrm{def}} l \to r$ and $\beth =^{\mathrm{def}} g \to d$ be rules of $\mathcal{H}$.

1. If there exist a context $C[\,]$, a non-variable term s, and a substitution θ such that $l \equiv C[s]$ and $s^\theta \equiv g^\theta$, then g *overlaps with* l. If $C[\,]$ is the empty context, then g *root* overlaps with l, otherwise g *properly* overlaps with l.

2. If g overlaps with l as in the first item, such that θ is a 'most general unifier', then the pair of reducts $(r^\theta, C[d]^\theta)$ is called a *critical pair* obtained by *superposition* of $\beth$ on $\aleph$. If its components are identical, then it is called a *trivial* critical pair.

3. The TRS $\mathcal{H}$ is *non-ambiguous* if there are no critical pairs between $\mathcal{H}$-rules. *Ambiguous* means not non-ambiguous.

4. A TRS is *orthogonal* if it is left-linear and non-ambiguous.

THEOREM 3.3.14. *Every orthogonal TRS is confluent.*

PROOF We show that the closure $\mathcal{I}(\mathcal{H})$ of an orthogonal TRS $\mathcal{H}$ is a non-overlapping left-linear pattern HORS. Then we are done, since the rewrite relations of $\mathcal{H}$ and $\mathcal{I}(\mathcal{H})$ coincide and non-overlapping left-linear pattern HORSs are confluent by Theorem 3.2.26.

By Proposition 3.3.12 it is sufficient to show that a critical pair between translated rules $\aleph'$, $\beth'$ in $\mathcal{I}(\mathcal{H})$ correspond to critical pairs between the original rules $\mathcal{H}$ and $\mathcal{I}$ in $\mathcal{H}$. This is analogous to the case for Nipkow's HRSs to be treated in Section 3.5. $\odot$

REMARK 3.3.15. Critical pairs for term rewriting systems are defined via first-order unification. It is not a priori clear that switching to higher-order unification does not produce additional (higher-order) unifiers for first-order terms. A careful inspection of Nipkow's unification algorithm for higher-order patterns in [Nip91, App. A] reveals that first-order terms can only have first-order most general unifiers.

EXAMPLE 3.3.16. Combinatory Logic of Example 3.3.2 is orthogonal.

3.3.3. Weakly Orthogonal TRSs

In this subsection all weakly orthogonal TRSs are shown to be confluent. As in the preceding section, we first give the standard definition of weak orthogonality for a TRS and then show that this implies weak orthogonality of its closure.

DEFINITION 3.3.17. A TRS $\mathcal{H}$ is *weakly orthogonal* if $\mathcal{H}$ is left-linear and all critical pairs are trivial.

THEOREM 3.3.18. *Every weakly orthogonal TRS is confluent.*

PROOF We show that the closure $\mathcal{I}(\mathcal{H})$ of a weakly orthogonal TRS $\mathcal{H}$ is a weakly orthogonal pattern HORS. This is direct from (the proof of) Theorem 3.3.14, since (trivial) critical pairs in $\mathcal{H}$ correspond to (trivial) critical pairs in $\mathcal{I}(\mathcal{H})$. By Theorem 3.2.34 $\mathcal{I}(\mathcal{H})$ is confluent. $\odot$

This result is by no means new. It is an immediate consequence of e.g. Lemma 3.3 in [Hue80]. However, it is not evident that the proof method employed there can be generalised to the higher-order case, as is the case for our method.

The simple proof-idea of this theorem may be obfuscated by the rather heavy technical machinery, so let us explain the idea. It is well-known that the parallel moves lemma holds for orthogonal term rewriting systems. That is, one can construct the following diagram

$$\begin{tikzcd}
s_1 \arrow[r, "u_1"'] \arrow[d, "v"] & s_2 \arrow[r, "u_2"'] \arrow[d, dashed, twoheadrightarrow, "v"] & s_3 \arrow[r, twoheadrightarrow] \arrow[d, dashed, twoheadrightarrow, "v"] & s_m \arrow[d, dashed, twoheadrightarrow, "v"] \\
t_1 \arrow[r, dashed, twoheadrightarrow, "u_1"'] & t_2 \arrow[r, dashed, twoheadrightarrow, "u_2"'] & t_3 \arrow[r, dashed, twoheadrightarrow] & t_n
\end{tikzcd}$$

in which in the conclusions of the diagrams only descendants of the rewrite steps on the opposite side are contracted. The essence of this construction is, that there exists for each term s_i a set of simultaneous redexes $\mathcal{U}_i$, such that there exist complete developments $d: s_i \twoheadrightarrow s_{i+1} \twoheadrightarrow t_{i+1}$ and $d': s_i \twoheadrightarrow t_i \twoheadrightarrow t_{i+1}$ of $\mathcal{U}_i$. What problems do arise, when orthogonality is relaxed to weak orthogonality? The only problem is that the redex u_{i+1} might overlap with some redexes in the set $\mathcal{V} =^{\mathrm{def}} \{\mathcal{U}_i \lfloor u_i \rfloor\}$ of residuals of $\mathcal{U}_i$ in s_{i+1}. But then we know by weak orthogonality, that there exists some step $u' \in \mathcal{V}$ doing exactly the same as u_{i+1}, hence by starting with this step u', we obtain a complete development of $\mathcal{V}$ which 'goes through' s_{i+2} as was required. For this to work, it is needed that simultaneity of a set of redexes is preserved by performing a rewrite step. This holds because of the Develop Lemma. Moreover, one needs that if the redex u_{i+1} does not overlap with any redex in $\mathcal{V}$, then the set $\mathcal{V} \cup \{u_{i+1}\}$ is simultaneous again. For this to hold, cubicity suffices.

We present two examples of weakly orthogonal term rewriting systems. Later on, we combine the first one with PCF to obtain a weakly orthogonal HORS which is confluent again.

EXAMPLE 3.3.19. Consider the rules for *parallel or* (cf. [BCL85, Prop. 7.1.1]):

$$\begin{aligned} \mathtt{por}(\mathtt{tt}, x) &\to \mathtt{tt} \\ \mathtt{por}(x, \mathtt{tt}) &\to \mathtt{tt} \\ \mathtt{por}(\mathtt{ff}, \mathtt{ff}) &\to \mathtt{ff} \end{aligned}$$

The first rule root overlaps with the second one at position 000 (after closing the rules) and vice versa, both by the unification pair $(\Box(\mathtt{tt}),\Box(\mathtt{tt}))$. The critical pairs for these overlaps are both trivial: $(\mathtt{tt},\mathtt{tt})$. We conclude from Theorem 3.3.18 that the TRS is confluent.

EXAMPLE 3.3.20. Consider the following rules for *successor* and *predecessor*:

$$\begin{aligned} \mathtt{+1}(\mathtt{-1}(x)) &\to x \\ \mathtt{-1}(\mathtt{+1}(x)) &\to x \end{aligned}$$

The first rule overlaps with the second one at position 010 and vice versa. The unification pairs are $(\xi.\Box(\mathtt{+1}(\xi)),\xi.\mathtt{+1}(\Box(\xi)))$ and $(\xi.\Box(\mathtt{-1}(\xi)),\xi.\mathtt{-1}(\Box(\xi)))$, with respective critical pairs $(\xi.\mathtt{+1}(\xi),\xi.\mathtt{+1}(\xi))$ and $(\xi.\mathtt{-1}(\xi),\xi.\mathtt{-1}(\xi))$. We conclude from Theorem 3.3.18 that these rules are confluent.

3.3.4. Combinations of TRSs

The first major result in the investigation of modularity of confluence for term rewriting systems is due to Toyama, who showed ([Toy87]) that direct sum, i.e. disjoint union, preserves confluence. Since then, this result has been strengthened in several ways allowing more and more shared symbols between the unified systems.

In combining TRSs, two steps are to be distinguished: first, extending the alphabets of the TRSs to a common one and then, combining the rewrite relations. In the case of confluence, adding symbols to the alphabet is not troublesome, because TRSs always contain variables. Roughly speaking, these variables 'behave' in the same way as the added symbols, hence confluence is preserved by extending the alphabet. However, extending the alphabet does not preserve confluence, when rewriting is restricted to so-called *ground* terms, that is, terms without variables, as witnessed by the TRS:

$$\begin{array}{lcl} \mathtt{G}(x) & \to & \mathtt{A} \\ \mathtt{G}(x) & \to & x \end{array}$$

This TRS is *ground* confluent, since every term (built from G and A) rewrites to A. However it is not confluent, as witnessed by

$$\mathtt{B} \leftarrow \mathtt{G}(\mathtt{B}) \to \mathtt{A}$$

REMARK 3.3.21. Continuing on this train of thought, one might wonder if the number of extra symbols needed to distinguish ground confluence from confluence is bounded (in the previous case one symbol, B, sufficed). This is not so, because one can adapt the TRS:

$$\begin{array}{lcl} \mathtt{G}(x)(x)(y) & \to & \mathtt{A} \\ \mathtt{G}(y)(x)(x) & \to & \mathtt{A} \\ \mathtt{G}(x)(y)(x) & \to & \mathtt{A} \\ \mathtt{G}(x)(y)(z) & \to & \mathtt{G}(x)(y)(\mathtt{A}) \end{array}$$

which is confluent if one symbol is added, but not so if two symbols are added

$$\mathtt{A} \leftarrow \mathtt{G}(\mathtt{B})(\mathtt{K})(\mathtt{K}) \to \mathtt{G}(\mathtt{B})(\mathtt{K})(\mathtt{A})$$

to an arbitrary number of symbols.

As soon as an arbitrary number of nullary (zero order) symbols is available, adding fresh m-ary (first order) symbols makes no difference. Intuitively, this is true, because TRS-rules 'cannot look inside subterms'. Technically, it is a direct consequence of Toyama's Theorem [Toy87, Thm. 4.1].

Usually, extending the alphabet is left implicit in dealing with combinations of TRSs. The reason we have dealt with it explicitly, is that for CRSs a distinction between confluence and *meta* confluence, similar to to one between ground confluence and confluence shows up.

Once it is established that extending the alphabet does not present problems, attention can be turned to combining the rewrite relations.

THEOREM 3.3.22. *The union of two left-linear confluent TRSs such that all critical pairs between them are trivial, is confluent.*

PROOF By Theorem 3.1.54. $\odot$

As was the case in the previous subsection, this result is by no means new. It is merely meant to set the stage for its extension to the more complex rewriting systems of the next sections. The result was basically obtained by Rosen in [Ros73, Thm. 6.5] and extended by Toyama [Toy88, Cor. 3.1]). The idea is to show that the rewrite relations commute and then to apply the Lemma of Hindley-Rosen (Example 2.3.9(1)).

3.4. Klop's Combinatory Reduction Systems

Combinatory reduction systems, or CRSs for short, were introduced by Klop in his PhD thesis ([Klo80]) and were designed to combine the usual first-order format of term rewriting with the presence of bound variables as in pure λ-calculus and various typed λ-calculi. Bound variables are also present in many other rewriting systems, such as systems with simplification rules for proof normalisation. The original idea of CRSs is due to Aczel, who introduced in [Acz78] a restricted class of CRSs and, under the assumption of orthogonality, proved confluence. By orthogonality it is meant that the rules are non-ambiguous (no overlap leading to a critical pair) and left-linear (no global comparison of terms necessary).

The plan for this section is as for the preceding section. First, we present CRSs in their usual presentation. Then, we show how each CRS $\mathcal{H}$ can be made to correspond to a pattern HORS $\mathcal{I}(\mathcal{H})$ such that their rewrite relations coincide. Finally, it is shown that (weak) orthogonality of a CRS $\mathcal{H}$ implies (weak) orthogonality of $\mathcal{I}(\mathcal{H})$. The translation we present is based on the encoding of CRSs into Nipkow's HRSs (see Section 3.5) given in [OR93]. Full proofs of the properties of the translation which are used here, can be found there.

Since their introduction CRSs have been subject to some (minor) change. We copy the presentation of CRSs from [KOR93] using our notation. The most notable change compared to the CRSs as introduced in [Klo80] is the switch from an 'applicative' format to a 'functional' format. The reason for choosing the functional format is that it is closer to the usual notation for term rewriting systems.

A CRS is a pair consisting of an alphabet and a set of rewrite rules. In a CRS a distinction is made between metaterms and terms. The left- and right-hand side of a rule are metaterms, and rules act upon terms. This distinction is made in order to stress the point that a reduction rule acts as a scheme, so its left- and right-hand side are not ordinary terms.

Definition 3.4.1. The alphabet of a CRS consists of

1. a set $\mathsf{Var} = \{\xi_m . m \geqslant 0\}$ of *variables* (also written as ξ, ζ, ς, ...),
2. a set Mvar of *metavariables* $\{x^k_m . m, k \geqslant 0\}$ (here k is the *arity* of x^k_m),
3. a set of *function symbols*, each with a fixed arity,

4. a binary operator for abstraction, written as $[\cdot]\cdot$,

5. improper symbols '(', ')' and ','.

The arities k of the metavariables x_m^k can always be read off from the metaterm in which they occur — hence we will often suppress these superscripts. For example, in $\mathtt{app}([\xi]x_0(\xi), x_1)$ the x_0 is unary and x_1 is nullary.

REMARK 3.4.2. In the literature on combinatory reduction systems (see also the bibliography of [KOR93]) variables are denoted by x, y, z, ... and metavariables by Z_n^k. Since CRS variables and metavariables correspond to HORS bound variables and free variables respectively, we deviate from the usual notation in order to remain consistent with the other notations in this thesis.

DEFINITION 3.4.3. The set Mter of metaterms of a CRS with an alphabet as in Definition 3.4.1 is defined inductively as follows:

1. variables are metaterms;

2. if s is a metaterm and ξ a variable, then $[\xi]s$ is a metaterm, obtained by *abstraction*;

3. if $\mathtt{F}$ is an m-ary function symbol ($m \geqslant 0$) and $s_1, \ldots, s_m$ are metaterms, then $\mathtt{F}(s_1, \ldots, s_m)$ is a metaterm;

4. if $s_1, \ldots, s_k$ ($k \geqslant 0$) are metaterms, then $x_m^k(s_1, \ldots, s_k)$ is a metaterm (in particular the x_m^0 are metaterms).

Note that metavariables x_m^{k+1} with arity > 0 are not metaterms; they need arguments. Metaterms without metavariables are terms. The set of terms is denoted as Ter.

A (meta)term is *closed* if every variable occurrence is bound, where the notion of "bound" is as in λ-calculus: an occurrence of a variable ξ is *bound* if it is in the scope of an abstractor $[\xi]$. It is *free* otherwise.

REMARK 3.4.4. In HORSs the 'naming problems' are confined to an isolated part: the substitution calculus. In the definition of CRSs this is not the case, variable naming problems are all over the place. Sloppywise, we adopt the following convention in this and the next section: If a term occurs in a certain mathematical context (e.g. definition, proof), then in this term all bound variables are chosen to be different from the free variables. This is Barendregt's *Variable Convention* ([Bar84, p. 26], see also [Fre62, p. 15]).

DEFINITION 3.4.5. A *rewrite rule* in a CRS is a pair (l,r), written as $l \to r$, where l and r are metaterms such that:

1. l and r are closed metaterms;

2. s has the form $\mathtt{F}(s_1, \ldots, s_m)$;
3. the metavariables x^k_m that occur in r also occur in l;
4. the metavariables x^k_m in l occur only in the form $x^k_m(\xi_1, \ldots, \xi_k)$, where the ξ_i $(i = 1, \ldots, k)$ are variables. Moreover, the ξ_i are pairwise distinct.

EXAMPLE 3.4.6. In CRS notation, the rule of beta-reduction in lambda calculus:

$$\mathsf{app}([\xi]x(\xi), y) \rightarrow_{\mathsf{beta}} x(y)$$

Application is expressed by the binary symbol app. Similarly, the rule of eta-reduction is:

$$[\xi]\mathsf{app}(y, \xi) \rightarrow_{\mathsf{eta}} y$$

In these rules, x is a unary metavariable and y a nullary metavariable.

It requires some subtlety to extract from the rewrite rules the actual rewrite relation that they generate. The reason for this is that substitution in CRSs is performed by replacing a metavariable by a (special form of a) λ-term, and by reducing, in the term obtained by this replacement, all *residuals* of β-redexes that are present in the initial term, i.e. by performing a development (or expanding `let`-constructs). The well-known result in λ-calculus that all developments are finite, guarantees that the substitution is well-defined.

In order to define valuations for CRSs we first introduce a new concept: the so-called *substitutes* (cf. [Kah93]). An m-ary substitute is a map of the form $\boldsymbol{\lambda}(x_1, \ldots, x_m).s$, where s is a term and $(x_1, \ldots, x_m)$ a tuple of m distinct variables. A substitute $\boldsymbol{\lambda}(x_1, \ldots, x_m).s$ can be applied to an m-tuple of terms $(t_1, \ldots, t_m)$, yielding s with $x_1, \ldots, x_m$ simultaneously replaced by $t_1, \ldots, t_m$ respectively:

$$(\boldsymbol{\lambda}(x_1, \ldots, x_m).s)(t_1, \ldots, t_m) =^{\mathrm{def}} [x_1 \mapsto t_1, \ldots, x_m \mapsto t_m]s$$

An *valuation* σ consists of assigning m-ary substitutes to m-ary metavariables:

$$\sigma(x) =^{\mathrm{def}} \boldsymbol{\lambda}(x_1, \ldots, x_m).s$$

It is extended to a mapping from terms to terms in the following way:

$$\begin{array}{rcl}
\xi^\sigma & =^{\mathrm{def}} & \xi \\
([\xi]s)^\sigma & =^{\mathrm{def}} & [\xi]s^\sigma \\
\mathtt{F}(t_1, \ldots, t_m)^\sigma & =^{\mathrm{def}} & \mathtt{F}({t_1}^\sigma, \ldots, {t_m}^\sigma) \\
x(t_1, \ldots, t_m)^\sigma & =^{\mathrm{def}} & \sigma(x)({t_1}^\sigma, \ldots, {t_m}^\sigma)
\end{array}$$

REMARK 3.4.7. Note that the result of applying $\sigma(x)$ to $({t_1}^\sigma, \ldots, {t_m}^\sigma)$ in the last clause is indeed a term. Substitution is defined as a one stage process as in [KOR93]. It could also be defined in two stages: first replace the metavariables by the terms assigned to them, and then explicitly develop the created redexes. This would yield a presentation closer to the one of HRSs and HORSs.

A variable in an instance of a metavariable should be bound only if it is bound in the occurrence of the metavariable. Unintended bindings would occur for instance in $([\xi]x)^\sigma$ if $\sigma(x) =^{\mathrm{def}} \xi$, but this is prevented by the Variable Convention of Remark 3.4.4.

Contexts are defined as for TRSs: a *context* is a term with an occurrence of a special symbol $[\,]$ called *hole*. As noted before, variables in a term s may be bound when put inside a context $C[\,]$, but this is intentional. For example, ξ can be put inside the context $[\xi][\,]$ resulting in the term $[\xi]\xi$.

REMARK 3.4.8. The main difference between a context $C\square$ as introduced for HORSs before and a CRS context $C[\,]$ is that in the former one may only fill the hole by closed terms, while in the latter also open terms may be filled in. There is no interaction between the contents of a closed box $\square$ and the context, while the contents of an open box $[\,]$ interact with their environment via the bound variables.

DEFINITION 3.4.9.
1. Let $l \to r$ be a rewrite rule, and σ be a valuation. Then $l^\sigma \to r^\sigma$ is called a *contraction*. The term l^σ is called a *redex*.
2. Let $l^\sigma \to r^\sigma$ be a contraction, and $C[\,]$ a context. Then $C[l^\sigma] \to C[r^\sigma]$ is called a *rewrite step*.

EXAMPLE 3.4.10. We reconstruct a step according to the beta-reduction rule of Example 3.4.6: Let the valuation $\sigma(x) =^{\mathrm{def}} \boldsymbol{\lambda}(z).s(z)(z)$, $\sigma(y) =^{\mathrm{def}} t(r)$ be given. Then we have the reduction step:

$$\begin{array}{rcl}
\mathsf{app}([\xi]x(\xi), y)^\sigma & \equiv & \mathsf{app}([\xi]x^\sigma(\xi), y^\sigma) \\
& \equiv & \mathsf{app}([\xi](\boldsymbol{\lambda}(z).s(z)(z))\xi, t(r)) \\
& \equiv & \mathsf{app}([\xi]s(\xi)(\xi), t(r)) \\
& \to_{\mathsf{beta}} & \\
x(y)^\sigma & \equiv & x^\sigma(y^\sigma) \\
& \equiv & (\boldsymbol{\lambda}(z).s(z)(z))t(r) \\
& \equiv & s(t(r))(t(r))
\end{array}$$

Note that in the CRS format there is no need for explicitly requiring that some variables are not allowed to occur in instances of metavariables. For instance, in $[\xi]x$, an instance of x is not allowed to contain free occurrences of ξ. In lambda calculus such a requirement cannot be made in the system itself; it has to be stated in the metalanguage, as is done for the eta-rule. In this sense the CRS formalism is more expressive than that of lambda calculus.

3.4.1. From CRS to HORS

In this section we give a translation from CRSs into pattern HORSs along the lines of the translation from TRSs into pattern HORSs of Section 3.3.1. This translation is based on the one presented in [OR93] from CRSs into HRSs, the only difference being the additional closure of the rules as for the TRS case.

Compared to the preceding section on TRSs we have to deal with one extra construct in the translation: the binary symbol $[\cdot]\cdot$ for abstraction. Of course, this symbol should be translated into the abstraction symbol $\cdot.\cdot$ of a HORS, but a direct translation presents typing problems. For example, the CRS term $s =^{\mathrm{def}} \mathtt{F}([\xi]\xi)$ is perfectly legitimate, but its 'translation' $\mathcal{I}(s) =^{\mathrm{def}} \mathtt{F}(\xi.\xi)$ cannot be simply typed, because $\mathtt{F}\colon o \to o$ and $\xi.\xi\colon \sigma \to \sigma$ for some type σ. The solution is to collapse the functional type of $\xi.\xi$ by means of a special symbol $\mathtt{abs}\colon (o \to o) \to o$ into the base type. The abstraction operator $[\cdot]\cdot$ can then be viewed as an abbreviation of $\mathtt{abs}(\cdot.\cdot)$ and the base type o can be thought of as the set of all terms. For example, $\mathcal{I}(s) =^{\mathrm{def}} \mathtt{F}(\mathtt{abs}(\xi.\xi))\colon o$.

Definition 3.4.11. The alphabet $\mathcal{A}$ of the closure $\mathcal{I}(\mathcal{H})$ of a CRS $\mathcal{H}$ with an alphabet as above consists of:

1. if ξ is a variable, then $\mathcal{I}(\xi)$ is a bound variable of type o,
2. if x^k is a k-ary metavariable, then $\mathcal{I}(x)$ is a free variable of type $o \to \ldots \to o$ of arity k,
3. if $\mathtt{F}$ is a m-ary function symbol, then $\mathcal{I}(\mathtt{F})\colon o \to \ldots \to o$ of arity m,
4. the symbol $\mathtt{abs}\colon (o \to o) \to o$.

The translation of a term s is inductively defined as for TRSs:

1. if s is ξ, then $\mathcal{I}(\xi)$,
2. if s is $[\xi]s'$, then $\mathtt{abs}(\mathcal{I}(\xi).\mathcal{I}(s'))$,
3. if s is $\mathtt{F}(s_1, \ldots, s_m)$, then $\mathcal{I}(\mathtt{F})(\mathcal{I}(s_1)) \ldots (\mathcal{I}(s_m))$,
4. if s is $x(s_1, \ldots, s_m)$, then $\mathcal{I}(x)(\mathcal{I}(s_1)) \ldots (\mathcal{I}(s_m))$.

Again, the translation $\mathcal{I}$ is required to be injective on symbols and then one easily checks the translation to be injective on terms as well and to yield terms, i.e. $\beta\overline{\eta}$-normal forms. The translation of terms is naturally extended to rules and sets of rules. The *closure* of a CRS $\mathcal{H}$ having set of rewrite rules $\mathcal{R}$, is the full sub-HORS $\mathcal{I}(\mathcal{H})$ of $\langle \mathcal{A}, \lambda^{\rightarrow}_{\overline{\eta}}, \overline{\mathcal{I}(\mathcal{R})}\rangle$, obtained by restricting $\mathbb{T}(\mathcal{A})$ to the set $\mathcal{I}(\mathsf{Ter})$ of translated terms. Here $\overline{\mathcal{I}(\mathcal{R})}$ is the closure of $\mathcal{I}(\mathcal{R})$ obtained by closing each rule with respect to the sequence of free variables in its left-hand side, from left to right.

Remark 3.4.12. Applying the translation $\mathcal{I}$ to the CRS term ξ results in the *raw* term $\mathcal{I}(\xi)$. To be precise, free variables in a CRS term should be translated into free variables by the translation. On the other hand, one could argue that CRS terms should be defined as closed metaterms having only nullary metavariables.

The remarks made for the first-order case (page 107) apply also in this case, but need some elaboration. The set of translated terms $S =^{\mathrm{def}} \mathcal{I}(\mathsf{Ter})$ is a subset of the set of second-order terms. More precisely, all the symbols occurring in a translated term have order zero or one, except for the symbol `abs` which has order two. As soon as the order of a term is increased by an abstraction, say $\xi.s\colon o \to o$ it is collapsed again to base type by applying `abs` to it, hence we call S the set of *collapsed terms*. Each rule is closed with respect to the (translation of) the metavariables in its left-hand side. By the condition on CRSs that rules are closed and that the metavariables in the right-hand side of a rule are contained in the ones of the left-hand side, this procedure produces closed HORS rules. By the restrictions put on the left-hand sides of CRS rules, their translations are patterns (cf. Definition 3.2.16 and Definition 3.4.5).

Proposition 3.4.13. *Let $\mathcal{H}$ be a CRS, let s be a term in $\mathcal{H}$ and let θ be a substitution. Then $\mathcal{I}(s^\theta) \equiv \overline{\mathcal{I}(s)}(\mathcal{I}(x_1{}^\theta))\dots(\mathcal{I}(x_m{}^\theta))\downarrow_{\beta\bar{\eta}}$, where $\overline{\mathcal{I}(s)}$ is the x_1, …, x_m-closure of $\mathcal{I}(s)$, and $\mathsf{Fvar}(s) \subseteq \{x_1, \dots, x_m\}$.*

Proof As for Proposition 3.4.13. $\odot$

Rewriting is restricted to collapsed terms. As in the first-order case, the subject-reduction property for this set must be shown.

Proposition 3.4.14. *Let s be a collapsed term in $\mathcal{I}(\mathcal{H})$. If u is an m-ary redex in s, then the context $C\Box$ into which u is extracted does not contain created abstractions, and the hole occurs as $\Box(\xi_1.\dots\xi_{n_1}.\mathcal{I}(s_1))\dots(\xi_1.\dots\xi_{n_m}.\mathcal{I}(s_m))$.*

Proof As in the proof of Proposition 3.3.6, one easily shows that the extraction cannot create λ-abstractions of order higher than one in the context $C\Box$, because this would imply the presence of a redex in $C\Box$. Let the hole occur as $\Box(s'_1)\dots(s'_m)$ in $C\Box$. Then reducing $\boxed{l}(s'_1)\dots(s'_m)$ to $\beta\bar{\eta}$-normal form must result in a collapsed term. By the conditions on patterns, each s'_i is applied to a sequence of distinct variables in the reduction to normal form. Hence s'_i must be of the form $\xi_1.\dots\xi_{n_i}.s''_i$ and the resulting normal form is a collapsed term which is a renaming of s''_i, so s''_i must be of the form $\mathcal{I}(s_i)$. $\odot$

From this we have that if $s \twoheadleftarrow_{\beta\bar{\eta}} C\boxed{\mathcal{I}(l)} \to C\boxed{\mathcal{I}(r)} \twoheadrightarrow_{\beta\bar{\eta}} t$ is a rewrite step in $\mathcal{I}(\mathcal{H})$ starting from the collapsed term s, then t is a collapsed term again.

Lemma 3.4.15. *Let $\mathcal{H}$ be a CRS, then $s \to_{\mathcal{H}} t \iff \mathcal{I}(s) \to_{\mathcal{I}(\mathcal{H})} \mathcal{I}(t)$.*

Proof As for Lemma 3.3.7, using the preceding propositions. (For a detailed proof of this lemma, see [OR93, Thm. 3.8,3.11]). $\odot$

The lemma expresses that a CRS and its associated pattern HORS are identical as far as its (operational) semantical, i.e. rewriting, aspects are concerned, so it is safe to identify them. However, it still must be shown that the syntactical aspects of both frameworks correspond nicely. This is the topic of the next subsection.

REMARK 3.4.16. The encoding of CRSs as HORSs is essentially based on an encoding of untyped λ-calculus in $\lambda^{\rightarrow}$-calculus. In [OR93], it was noted that the same idea of the encoding can be used to give a very simple proof of the Finite Developments theorem for λ-calculus, when termination of β-reduction in $\lambda^{\rightarrow}$-calculus is known. Define the translation $\mathcal{I}$ as follows:

$$
\begin{array}{rcl}
\mathcal{I}((\xi.s)t) & =^{\mathrm{def}} & (\xi.\mathcal{I}(s))\mathcal{I}(t) \\
\mathcal{I}(\xi) & =^{\mathrm{def}} & \xi\colon o \\
\mathcal{I}(\xi.s) & =^{\mathrm{def}} & \mathtt{abs}(\xi.\mathcal{I}(s)) \\
\mathcal{I}(s(t)) & =^{\mathrm{def}} & \mathtt{app}(\mathcal{I}(s))(\mathcal{I}(t))
\end{array}
$$

where the first rule has priority over the last one, and $\mathtt{abs}\colon (o \rightarrow o) \rightarrow o$ and $\mathtt{app}\colon o \rightarrow (o \rightarrow o)$ are fresh symbols (for example variables). It is easy to show that this translation gives simply typable terms and preserves β-redexes. Moreover, there is an exact correspondence between the development of a λ-term s and the reduction of its translation $\mathcal{I}(s)$. From $SN(\lambda^{\rightarrow})$, it can be concluded that every development must be finite. This proof seems to be not very well-known[1], but cf. [Kri90, pp. 45–49] for an encoding of developments in a typed λ-calculus with intersection types, due to M. Parigot. It would be interesting to see whether similar encodings can be found for the labelled λ-calculi as considered by Klop in [Klo80, Sec. 3].

3.4.2. Orthogonal CRSs

In this subsection we show that for a combinatory reduction system which is orthogonal in the standard sense ([Klo80, KOR93]), its closure is a pattern HORS which is orthogonal in our sense.

The method is the same as for the first-order case. First, it is shown that linear CRS metaterms translate to linear patterns and then it is shown that non-ambiguous CRSs translate into non-overlapping pattern HORSs. The situation is somewhat simpler than for TRSs in the latter respect, because orthogonality of CRSs is defined via absence of overlap, while orthogonality of TRSs was defined via absence of critical pairs. In fact, we do not know of published papers on critical pairs for CRSs.

DEFINITION 3.4.17. Let $\mathcal{H}$ be a CRS.

1. A $\mathcal{H}$-rule is *left-linear* if metavariables occur at most once in its left-hand side. The CRS $\mathcal{H}$ is left-linear if all its rules are so.

2. The CRS $\mathcal{H}$ is *non-overlapping* if the following holds: Let σ be a valuation. If the redex $l_i{}^{\sigma}$ for rule $\aleph_i$ contains an $\aleph_j$-redex $(i \neq j)$, then this $\aleph_j$-redex must be already contained in $x(\xi_1)\ldots(\xi_k)^{\sigma}$, for some subterm $x(\xi_1)\ldots(\xi_k)$ of l_i. Likewise if the $\aleph_i$-redex properly contains an $\aleph_i$-redex.

[1] As H. Barendregt pointed out, this proof was also found by R. Statman.

3. $\mathcal{H}$ is *orthogonal* if it is non-overlapping and left-linear.

THEOREM 3.4.18. *Every orthogonal CRS is confluent.*

PROOF We show that the closure $\mathcal{I}(\mathcal{H})$ of an orthogonal TRS $\mathcal{H}$ is a non-overlapping left-linear pattern HORS. Then we are done, since the rewrite relations of $\mathcal{H}$ and $\mathcal{I}(\mathcal{H})$ coincide and non-overlapping left-linear pattern HORSs are confluent by Theorem 3.2.26.

1. By the condition that metavariables occur at most once in a left-hand side l and each occurrence is of the form $x(\xi_1)\ldots(\xi_m)$, it follows that the closure of its translation $\overline{\mathcal{I}(l)}$ is a linear pattern.

2. It is sufficient to show that a critical pair between translated rules $\aleph'$, $\beth'$ gives rise to an overlap between the original rules $\aleph$, $\beth$.

$\odot$

REMARK 3.4.19. A priori, it might be the case (cf. also Remark 3.3.15) that Nipkow's unification algorithm generates higher-order unifiers for collapsed terms. A careful inspection of his algorithm shows, that only variables of order at most one can be introduced in the course of the algorithm, starting from collapsed patterns. From this it follows, that every critical peak for two metaterms is a metaterm again.

Much more information can be distilled from Theorem 3.2.26. The CRS satisfies the Finite Developments theorem ([Klo80, Thm. II.4.15]). This follows directly from applying the theory for orthogonal residual rewriting systems of Section 2.4 to the orthogonal pattern HORS associated to the CRS.

EXAMPLE 3.4.20. Both the beta- and the eta-rule of Example 3.4.6 are orthogonal on their own, but the combination of the two rules is not orthogonal.

EXAMPLE 3.4.21. All *Interaction Systems* as defined by Laneve in [Lan93, Sec. 3.3] are orthogonal CRSs. Our notion of permutation equivalence via RRSs, coincides with his direct one ([Lan93, Sec. 3.4]).

EXAMPLE 3.4.22. In this example we present the rewrite rules for Plotkin's PCF as described in [BCL85]. We forget about the restrictions on term formation in [BCL85], because they are not required for orthogonality and the restrictions are preserved by rewriting.

1. The operator symbols of PCF are:

   (a) `m`, for any natural number m,

   (b) `tt`, `ff` (truth values),

   (c) `+1`, `-1` (successor and predecessor),

(d) `0?` (test for zero),
(e) `if` (conditional),
(f) `Y` (fixed-point operator).

All these operators are nullary. Furthermore, there is a binary symbol `app` for application.

2. The rewrite rules of PCF are:

$$\begin{array}{rcl}
\mathtt{app}(\mathtt{+1},\mathtt{m}) & \to_{\mathsf{succ}} & \mathtt{m+1} \\
\mathtt{app}(\mathtt{-1},\mathtt{m+1}) & \to_{\mathsf{pred}} & \mathtt{m} \\
\mathtt{app}(\mathtt{0?},\mathtt{0}) & \to_{\mathsf{zero1}} & \mathtt{tt} \\
\mathtt{app}(\mathtt{0?},\mathtt{m+1}) & \to_{\mathsf{zero2}} & \mathtt{ff} \\
\mathtt{app}(\mathtt{app}(\mathtt{app}(\mathtt{if},\mathtt{tt}),x),y) & \to_{\mathsf{cond1}} & x \\
\mathtt{app}(\mathtt{app}(\mathtt{app}(\mathtt{if},\mathtt{ff}),x),y) & \to_{\mathsf{cond2}} & y \\
\mathtt{app}([\xi]x(\xi),y) & \to_{\mathsf{beta}} & x(y) \\
\mathtt{app}(\mathtt{Y},x) & \to_{\mathsf{fix}} & \mathtt{app}(x,\mathtt{app}(\mathtt{Y},x))
\end{array}$$

This CRS is orthogonal since there are no overlaps between rules.

Again as in the first-order case, the confluence result for orthogonal CRSs is by no means new. Since CRSs are less well-known than TRSs we give a short comparison of our proof with the proofs which can be found in the literature.

Analogous to the situation in λ-calculus, there are essentially two ways for proving confluence of orthogonal CRSs. A fast one via the so-called Tait & Martin-Löf method and a slow one via the Finite Developments theorem. The latter one is slow, because more information is carried along the proof giving a result stronger than just confluence.

Confluence for CRSs was first proven by Klop in his PhD thesis using the slow method, in two stages. He first proved confluence for CRSs by a weak version of the Finite Developments theorem ([Klo80, Thm. II.3.11]) and then extended this to the Finite Developments theorem ([Klo80, Thm. II.4.15]) by the method of 'reductions with memory', analogous to Nederpelt's method of 'reductions with scars' ([Ned73]). The method is in essence a reduction from a proof of strong normalisation to a proof of weak normalisation. First, it is shown that each development can be simulated by a reduction in which redexes are memorised, that is, no subterms are erased. Then, by a generalisation of the result of Church ([Chu41, p. 26]) that for λI-calculus (which is non-erasing) weak and strong normalisation coincide, it is shown that the non-erasing reduction must be finite and hence the development must be so.

REMARK 3.4.23. The method of reductions with memory as described by Klop in [Klo80, Sec. II.4] is not completely accurate. Problems crop up in the treatment of CRSs having rules such as

$$\mathtt{NO}([\xi]x) \to \mathtt{GO}$$

One can think of this rule as a 'no-occur' check rule. The rule cannot be applied to a term $\mathtt{NO}([\xi]s)$ until all occurrences of the bound variable ξ in s have been removed. This rule is similar to the eta-rule of lambda calculus. The point is that in Klop's method each subterm of a term is 'memorised'. In particular, occurrences of the bound variable ξ in s are never erased, preventing a correct simulation. For example, the reduction $\mathtt{NO}([\xi]\mathtt{NO}([\zeta]\xi)) \rightarrow \mathtt{NO}([\xi]\mathtt{GO}) \rightarrow \mathtt{GO}$ cannot be simulated. The first step translates to the step $\mathtt{NO}([\xi]\mathtt{NO}([\zeta]\xi)) \rightarrow \mathtt{NO}([\xi][\mathtt{GO}, \mathtt{NO}^*([\zeta]\xi)])$ in which the redex $\mathtt{NO}([\zeta]\xi)$ has been memorised as $[\ , \mathtt{NO}^*([\zeta]\xi)]$. Now we are stuck, because the reduction rule cannot be applied due to the occurrence of ξ in the memory part. Our memory is in the way. Note however that the reduction which is simulated is not a development. We think that restricting attention to simulating developments solves the problem. See the papers by Khasidashvili (e.g. [Kha92, Kha93]) for such an approach.

Aczel's version ([Acz78]) of the fast method was employed by Van Raamsdonk in [Raa93] (see also [KOR93]) to obtain a short proof of confluence for orthogonal CRSs. The Tait & Martin-Löf method is based on an inductive definition of a development of a set of redexes. Confluence follows from the diamond property for developments in an orthogonal CRS, which is shown by induction on the definition. Aczel's method as described in [Raa93] is based on an inductive definition of *superdevelopments*. In a superdevelopment not only redexes present in the initial term, but also redexes which are 'created upward' in a development may be contracted. The diamond property is then shown to hold for superdevelopments in orthogonal CRSs.

Remark 3.4.24. It is known that developments of an *arbitrary* set of redexes do not satisfy the diamond property for weakly orthogonal CRSs, because descendants cannot be defined properly for such systems (see [Klo80, Ch. III] for a way to circumvent this problem in the case of lambda calculus with beta- and eta-rule). But what about superdevelopments? The following example in lambda calculus with beta- and eta-rule shows that the diamond property also fails for superdevelopments. Consider the term $\mathsf{app}([\xi]\mathsf{app}([\zeta]\mathsf{app}(\zeta,\xi),I),x)$, and the superdevelopment starting from s:

$$\begin{array}{rcl}
\mathsf{app}([\xi]\underline{\mathsf{app}([\zeta]\mathsf{app}(\zeta,\xi),I)},x) & \rightarrow_{\mathsf{beta}} & \mathsf{app}(\underline{[\xi]\mathsf{app}(I,\xi)},x) \\
& \rightarrow_{\mathsf{eta}} & \underline{\mathsf{app}(I,x)} \\
& \rightarrow_{\mathsf{beta}} & x
\end{array}$$

where $I =^{\mathrm{def}} [\varsigma]\varsigma$. This is a superdevelopment, because only 'upward created' redexes are contracted in the reduction (the underlined redexes 'move to the left'). On the other hand, the lambda term s can also be superdeveloped as:

$$\underline{\mathsf{app}([\xi]\mathsf{app}([\zeta]\mathsf{app}(\zeta,\xi),I),x)} \quad \rightarrow_{\mathsf{beta}} \quad \mathsf{app}([\zeta]\mathsf{app}(\zeta,x),I)$$

The terms in which the two developments end, can not be joined in one superdevelopment step. The first one is a normal form, and from $\mathsf{app}([\zeta]\mathsf{app}(\zeta,x),I)$, the only superdevelopment which can be performed is:

$$\underline{\mathtt{app}([\zeta]\mathtt{app}(\zeta, x), I)} \quad \to_{\mathsf{beta}} \quad \mathtt{app}(I, x)$$

The beta redex $\mathtt{app}(I, x)$ cannot be contracted in this superdevelopment, because it was 'downward created'. Another superdevelopment is needed to reach the common reduct x from it.

The method we have employed to prove FD is close to Klop's method. The first part of his proof roughly corresponds to our Development Lemma 3.1.43, but in his extension from the Church-Rosser property to the full fledged Finite Developments theorem he had to resort to the 'coding trick' described above. We like to think of it as coding up a proof that inside-out normalisation is equivalent to strong normalisation for orthogonal systems in the system itself. It is a beautiful trick, but nevertheless a trick in our opinion. Our proof is technical, but in principle just a matter of technique.

Recently, an elegant abstract version of the Finite Developments theorem was obtained by Melliès ([Mel93]). He presents four axioms on the way in which redexes are duplicated guaranteeing finiteness of developments. Then, according to Proposition 2.4.16, proving FD is reduced to proving that the considered rewriting system has elementary diagrams (see Definition 2.4.13), which is expressed by his fifth axiom. The abstract conditions on duplication of redexes are satisfied by both orthogonal TRSs and CRSs, and should be applicable to any calculus having a 'block structured' binding mechanism. Compared to our method the axioms in his paper are quite simple, but simple axioms are not always easy to show. We leave it to further research to compare both methods.

3.4.3. Weakly Orthogonal CRSs

In this subsection all weakly orthogonal CRSs are shown to be confluent, thereby solving a problem which was raised in [DJK93, Problem 61].

As in the preceding section, we first give the standard definition of weak orthogonality for a CRS and then show that this implies weak orthogonality of its closure.

DEFINITION 3.4.25. A CRS $\mathcal{H}$ is said to have *trivial* overlaps, if the following holds: Let σ be a valuation. If the redex ${l_i}^\sigma$ for rule $\aleph_i$ contains an $\aleph_j$-redex, then either

1. this $\aleph_j$-redex is contained in a substitution instance $x(\xi_1)\dots(\xi_k)^\sigma$, of some subterm $x(\xi_1)\dots(\xi_k)$ occurring in l_i (the redexes do not overlap), or

2. contracting either of the redexes produces the same result (the redexes have a trivial overlap).

A CRS $\mathcal{H}$ is *weakly orthogonal*, if it is left-linear and has trivial overlaps.

THEOREM 3.4.26. *Every weakly orthogonal CRS is confluent.*

PROOF We show that the closure $\mathcal{I}(\mathcal{H})$ of a weakly orthogonal CRS $\mathcal{H}$ is a weakly orthogonal pattern HORS. This is direct from (the proof of) Theorem 3.4.18, since trivial overlaps in $\mathcal{H}$ correspond to trivial overlaps in $\mathcal{I}(\mathcal{H})$. By Theorem 3.2.34 $\mathcal{I}(\mathcal{H})$ is confluent. $\odot$

EXAMPLE 3.4.27. The CRS of Example 3.4.6 is weakly orthogonal, hence confluent. (This example will be treated in more detail in the next section on HRSs.)

EXAMPLE 3.4.28. The combination of PCF of Example 3.4.22 with parallel or of Example 3.3.19 gives a weakly orthogonal CRS.

The above proof of confluence for weakly orthogonal CRSs was obtained via the slow method, i.e. via FD. One might wonder whether there is also a more direct proof à la Tait & Martin-Löf. As Remark 3.4.24 shows, developments of arbitrary sets of redexes present problems. Nevertheless, developments of *simultaneous* sets of redexes do satisfy the diamond property. This was shown by Van Raamsdonk ([OR94]).

3.4.4. Combinations of CRSs

In this subsection weakly orthogonal combinations of CRSs are shown to be confluent, thereby solving a problem which was raised by the author in [DJK93, Problem 62].

THEOREM 3.4.29. *Let $\mathcal{H}$, $\mathcal{J}$ be left-linear confluent CRSs on the same alphabet having sets of rules $\mathcal{R}$ and $\mathcal{S}$. The union $\mathcal{H} \cup \mathcal{J}$ is confluent if the rules of $\mathcal{R}$ are weakly non-overlapping with those in $\mathcal{S}$.*

PROOF Direct from Theorem 3.1.54. $\odot$

EXAMPLE 3.4.30. In [Mül92] it is proved that the combination of the beta-rule with a left-linear confluent algebraic system, such that its rules do not contain *active* variables, i.e. no subterms of the form $x(s)$, is confluent again. The condition that the algebraic rules do not contain active variables guarantees that they are orthogonal to the beta-rule, so the result follows from Theorem 3.4.29.

EXAMPLE 3.4.31. The confluence result of Example 3.4.28 can also be obtained by applying Theorem 3.4.29 to PCF and parallel or.

One might think that the restriction that $\mathcal{H}$ and $\mathcal{J}$ are CRSs on the same alphabet can be lifted. This is not the case since confluence for CRSs was defined on terms, not on metaterms. Just like in the first-order case, where a ground confluent TRS need not be confluent, a CRS might be confluent but not *meta confluent*, i.e. not confluent on metaterms, as shown by the following rules:

$$\begin{array}{rcl} \mathtt{G}(\mathtt{G}(x)) & \to & \mathtt{G}(x) \\ \mathtt{G}([\xi]x(\xi)) & \to & \mathtt{G}(x(\mathtt{A})) \\ \mathtt{G}([\xi]x(\xi)) & \to & \mathtt{G}([\xi]x(x(\xi))) \end{array}$$

The first two rules are confluent because they are locally confluent and terminating, and the last one can be shown to be redundant for terms by induction on the structure of the term substituted for x, so the CRS is confluent. Adding nullary (zero order) operators to the CRS doesn't influence its confluence, but adding a unary one does. $\mathtt{G}(\mathtt{H}(\mathtt{A})) \leftarrow \mathtt{G}([\xi]\mathtt{H}(\xi)) \to \mathtt{G}([\xi]\mathtt{H}(\mathtt{H}(\xi))) \to \mathtt{G}(\mathtt{H}(\mathtt{H}(\mathtt{A})))$. We conclude that the CRS is not meta confluent. For orthogonal systems there is no problem, since orthogonality is preserved when operator symbols are added to the alphabet. Orthogonal CRSs are meta confluent.

Remark 3.4.32. A similar difference shows up when considering other properties of CRSs. For example, the notions of *SN* and *meta SN* differ, as shown by the following (admittedly contrived) CRS rule:

$$\mathtt{F}([\xi][\zeta]x(\xi,\zeta)) \quad \to \quad x([\xi][\zeta]x(\zeta,\xi),[\xi]\xi)$$

This CRS is *SN* (exercise, consider three different kinds of substitutions for x), but not so if a binary function symbol G is added:

$$\begin{array}{rcl} \mathtt{F}([\xi][\zeta]\mathtt{G}(\mathtt{F}(\xi),\mathtt{F}(\zeta))) & \to & \mathtt{G}(\underline{\mathtt{F}([\xi][\zeta]\mathtt{G}(\mathtt{F}(\zeta),\mathtt{F}(\xi)))},\mathtt{F}([\xi]\xi)) \\ & \to & \mathtt{G}(\mathtt{G}(\mathtt{F}([\xi]\xi),\underline{\mathtt{F}([\xi][\zeta]\mathtt{G}(\mathtt{F}(\xi),\mathtt{F}(\zeta)))}),\mathtt{F}([\xi]\xi)) \\ & \to & \ldots \end{array}$$

The example crucially relies on the fact that there are no binary function symbols in the original CRS. As soon as there is one such symbol, the difference between *SN* and meta *SN* disappears.

The modularity result above is about combinations of left-linear CRSs. What about non left-linear CRSs? In general, the problem with non left-linear rules is that they test for syntactic equality of their subterms. If such a 'tested' subterm is not in normal form, then equality is lost if different rewrites are performed in different 'tested' occurrences of the subterm. What is needed is that any number of rewrites starting from such a term can be joined again, that is, a rewrite cofinal in the rewrite graph below the term (see Definition 2.2.7).

In the case of a normalising system, one can take any rewrite to normal form for the cofinal rewrite. This approach is taken in many papers on the combination of first-order rewriting systems with some kind of lambda calculus and possibly some higher-order rules. See [BF] and its bibliography for many results.

In the case of the disjoint union of two confluent TRSs, one does not have any normalisation properties, so the only way to go is to construct the cofinal rewrite. This is not easy, but it can be done as was shown by Toyama [Toy87] (see also [KMTV91]).

3.5. Nipkow's Higher-order Rewrite Systems

We first recall the definition of Higher-order Rewrite System (HRS) as introduced by Nipkow in [Nip91], following the presentation of [Nip93], but using our terminology.

DEFINITION 3.5.1. 1. An alphabet $\mathcal{A}$ is an alphabet for a HORS having $\lambda_{\overline{\eta}}^{\rightarrow}$ as substitution calculus. The set $\mathbb{T}(\mathcal{A})$ of *terms* is the same as in the case of such a HORS, i.e. a term is a preterm in $\beta\overline{\eta}$-normal form.

2. A term s is called a (*higher-order*) *pattern* if every occurrence of a free variable x is in a subterm $x(s_1)\dots(s_m)$ of s such that $s_1{\downarrow_\eta},\dots,s_m{\downarrow_\eta}$ is a list of distinct bound variables.

3. Contexts and substitutions are defined as for TRSs, that is, every context $C[\,]$ contains exactly one occurrence of a hole $[\,]$ and a substitution θ maps variables to terms and acts homomorphically:

$$\begin{aligned} (s(t))^\theta &\equiv s^\theta(t^\theta) \\ (\xi.s)^\theta &\equiv \xi.s^\theta \end{aligned}$$

Because HRS rules consist of pairs of terms containing free variables, we assume the Variable Convention as in the preceding section (see Remark 3.4.4).

DEFINITION 3.5.2. A *rewrite rule* is a pair $l \to r$ such that l is a pattern but not η-equivalent to a free variable, l and r are terms of the same base type, and $\mathsf{Fvar}(l) \supseteq \mathsf{Fvar}(r)$. A *Higher-order Rewrite System* (for short: *HRS*) is a set of rewrite rules. An HRS $\mathcal{H}$ induces a relation $\to_{\mathcal{H}}$ on terms:

$$s \to_{\mathcal{H}} t =^{\mathrm{def}} \exists l \to r \in \mathcal{H}.\exists\theta.\exists C[\,].s \equiv C[l^\theta{\downarrow_\beta}] \;\&\; t \equiv C[r^\theta{\downarrow_\beta}]$$

An equivalent ([Nip93, p. 309]), inference-rule based formulation of the rewrite step relation is:

$$\frac{l \to r \in \mathcal{H}}{l^\theta{\downarrow_\beta} \to_{\mathcal{H}} r^\theta{\downarrow_\beta}}\,(\mathsf{sub}) \qquad \frac{s \to_{\mathcal{H}} t}{\xi.s \to_{\mathcal{H}} \xi.t}\,(\mathsf{abs})$$

$$\frac{s_n \to_{\mathcal{H}} s'_n}{\mathsf{F}(s_1)\dots(s_n)\dots(s_m) \to_{\mathcal{H}} \mathsf{F}(s_1)\dots(s'_n)\dots(s_m)}\,(\mathsf{app})$$

3.5.1. From HRS to HORS

There is really nothing to it, to associate a pattern HORS to a HRS, since the former were defined as closures of the latter.

DEFINITION 3.5.3. Let $\mathcal{H}$ be a HRS having alphabet $\mathcal{A}$ and set of rules $\mathcal{R}$. The *closure* of $\mathcal{H}$ is the pattern HORS $\mathcal{I}(\mathcal{H}) =^{\text{def}} \langle \mathcal{A}, \lambda_{\overline{\eta}}^{\rightarrow}, \overline{\mathcal{R}} \rangle$.

PROPOSITION 3.5.4. *Let $\mathcal{H}$ be a HRS, let s be a term in $\mathcal{H}$ and let θ be a substitution. Then $\mathcal{I}(s^\theta) \equiv \overline{s}(x_1{}^\theta)\ldots(x_m{}^\theta)\!\downarrow_{\beta\overline{\eta}}$, where $\overline{s}$ is the $x_1, \ldots, x_m$-closure of s, and $\mathsf{Fvar}(s) \subseteq \{x_1, \ldots, x_m\}$.*

By this proposition, it is immediate that rewriting in a HRS corresponds to rewriting in its associated pattern HORS.

LEMMA 3.5.5. *Let $\mathcal{H}$ be a HRS, then $s \rightarrow_{\mathcal{H}} t \iff s \rightarrow_{\mathcal{I}(\mathcal{H})} t$.*

The lemma shows that there is no real difference between a HRS and its associated pattern HORS. Nevertheless, we prefer pattern HORSs over HRS. First, because they fit nicely in the general theory of HORSs. Second, because pattern HORSs are cleaner in some sense. In a HRS, the rewrite step relation is defined via closure under substitutions and contexts of the rewrite rules. In a HORS, these closures are implicitly executed by the substitution calculus and only literal replacement is performed, thereby completely separating 'replacement' as specified by the rules of the HORS from 'searching for redexes' as specified by the substitution calculus.

From our point of view, closure under contexts and substitution should capture all possible conversions which could lead to a replacement step. That is, if it is known that for some rule $\aleph =^{\text{def}} l \rightarrow r$, the closure $\overline{l}$ of its left-hand side can be extracted from s into context $D\Box$, i.e. $s \leftrightarrow^{*}_{\beta\overline{\eta}} D\boxed{\overline{l}}$, then there should exist a context $C[\,]$ and a substitution θ, such that $s \equiv C[l^\theta]$. This is true for HRSs only if all the rules are of base type, as shown by the following example taken from [Nip93, p. 309]. Consider some left-hand side $l =^{\text{def}} \xi.\mathtt{G}(\xi)(x(\xi))$ which is not of base type, and a term $s =^{\text{def}} \mathtt{G}(\mathtt{A})(\mathtt{H}(\mathtt{A}))$. Then, there does not exist $C[\,]$ and θ such that

$$s \equiv C[l^\theta]$$

However, the x-closure, $\overline{l} =^{\text{def}} \zeta.\xi.\mathtt{G}(\xi)(\zeta(\xi))$, of l can be extracted from s as witnessed by

$$s \twoheadleftarrow_{\beta\overline{\eta}} \boxed{\overline{l}}(\varsigma.\mathtt{H}(\varsigma))(\mathtt{A}).$$

This does not show that the definition of HRSs is wrong. It merely shows that care has to be taken in defining the rewrite relation via contexts and substitutions.

3.5.2. Orthogonal HRSs

A HRS is defined to be orthogonal if there do not exist critical pairs between its rules.

DEFINITION 3.5.6. ([Nip93, Def. 3.6,3.7])

1. First we define the *closed subpreterm* $\phi\backslash\backslash s$ at some position ϕ in a raw preterm s inductively like $\backslash$ (see Definition 3.1.4), but now

$$0\phi\backslash\backslash\xi.s =^{\text{def}} \xi.\phi\backslash\backslash s.$$

The binder is kept, instead of thrown away as in the definition of $\backslash$. The closed subpreterm at ϕ in s is of the form $\xi_1.\ldots\xi_m.s'$, where ξ_1, ..., ξ_m is the list of variables bound 'above' ϕ in s. For example, $0101\backslash\backslash\xi.\mathtt{F}(\zeta.\mathtt{G}(\zeta)) \equiv \xi.\zeta.\zeta$.

2. Two rules $\aleph =^{\text{def}} l \to r$ and $\daleth =^{\text{def}} g \to d$, a position $\phi \in \mathsf{Pos}(l)$, and a most general unifier θ of $\phi\backslash\backslash l$ and $\xi_1.\ldots\xi_m.g^{\sigma}$ such that

   (a) the head of $\phi\backslash\backslash l$ is not free in l,

   (b) $\xi_1,\ldots,\xi_m$ is the list of variables in the binder of $\phi\backslash\backslash l$, and σ maps each free variable x in g to a term $x'(\xi_1)\ldots(\xi_m)$, where x' is fresh,

   determine a *critical pair* $(r^{\theta}{\downarrow_{\beta}},(l[d^{\sigma}]_{\phi})^{\theta}{\downarrow_{\beta}})$. The critical pair is *trivial*, if its components are identical.

3. A HRS is *orthogonal* if there are no critical pairs except for the ones between a rule and itself at position ε. It is *weakly* orthogonal if all critical pairs are trivial.

In order to prove confluence for orthogonal HRSs it suffices to show that the absence of critical pairs in a HRS as defined above, implies the absence of critical pairs in its associated pattern HORS, as defined in Definition 3.2.32.

PROPOSITION 3.5.7. *Let $\mathcal{H}$ be a HRS. The function mapping each critical pair in $\mathcal{H}$ to its closure is a bijective correspondence between the critical pairs of $\mathcal{H}$ and $\mathcal{I}(\mathcal{H})$.*

PROOF Let $\aleph =^{\text{def}} l \to r$, $\daleth =^{\text{def}} g \to d$ be rules of the HRS. Let y_1, ..., y_n be the list of free variables in l and x_1, ..., x_m the list of free variables in g.

1. Suppose $\aleph$, $\daleth$, ϕ and θ determine the critical pair $(r^{\theta}{\downarrow_{\beta}},(l[d^{\sigma}]_{\phi})^{\theta}{\downarrow_{\beta}})$ in $\mathcal{H}$. By Proposition 3.5.4, $l^{\theta}{\downarrow_{\beta}} \twoheadleftarrow_{\beta\bar{\eta}} \boxed{\overline{l}}(y_1{}^{\theta})\ldots(y_n{}^{\theta})$ and $(l[g^{\sigma}]_{\phi})^{\theta}{\downarrow_{\beta}} \twoheadleftarrow_{\beta\bar{\eta}}$ $\overline{l[\boxed{\overline{g}}(x_1{}^{\sigma})\ldots(x_m{}^{\sigma})]}(y_1{}^{\theta})\ldots(y_n{}^{\theta})$. By abstracting over both left-hand sides and closing with respect to the free variables in both preterms, a unification pair $(C\Box,D\Box)$ is obtained. This unification pair gives rise to the critical pair $(C\boxed{r}{\downarrow_{\beta\bar{\eta}}},D\boxed{d}{\downarrow_{\beta\bar{\eta}}})$ at $\phi\,;\psi$, where ψ is the position of the head in g. in $\mathcal{I}(\mathcal{H})$.

2. Suppose there exists a unification pair for the rules $\overline{\aleph}$ and $\overline{\beth}$ at position ϕ and suppose that it is the closure of the pair $(\Box(t_1)\ldots(t_n), D\Box)$. From this information the substitution θ for the critical pair between $\aleph$ and $\beth$ has to be constructed. We consider both components of the pair in turn.

   (a) The definition of θ on the free variables in l is easy to construct. Define ${y_i}^\theta =^{\mathrm{def}} t_i$. By Proposition 3.5.4, $l^\theta{\downarrow_{\beta\bar{\eta}}} \equiv \boxed{l}(t_1)\ldots(t_n){\downarrow_{\beta\bar{\eta}}}$.

   (b) The definition of θ on the free variables in g is a lot harder to reconstruct. First, the substitution σ has to be extracted from $F\Box$. Suppose the hole occurs in $D\Box$ as $\Box(s'_1)\ldots(s'_m)$, then $s'_i \leftrightarrow^*_{\beta\bar{\eta}} s_i(\xi_1)\ldots(\xi_k)$, for each s'_i, such that s_i does not contain bound variables among $\xi_1, \ldots, \xi_k$, the list of variables bound 'above' the hole in $D\Box$. Choose fresh variables $x'_1, \ldots, x'_m$ and define ${x_i}^\sigma =^{\mathrm{def}} x'_i(\xi_1)\ldots(\xi_k)$. Finally, define ${x'_i}^\theta =^{\mathrm{def}} s_i$. It is routine to check that the rules $\aleph$, $\beth$ and the substitution θ determine a critical pair and to compute the position.

$\odot$

This correspondence between critical pairs of a HRS $\mathcal{H}$ and its translation $\mathcal{I}(\mathcal{H})$, enables us to prove the following result by Nipkow [Nip93, Cor. 4.9].

THEOREM 3.5.8. *Every orthogonal HRS is confluent.*

PROOF By Lemma 3.5.5, a HRS $\mathcal{H}$ is confluent if and only if the pattern HORS $\mathcal{I}(\mathcal{H})$ is confluent. By Proposition 3.5.7, $\mathcal{H}$ is orthogonal if and only if $\mathcal{I}(\mathcal{H})$ is orthogonal. $\odot$

In proving this theorem via FD, we have carried out the task of lifting a large body of λ-calculus reduction theory to HRSs as was suggested by Nipkow ([Nip93, p. 316]). This could have been done more easily as suggested there, by adapting the methods employed by Klop in [Klo80]. We think the more abstract route via HORSs taken here is justified, because it sheds more light on the reasons why the constructions work.

In CRSs as well as in HRSs, restricting left-hand sides to patterns is 'built into the system'. For the study of properties of HORSs other than orthogonality, these restrictions are often superfluous and even for studying questions of orthogonality the restrictions to patterns can be lifted sometimes.

EXAMPLE 3.5.9. Consider the rule $\aleph$:

$$\mathtt{F}(\xi.x(\mathtt{cons}(\mathtt{0})(\xi))) \quad\to\quad \mathtt{G}(\xi.x(\xi))$$

taken from [KOR93]. This rule strips away the head "0" of a "cons" in an instantiation of x, if *all* occurrences of the bound variable ξ are of the form $\mathtt{cons}(\mathtt{0})(\xi)$. If this is not the case, then the rule is not applicable. This rule does not fit in the scope of CRSs or HRSs, but as one easily checks, no problems whatsoever appear in the treatment of this rule as an orthogonal one. The rule which can strip away the head "0" of any number of "cons"s can be written as:

$$\mathtt{F}(\xi.x(\xi)(\mathtt{cons}(\mathtt{0})(\xi))) \quad \to \quad \mathtt{G}(\xi.x(\xi)(\xi))$$

This rule is not orthogonal any more.

As the example makes clear, the formalism employed thus far is not very expressive concerning the term which may be substituted for the variable x. For example, one could imagine wanting to express that precisely one head "0" of a "cons" can be stripped away by the rule $\aleph$ in the example above. This could be done by 'annotating' the variable x, or stated differently, by giving the variable x some type expressing its linearity.

3.5.3. Weakly Orthogonal HRSs

The correspondence between HRSs and pattern HORSs allows one to rephrase Theorem 2.4.12 as:

THEOREM 3.5.10. *Every weakly orthogonal HRS is confluent.*

The most famous example of a weakly orthogonal higher-order rewriting system is lambda beta-eta calculus. Its presentation as a HRS/left-linear pattern HORS is as follows. The alphabet of lambda beta-eta contains two operator symbols, one for abstraction and one for application:

$$\mathsf{abs}\colon (o \to o) \to o$$

$$\mathsf{app}\colon o \to (o \to o)$$

The rewrite rules of lambda beta-eta are:

$$\begin{aligned} \mathsf{app}(\mathsf{abs}(\xi.x(\xi)))(y) \quad &\to_{\mathsf{beta}} \quad x(y) \\ \mathsf{abs}(\xi.\mathsf{app}(x)(\xi)) \quad &\to_{\mathsf{eta}} \quad x \end{aligned}$$

There are two unification pairs for these rules, which are the closures of the pairs:

$$(\Box(\xi.\mathsf{app}(x)(\xi))(y), \mathsf{app}(\Box(x))(y))$$

$$(\Box(\mathsf{abs}(\xi.x(\xi))), \mathsf{abs}(\xi.\Box(x)(\xi)))$$

giving rise to two trivial critical pairs, the components of which are equal to (the closures of):

$$\mathsf{app}(\mathsf{abs}(\xi.\mathsf{app}(x)(\xi)))(y)$$

$$\mathsf{abs}(\xi.x(\xi))$$

By the theorem we immediately obtain confluence.

Remark 3.5.11. This proof of confluence may look like cheating, because one could argue that confluence of λ-calculus (the substitution calculus) is used to prove its own own confluence (the object calculus). Although the machinery employed is certainly overkill, we do not employ circular reasoning since we are dealing with different calculi. The substitution calculus is $\lambda^{\rightarrow}_{\overline{\eta}}$ while the object calculus is lambda beta-eta calculus. The fact that the former is terminating makes its proof of confluence essentially easier.

Remark 3.5.12. Another proof of confluence of lambda beta-eta can be obtained by the method of decreasing diagrams of Section 2.3, noting that the eta-rule does not have splitting effect on beta, that is, we have by simple case analysis the diagram:

$$
\begin{tikzcd}
\cdot \arrow[r, "\mathsf{beta}"] \arrow[d, "\mathsf{eta}"'] \arrow[dr, phantom, "="] & \cdot \arrow[d, dashed, "\mathsf{eta}"] \\
\cdot \arrow[r, dashed, "\mathsf{beta}"'] & \cdot
\end{tikzcd}
$$

Taking $\mathsf{beta} \succ \mathsf{eta}$, the diagram is directly seen to be decreasing, so beta commutes with eta. Since both rules are orthogonal themselves, both beta and eta are confluent. Now we can apply the Lemma of Hindley-Rosen to obtain confluence of lambda beta-eta.

The question is of course, does this system correspond exactly to the usual lambda beta-eta calculus? The answer is *no*, there are a lot of junk terms. For example, $\xi.\xi$ does not correspond to any ordinary lambda term. The junk terms can easily be removed from the HORS by inductively defining the set of 'real' lambda terms. First a new type 'Λ' is introduced, which denotes the set of all lambda terms. This is a restriction of the type 'o' denoting the set of all terms. Term formation rules are:

$$
\frac{}{x\colon\Lambda}\,(\mathsf{var}) \qquad \frac{x\colon\Lambda \quad s\colon\Lambda}{\mathtt{abs}(\xi.s')\colon\Lambda}\,(\mathsf{abs}) \qquad \frac{s\colon\Lambda \quad t\colon\Lambda}{\mathtt{app}(s)(t)\colon\Lambda}\,(\mathsf{app})
$$

where $\xi.s'$ is an x-closure of s. Now we find ourselves in the peculiar situation that a subject-reduction theorem for an untyped lambda calculus has to be proven, since we want this set of 'real' terms to be closed under HORS-steps. This is routine.

The question what a model of the (untyped) lambda calculus is (see e.g. [HL80, Mey82]), has not been answered properly in my opinion. The answer I suggest is that a model of the lambda calculus is a model of the HORS encoding it. This raises the question what a model of a HORS is. Due to several kinds of limitations I've not included a chapter on this subject, but it might be helpful to present the idea for the case of lambda beta-eta.

First, a model of the untyped lambda calculus should give interpretations of the type Λ in some set A, and interpretations of the operators `app` and `abs` satisfying the above formation rules. Preterms are interpreted as

$$\begin{array}{lll} [\![s(t)]\!] & =^{\text{def}} & [\![s]\!]([\![t]\!]) \\ [\![\xi.s']\!] & =^{\text{def}} & \boldsymbol{\lambda}[\![\xi]\!].[\![s]\!]_{x\mapsto[\![\xi]\!]} \end{array}$$

where $\xi.s'$ is the x-closure of s. Second, a model should model the beta-rule and eta-rule as usual. Interpreting the beta- and eta-rules in this way, one gets the identities:

$$[\![\mathsf{app}]\!] \circ [\![\mathsf{abs}]\!] = \mathsf{id}\colon \mathbf{Rep}(A) \rightarrow_{[\![\mathsf{abs}]\!]} A \rightarrow_{[\![\mathsf{app}]\!]} \mathbf{Rep}(A)$$

$$[\![\mathsf{abs}]\!] \circ [\![\mathsf{app}]\!] = \mathsf{id}\colon A \rightarrow_{[\![\mathsf{app}]\!]} \mathbf{Rep}(A) \rightarrow_{[\![\mathsf{abs}]\!]} A$$

for some subset $\mathbf{Rep}(A)$ of the set of all functions from A to itself.

Because we only wanted to sketch what a model should be, we do not answer the question how to actually construct models of the lambda beta-eta calculus. For information on this, the reader should definitely consult [Plo93].

3.5.4. Combinations of HRSs

We can reformulate Theorem 3.1.54 in the case of HRSs as:

THEOREM 3.5.13. *Every weakly orthogonal combination of confluent HRSs is confluent.*

This generalises a result of Nipkow ([Nip93, Thm. 6.1]), where one of the systems is required to have non-duplicating rules. This seems to be the most general confluence result about combinations of rewriting systems at present, but extensions are certainly desirable.

First note that the result by Toyama, stating that combinations of confluent TRSs are confluent again, does not hold for CRSs or HRSs. Both the beta-rule and the non-left-linear rule

$$\mathtt{F}(x)(x) \rightarrow \mathtt{G}$$

are confluent, but their disjoint union is not confluent [Klo80].

The most general, 'critical pair'-based confluence result for TRSs is a result by Toyama ([Toy88, Thm. 3.1]), generalising the result by Huet ([Hue80]), that all left-linear 'parallel closed' term rewriting systems are confluent. It would be interesting to see whether these results can be lifted to the higher-order case. This is not a priori clear, since the proofs employed rely on the fact that TRS rules do not 'nest' subterms.

3.5.5. From HRSs to CRSs

Although it has always been clear that Klop's CRSs and Nipkow's HRSs were equally expressive in some sense, the exact relationship between these systems was not obvious. In [OR93] these two formalism were compared. As it turns out, both formats are roughly co-extensive, and have the same expressive power. Due to the restriction to patterns of HRSs, the formats have the same 'matching' power, but HRSs have more 'substitution' power. The difference lies in the substitution calculus employed. For HRSs this is $\lambda_{\overline{\eta}}^{\rightarrow}$-calculus, while for CRSs it is the language of 'developments'. As we have shown, developments can be encoded into $\lambda_{\overline{\eta}}^{\rightarrow}$, but not the other way around. Nevertheless, an encoding of HRSs into CRSs can be given by adding an 'explicit' substitution rule to the latter. In [OR93] it was shown that the encoding allows to transfer the 'confluence by orthogonality' result from CRSs to HRSs.

3.5.6. A list-extension of lambda calculus

In this paragraph we combine the method of confluence by decreasing diagrams with the method of confluence by orthogonality to obtain confluence of a list-oriented extension of lambda calculus. This system has a non-trivial critical pair, so it is not automatically confluent.

EXAMPLE 3.5.14. Consider a list-oriented extension of lambda-calculus $\mathcal{H}$, having the closures of the following rules as rewrite rules.

$$
\begin{array}{rcl}
\mathsf{app}(\mathsf{abs}(\xi.x(\xi)))(y) & \rightarrow_{\mathsf{beta}} & x(y) \\
\mathsf{app}([x_1,\ldots,x_m])(y) & \rightarrow_{\mathsf{gamma1}} & [\mathsf{app}(x_1)(y),\ldots,\mathsf{app}(x_m)(y)] \\
\mathsf{abs}(\varsigma.[x_1(\varsigma),\ldots,x_m(\varsigma)]) & \rightarrow_{\mathsf{gamma2}} & [\mathsf{abs}(\varsigma.x_1(\varsigma)),\ldots,\mathsf{abs}(\varsigma.x_m(\varsigma))]
\end{array}
$$

These are the rules of Révész [Rév92] in our notation. The main purpose of that paper is to prove the consistency of this extended lambda-calculus by showing that it satisfies the Church-Rosser theorem. We think the following proof is both simpler and more intuitive than the proof given there.

CLAIM $\mathcal{H}$ is confluent.

PROOF The pattern HORS $\mathcal{H}$ is 'almost' orthogonal. It is left-linear and the only ambiguity is between rules beta and $\mathsf{gamma2}$. In order to prove confluence it suffices to prove that $\mathsf{beta}+\mathsf{gamma1}$ commutes with $\mathsf{gamma2}$ by the Lemma of Hindley-Rosen (Example 2.3.9(1)), because both $\mathsf{beta}+\mathsf{gamma1}$ and $\mathsf{gamma2}$ are orthogonal, hence confluent by Theorem 3.2.26. There are three possible local divergences between these rules, which are easily completed to confluence diagrams. The first two are orthogonal divergences which can be completed by the usual 'parallel moves', the last one is the overlap-case.

$$
\begin{tikzcd}
\cdot \arrow[r, "\mathsf{gamma1}"] \arrow[d, "\mathsf{gamma2}"'] & \cdot \arrow[d, two heads, "\mathsf{gamma2}"'] \\
\cdot \arrow[r, "\mathsf{gamma1}"] & \cdot
\end{tikzcd}
\qquad
\begin{tikzcd}
\cdot \arrow[r, "\mathsf{beta}"] \arrow[d, "\mathsf{gamma2}"'] & \cdot \arrow[d, two heads, "\mathsf{gamma2}"'] \\
\cdot \arrow[r, "\mathsf{beta}"] & \cdot
\end{tikzcd}
$$

$$\begin{tikzcd}
\mathsf{app}(\mathsf{abs}(\xi.[s_1(\xi),\ldots,s_m(\xi)]))(t) \arrow[r, "\mathsf{gamma2}"] \arrow[dd, "\mathsf{beta}"'] & \mathsf{app}([\mathsf{abs}(\xi.s_1(\xi)),\ldots,\mathsf{abs}(\xi.s_m(\xi))])(t) \arrow[d, "\mathsf{gamma1}"'] \\
& {[\mathsf{app}(\mathsf{abs}(\xi.s_1(\xi)))(t),\ldots,\mathsf{app}(\mathsf{abs}(\xi.s_m(\xi)))(t)]} \arrow[d, "\mathsf{beta}"'] \\
{[s_1(t),\ldots,s_m(t)]} \arrow[r, phantom, "\equiv"] & {[s_1(t),\ldots,s_m(t)]}
\end{tikzcd}$$

By Corollary 2.3.8 it suffices to find an LRS $\mathcal{I}$ which is step-equivalent to $\mathcal{H}$ such that all the local confluence diagrams are decreasing. We can reason as follows:

1. In searching for an ordering of the labels, we notice that in the first two diagrams both gamma1 and beta have splitting effect on gamma2. So we should order gamma2 below gamma1 and beta.

2. Looking at the overlap-diagram a difficulty is spotted in that gamma2 has splitting effect on beta.

3. However, this can easily be overcome by noticing that each of the bodies of the beta-redexes in the right side of the diagram, contains less brackets than the body of the beta-redex in the left side.

4. Hence, we can refine beta into beta_n's where n is the number of brackets in the body of the contracted beta-redex.

Formally, if $s \rightarrow_{\mathsf{beta}} t$ by contracting the redex $\mathsf{app}(\mathsf{abs}(\xi.s(\xi)))t$, and n is the number of brackets $([,])$ in s, then $s \rightarrow_{\mathsf{beta}_n} t$ in the LRS. Define the order $\succ$ on $\{\mathsf{gamma1}, \mathsf{gamma2}, \mathsf{beta}_n.n \in \mathbb{N}\}$ by $\forall n \in \mathbb{N}.\mathsf{beta}_{n+1} \succ \mathsf{beta}_n \succ \mathsf{gamma1} \succ \mathsf{gamma2}$. The refined confluence diagrams corresponding to the ones above are decreasing. (Note that the number of brackets in the body of a beta-redex is invariant with respect to gamma2-rewriting in the non-overlap case.) $\odot$

Bibliography

[AB78] G. Ausiello and C. Böhm, editors. *Automata, Languages and Programming, Fifth Colloquium, Udine, Italy, July 17–21, 1978*, volume 62 of *Lecture Notes in Computer Science*. Springer-Verlag, 1978.

[Acz78] Peter Aczel. A general Church-Rosser theorem. Technical report, University of Manchester, July 1978.

[AGM92a] S. Abramsky, Dov M. Gabbay, and T. S. E. Maibaum, editors. *Handbook of Logic in Computer Science*, volume 2, Background: Computational Structures. Oxford University Press, New York, 1992.

[AGM92b] S. Abramsky, Dov M. Gabbay, and T. S. E. Maibaum, editors. *Handbook of Logic in Computer Science*, volume 1, Background: Mathematical Structures. Oxford University Press, New York, 1992.

[Aka93] Yohji Akama. On Mints' reduction for ccc-calculus. in [BG93, pp. 1–12], 1993.

[AL92] A. Asperti and C. Laneve. Interaction systems I, the theory of optimal reductions. Rapports de Recherche 1748, INRIA-Rocquencourt, September 1992.

[Bar84] H. P. Barendregt. *The Lambda Calculus, Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. Elsevier Science Publishers B.V., Amsterdam, revised edition, 1984.

[Bar92] H. P. Barendregt. Lambda calculi with types. in [AGM92a, pp. 117–310], 1992.

[BCL85] G. Berry, P.-L. Curien, and J.-J. Lévy. Full abstraction for sequential languages: the state of the art. in [NR85, Ch. 3], 1985.

[Ber78] Gérard Berry. Stable models of typed λ-calculi. in [AB78, pp. 72–89], 1978.

[BF] Franco Barbanera and Maribel Fernández. Combining first and higher order rewrite systems with type assignment systems. in [BG93, pp. 60–74].

[BG93] M. Bezem and J. F. Groote, editors. *Proceedings of the International Conference on Typed Lambda Calculi and Applications, TLCA'93, March 1993, Utrecht, The Netherlands*, volume 664 of *Lecture Notes in Computer Science*, Berlin Heidelberg, 1993. Springer-Verlag.

[Bli93] Wayne D. Blizard. Dedekind multisets and function shells. *Theoretical Computer Science*, 110(1):79–98, March 1993.

[BO93] Ronald V. Book and Friedrich Otto. *String-Rewriting Systems.* Texts and Monographs in Computer Science. Springer-Verlag, 1993.

[Boo91] Ronald V. Book, editor. *Rewriting Techniques and Applications, 4th International Conference, RTA-91, Como, Italy, April 10–12, 1991*, volume 488 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Heidelberg, 1991.

[Bru78] N. G. de Bruijn. A note on weak diamond properties. Memorandum 78-08, Eindhoven University of Technology, August 1978.

[CF58] Haskell B. Curry and Robert Feys. *Combinatory Logic, volume I.* Studies in Logic and the Foundations of Mathematics. North-Holland, 1958.

[CG91] Pierre-Louis Curien and Giorgio Ghelli. On confluence for weakly normalizing systems. in [Boo91, pp. 215–225], 1991.

[CHL92] Pierre-Louis Curien, Thérèse Hardin, and Jean-Jacques Lévy. Confluence properties of weak and strong calculi of explicit substitutions. Rapports de Recherche 1617, INRIA-Rocquencourt, February 1992.

[Chu32] Alonzo Church. A set of postulates for the foundation of logic. *Annals of Mathematics*, 33:346–366, 1932.

[Chu33] Alonzo Church. A set of postulates for the foundation of logic. (second paper.). *Annals of Mathematics*, 34:839–864, 1933.

[Chu41] A. Church. *The Calculi of Lambda Conversion.* Princeton University Press, 1941.

[CR36] Alonzo Church and J. B. Rosser. Some properties of conversion. *Transactions of the American Mathematical Society*, 39:472–482, January to June 1936.

[CR93] A. Corradini and F. Rossi. A new term graph rewriting formalism: Hyperedge replacement jungle rewriting. in [SPE93, Ch. 8], 1993.

[Cro75] J. N. Crossley, editor. *Algebra and Logic, Papers from the 1974 Summer Research Institute of the Australian Mathematical Society, Monash Univerisity, Australia*, volume 450 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin Heidelberg, 1975.

[Cur30a] H. B. Curry. Grundlagen der kombinatorischen Logik. Teil I. *American Journal of Mathematics*, LII:509–536, 1930.

[Cur30b] H. B. Curry. Grundlagen der kombinatorischen Logik. Teil II. *American Journal of Mathematics*, LII:789–834, 1930.

[DD93] Arie van Deursen and T. B. Dinesh. Origin tracking for higher-order term rewriting systems. in [HOA93], 1993.

[DJ90] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. in [Lee90, Ch. 6 pp. 243–320], 1990.

[DJ91] Nachum Dershowitz and Jean-Pierre Jouannaud. Notations for rewriting. *Bulletin of the European Association for Theoretical Computer Science*, 43:162–172, February 1991.

[DJK93] Nachum Dershowitz, Jean-Pierre Jouannaud, and Jan Willem Klop. More problems in rewriting. in [Kir93, pp. 468–487], 1993.

[DKP91] Nachum Dershowitz, Stéphane Kaplan, and David A. Plaisted. Rewrite, rewrite, rewrite, rewrite, rewrite, ... *Theoretical Computer Science*, 83(1):71–96, 1991.

[DKS86] J. Demetrovics, G. Katona, and A. Salomaa, editors. *Algebra, Combinatorics and Logic in Computer Science, Györ (Hungary, 1983, Vol. II*, Colloquia Mathematica Societatis János Bolyai, 42. North-Holland, 1986.

[DKT93] A. van Deursen, P. Klint, and F. Tip. Origin tracking. *Journal of Symbolic Computation*, 15:523–545, 1993.

[DM79] Nachum Dershowitz and Zohar Manna. Proving termination with multiset orderings. *Communications of the ACM*, 22(8):465–476, August 1979.

[DP90] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 1990.

[EHSH92] L.-H. Eriksson, L. Hallnäs, and P. Schroeder-Heister, editors. *Extensions of Logic Programming, Second International Workshop, ELP '91, Stockholm, Sweden, January 27–29, 1991*, volume 596 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin Heidelberg, 1992.

[Fel91] Amy Felty. A logic programming approach to implementing higher-order term rewriting. in [EHSH92, pp. 135–161], 1991.

[Fre62] G. Frege. *Grundgesetze der Arithmetik. Begriffschriftlich abgeleitet*, volume I. Georg Olms Verlagsbuchhandlung, Hildesheim, 1962. Unveränderte reprografischer Nachdruck der Auflage Jena 1893.

[FW91] William M. Farmer and Ronald J. Watro. Redex capturing in term graph rewriting. in [Boo91, pp. 13–24], 1991. (Concise Version).

[Ges90] Alfons Geser. *Relative Termination.* PhD thesis, University of Passau, 1990.

[GLM92] Georges Gonthier, Jean-Jacques Lévy, and Paul-André Melliès. An abstract standardisation theorem. in [LIC92, pp. 72–81], 1992.

[Har93] T. Hardin. How to get confluence for explicit substitutions. in [SPE93, Ch. 3], 1993.

[Hei67] J. van Heijenoort, editor. *From Frege to Gödel A Source Book in Mathematical Logic, 1879–1931.* Source Books in the History of the Sciences. Harvard University Press, Cambridge, Massachusetts, 1967.

[Hin64] J. R. Hindley. *The Church-Rosser Property and a Result in Combinatory Logic.* PhD thesis, University of Newcastle-upon-Tyne, 1964.

[Hin78a] R. Hindley. Reductions of residuals are finite. *Transactions of the American Mathematical Society*, 240:345–361, June 1978.

[Hin78b] R. Hindley. Standard and normal reductions. *Transactions of the American Mathematical Society*, 241:253–271, July 1978.

[HL80] R. Hindley and G. Longo. Lambda-calculus models and extensionality. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 26:289–310, 1980.

[HL91a] Gérard Huet and Jean-Jacques Lévy. Computations in orthogonal rewriting systems, I. in [LP91, Ch. 11], 1991.

[HL91b] Gérard Huet and Jean-Jacques Lévy. Computations in orthogonal rewriting systems, II. in [LP91, Ch. 12], 1991.

[HOA93] *HOA '93, An International Workshop on Higher Order Algebra, Logic and Term Rewriting, 23rd–24th September 1993, CWI, Amsterdam, The Netherlands*, 1993. Participant's Proceedings.

[Hof92] Berthold Hoffman. Term rewriting with sharing and memoïzation. in [KL92, pp. 128–142], 1992.

[HS86] J. Roger Hindley and Jonathan P. Seldin. *Introduction to Combinators and λ-Calculus*, volume 1 of *London Mathematical Society Students Texts.* Cambridge University Press, 1986.

[Hue80] Gérard Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the Association for Computing Machinery*, 27(4):797–821, October 1980.

[Jan88] Matthias Jantzen. *Confluent String Rewriting*, volume 14 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Berlin Heidelberg, 1988.

[JK86] Jean-Pierre Jouannaud and Hélène Kirchner. Completion of a set of rules modulo a set of equations. *SIAM Journal of Computing*, 15(4):1155–1194, November 1986.

[JL82] Jean-Pierre Jouannaud and Pierre Lescanne. On multiset orderings. *Information Processing Letters*, 15(2):57–63, September 1982.

[Kah] Stefan Kahrs. Context rewriting. in [RR93, pp. 21–35].

[Kah93] Stefan Kahrs. Compilation of combinatory reduction systems. in [HOA93], 1993.

[Ken91] Richard Kennaway. Transfinite abstract reduction systems. Talk presented at TeReSe-meeting, July 1993, Amsterdam, 1991.

[Ken92] J. R. Kennaway. On transfinite abstract reduction systems. Report CS-R9205, CWI, January 1992.

[Kha90] Z. O. Khasidashvili. Expression reduction systems. In *Proceedings of I. Vekua Institute of Applied Mathematics*, volume 36, pages 200–220, Tbilisi, 1990.

[Kha92] Zurab Khasidashvili. The Church-Rosser theorem in orthogonal combinatory reduction systems. Rapports de Recherche 1825, INRIA-Rocquencourt, December 1992.

[Kha93] Z. Khasidashvili. Perpetual reductions in orthogonal combinatory reduction systems. Report CS-R9349, CWI, July 1993.

[Kir93] Claude Kirchner, editor. *Rewriting Techniques and Applications, 5th International Conference, RTA-93, Montreal, Canada, June 16–18, 1993*, volume 690 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Heidelberg, 1993.

[KKSV93] Richard Kennaway, Jan Willem Klop, Ronan Sleep, and Fer-Jan de Vries. Transfinite reductions in orthogonal term rewriting systems. Technical Report SIS-C93-10, School of Information Systems, University of East Anglia, Norwich, U.K., October 1993. to appear in Information and Computation.

[KL92] H. Kirchner and G. Levi, editors. *Algebraic and Logic Programming, Third International Conference, Volterra, Italy, September 2–4, 1992*, volume 632 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992.

[Klo80] J. W. Klop. *Combinatory Reduction Systems.* PhD thesis, Rijksuniversiteit Utrecht, June 1980. Mathematical Centre Tracts 127.

[Klo90] J. W. Klop. Term rewriting systems. Report CS-R9073, Centre for Mathematics and Computer Science, December 1990.

[Klo92] J. W. Klop. Term rewriting systems. in [AGM92a, pp. 1–116], 1992.

[KMTV91] J. W. Klop, A. Middeldorp, Y. Toyama, and R. de Vrijer. A simplified proof of Toyama's theorem. IR 270, Vrije Universiteit, Amsterdam, December 1991. to appear in Information Processing Letters.

[KOR93] J. W. Klop, V. van Oostrom, and F. van Raamsdonk. Combinatory reduction systems, introduction and survey. *Theoretical Computer Science*, 121(1–2):279–308, December 1993.

[Kri90] J. L. Krivine. *Lambda-calcul, types et modèles.* Études et recherches en informatique. Masson, Paris, 1990.

[Laf90] Yves Lafont. Interaction nets. In *Proceedings* 17^{th} *ACM Symposium on Principles of Programming Languages*, pages 95–108, 1990.

[Lan93] Cosimo Laneve. *Optimality and Concurrency in Interaction Systems.* PhD thesis, Dipartimento di Informatica, Università degli Studi di Pisa, March 1993.

[Lee90] Jan van Leeuwen, editor. *Handbook of Theoretical Computer Science*, volume B : Formal Models and Semantics. Elsevier Science Publishers B.V., Amsterdam, 1990.

[Lév80] Jean-Jacques Lévy. Optimal reductions in the lambda-calculus. in [SH80, pp. 159–191], 1980.

[LFP82] *Proceedings of the ACM Symposium on LISP and Functional Programming*, 1982.

[LIC91] *Proceedings of the Sixth Annual IEEE Symposium on Logic in Computer Science, Amsterdam, The Netherlands, July 15–18, 1991*, Los Alamitos, California, 1991. IEEE Computer Society Press.

[LIC92] *Proceedings of the Seventh Annual IEEE Symposium on Logic in Computer Science, Santa Cruz, California, June 22–25, 1992*, Los Alamitos, California, 1992. IEEE Computer Society Press.

[Lip93] Ernst Lippe. Generating tables for bottom-up matching. in [Kir93, pp. 274–288], 1993.

[LP91] Jean-Louis Lassez and Gordon Plotkin, editors. *Computational Logic: Essays in Honor of Alan Robinson.* The MIT Press, Cambridge, Massachusetts, 1991.

[Man93] Ken Mano, September 1993. Personal communication.

[Mar87] Ursula Martin. Extension functions for multiset orderings. *Information Processing Letters*, 26:181–186, December 1987.

[Mar89] Ursula Martin. A geometrical approach to multiset orderings. *Theoretical Computer Science*, 67(1):37–54, 1989.

[Mar90] Ursula Martin. A note on division orderings on strings. *Information Processing Letters*, 36(5):237–240, December 1990.

[Mel93] P.-A. Melliès. An abstract theorem of finite developments. Talk presented at CONFER-meeting, september 1993, Amsterdam, 1993.

[Mey82] Albert R. Meyer. What is a model of the lambda calculus? *Information and Control*, 52(1):87–122, January 1982.

[Mil91] Dale Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. in [SH91, pp. 253–281], 1991.

[MMMS90] M. Main, A. Melton, M. Mislove, and D. Schmidt, editors. *Mathematical Foundations of Programming Sciences, 5th International Conference, Tulane University, New Orleans, Louisiana, USA, March/April 1989*, volume 442 of *Lecture Notes in Computer Science*, Berlin Heidelberg, 1990. Springer-Verlag.

[MS91] Aart Middeldorp and Mirjana Starčević. A rewrite approach to polynomial ideal theory. Report CS-R9160, Centre for Mathematics and Computer Science, December 1991.

[Mül92] Fritz Müller. Confluence of the lambda calculus with left-linear algebraic rewriting. *Information Processing Letters*, 41:293–299, April 1992.

[Ned73] R. P. Nederpelt. *Strong Normalization in a Typed Lambda Calculus with Lambda Structured Types.* PhD thesis, Technische Hogeschool Eindhoven, June 1973.

[Ned92] R. P. Nederpelt. The fine-structure of lambda calculus. Computing Science Notes 92/07, Eindhoven University of Technology, April 1992.

[New42] M. H. A. Newman. On theories with a combinatorial definition of "equivalence". *Annals of Mathematics*, 43(2):223–243, April 1942.

[Nip91] Tobias Nipkow. Higher-order critical pairs. in [LIC91, pp. 342–349], 1991.

[Nip93] Tobias Nipkow. Orthogonal higher-order rewrite systems are confluent. in [BG93, pp. 306–317], 1993.

[NR85] Maurice Nivat and John C. Reynolds, editors. *Algebraic methods in semantics*. Cambridge University Press, 1985.

[NW63] C. St. J. A. Nash-Williams. On well-quasi-ordering finite trees. *Proceedings of the Cambridge Philosophical Society*, 59:833–835, 1963.

[Oos92] V. van Oostrom. Confluence by decreasing diagrams. IR 298, Vrije Universiteit, Amsterdam, August 1992.

[Oos94] Vincent van Oostrom. Confluence by decreasing diagrams. *Theoretical Computer Science*, 1994. To appear. Available as [Oos92].

[OR93] V. van Oostrom and F. van Raamsdonk. Comparing combinatory reduction systems and higher-order rewrite systems. IR 333, Vrije Universiteit, Amsterdam, August 1993.

[OR94] V. van Oostrom and F. van Raamsdonk. Weak orthogonality implies confluence: the higher-order case. accepted for Logical Foundations of Computer Science '94, 1994.

[Plo93] Gordon D. Plotkin. Set-theoretical and other elementary models of the λ-calculus. *Theoretical Computer Science*, 121(1–2):351–409, December 1993.

[Plu93a] Detlef Plump. *Evaluation of Functional Expressions by Hypergraph Rewriting*. PhD thesis, Universität Bremen, April 1993.

[Plu93b] Detlef Plump. Hypergraph rewriting: Critical pairs and undecidability of confluence. in [SPE93, Ch. 15], 1993.

[Pol93] Jaco van de Pol. Termination proofs for higher-order rewrite systems. in [HOA93], 1993.

[Pra65] Dag Prawitz. *Natural Deduction, A Proof-Theoretical Study*, volume 3 of *Stockholm Studies in Philosophy*. Almqvist & Wiksell, Stockholm, 1965.

[Qia92] Z. Qian. Unification of higher-order patterns in linear time and space. Technical Report 5/92, Universität Bremen, October 1992.

[Raa93] Femke van Raamsdonk. Confluence and superdevelopments. in [Kir93, pp. 168–182], 1993.

[Rév92] György E. Révész. A list-oriented extension of the lambda-calculus satisfying the Church-Rosser theorem. *Theoretical Computer Science*, pages 75–89, 1992.

[Ros73] Barry K. Rosen. Tree-manipulating systems and Church-Rosser theorems. *Journal of the Association for Computing Machinery*, 20(1):160–187, January 1973.

[Ros82] J. Barkley Rosser. Highlights of the history of the lambda-calculus. in [LFP82, pp. 216–225], 1982.

[RR93] M. Rusinowitch and J. L. Rémy, editors. *Conditional Term Rewriting Systems, Third International Workshop, CTRS-92, Pont-à-Mousson, France, July 8–10, 1992*, volume 656 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[Sch24] M. Schönfinkel. Über die Bausteine der mathematischen Logik. *Mathematische Annalen*, 92:305–316, 1924. (English translation in [Hei67, pp. 355-366]).

[SH80] J. P. Seldin and J. R. Hindley, editors. *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism.* Academic Press Inc., 1980.

[SH91] P. Schroeder-Heister, editor. *Extensions of Logic Programming, International Workshop, Tübingen, FRG, December 8–10, 1989*, volume 475 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1991.

[Smy92] M. B. Smyth. Topology. in [AGM92b, pp. 641–762], 1992.

[SPE93] M. R. Sleep, M. J. Plasmeijer, and M. C. J. D. van Eekelen, editors. *Term Graph Rewriting, Theory and Practice*. John Wiley & Sons, 1993.

[Sta74] John Staples. Church-Rosser theorems for replacement systems. in [Cro75, pp. 291–307], 1974.

[Sta90] Eugene W. Stark. Connections between a concrete and an abstract model of concurrent systems. in [MMMS90, pp. 53–79], 1990.

[Tak93] Masako Takahashi. λ-calculi with conditional rules. in [BG93, pp. 406–417], 1993.

[Toy87] Yoshihito Toyama. On the Church-Rosser property for the direct sum of term rewriting systems. *Journal of the Association for Computing Machinery*, 34(1):128–143, January 1987.

[Toy88] Yoshihito Toyama. Commutativity of term rewriting systems. In F. Fuchi and L. Kott, editors, *Programming of Future Generation Computer*, volume II, pages 393–407. North-Holland, 1988.

[Vri87] R. C. de Vrijer. *Surjective Pairing and Strong Normalization: Two Themes in Lambda Calculus*. PhD thesis, Universiteit van Amsterdam, January 1987.

[WB83] F. Winkler and B. Buchberger. A criterion for eliminating unnecessary reductions in the Knuth-Bendix algorithm. in [DKS86, pp. 849–869], 1983.

[Wol93] D. A. Wolfram. *The Clausal Theory of Types*, volume 21 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1993.

[YH89] Hirofumi Yokouchi and Teruo Hikita. A rewriting system for categorical combinators with multiple arguments. March 1989.

Samenvatting

In het Nederlands vertaald, luidt de titel van het onderhavige proefschrift: "Confluentie voor Abstract en Hogere-Orde Herschrijven". Aan de hand van de titel wordt in deze samenvatting gepoogd de lezer vertrouwd te maken met het concept herschrijven en de daarbij behorende problematiek, die in dit proefschrift bestudeerd wordt.

Herschrijven Het woord herschrijven roept verscheidene associaties op. Bij herschrijven wordt onmiddellijk gedacht aan schrijven. Dit kan het schrijven van een samenvatting zijn, maar ook het schrijven van een som zoals $2 + 2$. De tweede associatie die opgeroepen wordt, is die van *her*schrijven, dat wil zeggen opnieuw schrijven. Waarom zou men iets opnieuw schrijven dat al geschreven is? Er zijn verschillende redenen waarom men dit zou willen: de oorspronkelijke tekst is bijvoorbeeld niet duidelijk, niet eenvoudig genoeg, hij is te lang of hij bevat spelfouten. De bedoeling is dan om een nieuwe tekst te genereren die beter is, maar de oorspronkelijke betekenis van de tekst behoudt. Denkende aan sommen als wiskundige teksten, kan de som $2 + 2$ herschreven worden als het getal 4. Het getal 4 heeft dezelfde betekenis als $2 + 2$, maar is eenvoudiger en duidelijker. Het verschil tussen herschrijven van tekst en het herschrijven van sommen is dat voor het laatste een verzameling van reken*regels* bestaat die voor iedere som, door herhaald toepassen van de regels een antwoord oplevert. Een antwoord is hier een som die niet meer vereenvoudigd kan worden. Het zou gemakkelijk zijn als er ook voor het herschrijven van teksten zo'n verzameling regels bestond die een willekeurige tekst naar een duidelijkste vorm transformeert. Helaas is dit niet het geval. In dit proefschrift worden alleen herschrijfsystemen bestudeerd, waarbij het herschrijven plaatsvindt aan de hand van *herschrijfregels*.

Confluentie Als men aan het herschrijven slaat door het toepassen van herschrijfregels, dan kan het licht voorkomen dat er een moment is waarop er meerdere regels zijn die toegepast kunnen worden. Bijvoorbeeld, de som $(2 + 2) \times 5$ kan herschreven worden als 4×5, maar ook als $(2 \times 5) + (2 \times 5)$. De som $(2 + 2) \times 5$ noemen we om deze reden een *splitsingspunt*. Je kunt je afvragen of in zo'n geval het uiteindelijke antwoord afhangt van de keuze voor de eerste of de tweede berekeningsmethode. In dit geval is het antwoord nee, als we verder rekenen dan zal het antwoord altijd 20 zijn. Een herschrijfsysteem waarbij het resultaat niet afhangt van de keuzes die gemaakt worden bij het herschrijven heet *confluent*.

Het is een geruststellende gedachte dat rekenen confluent is, maar is ieder herschrijfsysteem confluent? Het antwoord is nee. Beschouw het geval van 'doublethink' rekenen. Dat is gewoon rekenen met als extra regel: $2 + 2 \to 5$, dat wil zeggen $2 + 2$ kan vereenvoudigd worden tot 5. Met deze extra regel kunnen we nu zowel het antwoord 20 als het antwoord 25 uit de som $(2+2) \times 5$

krijgen. Dus 'doublethink' rekenen is niet confluent. De oorzaak hiervan is dat de som $2+2$ op twee manieren uitgerekend kan worden. Ten eerste met de normale rekenregel $2+2 \to 4$ tot 4 en ten tweede met de 'doublethink' regel tot 5. De som $2+2$ noemen we een *fataal* splitsingspunt, omdat het uitrekenen ervan tot onverenigbare resultaten kan leiden.

Men zou kunnen aanvoeren dat dit een flauw voorbeeld is, want $2+2$ *is* niet gelijk aan 5. Echter, dit voorbeeld probeert alleen duidelijk te maken dat confluentie van een herschrijfsysteem bepaald wordt door de herschrijfregels van dat systeem. In dit proefschrift worden methoden bestudeerd om confluentie te bewijzen door het beschouwen van de vorm van de herschrijfregels.

Confluentie voor Abstract Herschrijven Hoe kan het dat een herschrijfsysteem niet confluent is? Zoals uit het bovenstaande duidelijk zal zijn, kan een systeem alleen niet confluent zijn als er sprake is van fatale splitsingspunten. De vraag wordt nu: hoe kunnen we garanderen dat er geen fatale splitsingspunten zijn? In Hoofdstuk 2 van dit proefschrift worden voorwaarden op de vorm van de herschrijfregels geformuleerd, die dit garanderen.

We illustreren de problematiek en de gevonden oplossing aan de hand van een eenvoudig (tegen)voorbeeld. Beschouw het herschrijfsysteem:

$$c_2 \leftarrow a \leftrightarrow b \rightarrow c_1$$

In dit systeem herschrijft a zowel naar b als naar c_2, en herschrijft b naar zowel a als naar c_1, dus a en b zijn splitsingspunten. Zijn het ook fatale splitsingspunten? In eerste instantie zou men geneigd kunnen zijn te zeggen nee, want het doen van één herschrijfstap vanuit a is nooit fataal: a kan herschreven worden naar zowel b als naar c_2 en die zijn te verenigen, want b kan herschreven worden naar c_2 (via a). In een plaatje:

$$\begin{tikzcd}
a \arrow[rr] \arrow[d] & & c_2 \arrow[d, phantom, "|||"] \\
b \arrow[r] & a \arrow[r] & c_2
\end{tikzcd}$$

Echter, vanuit a hebben we ook de paden $a \to b \to c_1$ en $a \to c_2$, die niet meer te verenigen zijn.

Het belangrijkste resultaat van Hoofdstuk 2 is de volgende voorwaarde die confluentie garandeert.

> Er is voor iedere splitsing ${}_b\swarrow^{a}\searrow_c$ een punt d, en er zijn paden p_b en p_c vanuit a naar d via respectievelijk b en c, zodanig dat het aflopen van p_b (p_c) niet 'moeilijker' is dan het doen van beide stappen vanuit a.

Deze voorwaarde is niet vervuld in het bovenstaande tegenvoorbeeld, als we het splitsingspunt ${}_b\swarrow^{a}\searrow_{c_2}$ beschouwen. Er zijn wel paden p_b: $a \to b \to a \to c_2$ en p_{c_2}: $a \to c_2$ naar het gemeenschappelijke punt c_2 (zie het plaatje), maar het uitvoeren van de drie stappen $a \to b$, $b \to a$ en $a \to c_2$ van het pad p_b is zeker moeilijker dan het uitvoeren van de twee stappen $a \to b$ en $a \to c_2$, die mogelijk zijn vanuit a.

Deze confluentie conditie zegt alleen iets over de vorm van de herschrijfregels, niets over de vorm van de 'objecten' waarop die regels van toepassing zijn. Of we in het voorbeeld bij a aan Amsterdam denken of aan appels, het herschrijfsysteem blijft niet confluent. Om deze reden wordt dit *abstract* herschrijven genoemd.

Confluentie voor Hogere-Orde Herschrijven Als de herschrijfregels in een herschrijfsysteem gebruik maken van de vorm van de objecten, dan is er sprake van *term*herschrijven. In zo'n geval is het ook zinvol om te spreken van regels die op verschillende *delen* van een term van toepassing zijn. We kunnen bijvoorbeeld een portemonnaie zien als een term bestaande uit de munten die in de portemonnaie zitten. Op een portemonnaie kunnen de volgende herschrijfregels ('wisselregels') toegepast worden, die de waarde van de inhoud ongemoeid laten:

- een gulden kan gewisseld worden voor vier kwartjes,
- vier kwartjes kunnen gewisseld worden voor een gulden.

Deze regels zijn in een zekere zin onafhankelijk van elkaar, of, *orthogonaal* in termherschrijfjargon, want ze zijn van toepassing op verschillende munten in de portemonnaie. Voor een grote klasse van termherschrijfsystemen is het bekend dat orthogonale regels confluentie garanderen. Dit kan bewezen worden met behulp van een 'eindige ontwikkelingen' stelling, die in het geval van dit voorbeeld zegt dat het zoveel mogelijk toepassen van de regels op de in het begin in de portemonnaie aanwezige munten een keer stopt en altijd dezelfde inhoud oplevert. (Dit is niet waar als ook de door wisselen verkregen munten weer gewisseld mogen worden, want dan kan een gulden gewisseld worden voor vier kwartjes en deze weer voor een gulden enz.)

In Hoofdstuk 3 wordt de 'eindige ontwikkelingen' stelling bewezen voor een zeer uitgebreide klasse van termherschrijfsystemen, de zogenaamde hogere-orde herschrijfsystemen. Dit wordt gedaan door middel van een nauwkeurige analyse van herschrijfstappen. Een wisselstap is opgebouwd uit drie onderdelen: eerst wordt een gulden in de portemonnaie opgezocht en eruit gehaald, vervolgens wordt deze gewisseld voor vier kwartjes en tenslotte worden de kwartjes in de portemonnaie gedaan. Ruwweg volgt het eindig zijn van 'ontwikkelingen' nu uit het feit dat maar een eindig aantal munten tegelijkertijd uit een portemonnaie gehaald kan worden.

Opmerking: Door gebruik te maken van alledaagse beelden om de ideeën in dit proefschrift voor het voetlicht te brengen, kan deze samenvatting een verkeerde indruk geven van het onderzoek dat is verricht voor de totstandkoming van het proefschrift. Een korte blik op de inhoud zal echter duidelijk maken dat het uitwerken van deze ideeën tot mathematisch precieze stellingen verre van alledaags is.