Centrum voor Wiskunde en Informatica

REPORTRAPPORT

Weak orthogonality implies confluence: the higher-order case

V. van Oostrom and F. van Raamsdonk

Computer Science/Department of Software Technology

CS-R9501 1995

Report CS-R9501
ISSN 0169-118X

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Weak Orthogonality Implies Confluence: the Higher-Order Case

Vincent van Oostrom
NTT Basic Research Laboratories
3-1 Wakamiya Morinosato, Atsugi-shi
Kanagawa 243-01
Japan
oostrom@theory.ntt.jp

Femke van Raamsdonk
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
femke@cwi.nl

Abstract

In this paper we prove confluence for weakly orthogonal Higher-Order Rewriting Systems. This generalises all the known 'confluence by orthogonality' results.

AMS Subject Classification (1991): 68Q50
CR Subject Classification (1991): F.4.1, F.3.3
Keywords & Phrases: higher-order rewriting, weak orthogonality, confluence.
Note: Most of the research of the first author has been carried out during his employment at the Vrije Universiteit, Amsterdam, The Netherlands. The research of the second author is supported by NWO/SION project 612-316-606.

1. Introduction

This paper deals with higher-order term rewriting. Since our approach of higher-order term rewriting is different from the usual one, both in respect to the concept of 'higher-order' and to the notion of 'term rewriting', we first comment on our approach and the terminology used, before stating the general confluence result.

term rewriting. In term rewriting as usually defined (see e.g. [DJ89, Klo92, Klo80, Nip91]) rewrite steps are generated by the rewrite rules via 'contexts and substitutions': in order to apply a rewrite rule $l \to r$ to some term s, one has to find a context $C[\,]$ and a substitution σ, such that s is the result of 'evaluating' $C[l^\sigma]$. If this is the case then s is said to rewrite in one step to the term t resulting from 'evaluating' $C[r^\sigma]$. In our treatment, the informal notion of 'evaluation' occurring here, is formalised by means of a calculus, named *substitution calculus.* A rewrite step then consists of a conversion in the substitution calculus, the actual replacement step (the left-hand side of the rule is replaced by the right-hand side), and another conversion in the substitution calculus. In other words, rewrite steps are defined as *replacement steps modulo the substitution calculus.* In any 'suitable' substitution calculus, the first conversion can actually be obtained by an *expansion* and the second conversion by a *reduction* in the substitution calculus.

One can wonder what kind of calculus the substitution calculus should be. Well, it should at least be able to mimic the evaluation of $C[l^\sigma]$ to s (and of $C[r^\sigma]$ to t). Now, noting that the only thing which happens in the evaluations is '(un)plugging' of terms, it seems reasonable to propose that the substitution calculus be some kind of lambda calculus. The definition of a rewrite step gives rise to the analogy *rewriting = substitution + rules*. Exploiting the Curry-Howard correspondence (in case the substitution calculus is a typed λ-calculus), one can view evaluation also as *proof transformations* of some *logic*, so we arrive at the analogy *rewriting = logic + rules*.

The previous paragraph may seem rather fanciful, but it actually works quite nicely. Ordinary term rewriting systems (TRSs), as well as Klops CRSs and Nipkows HRSs can be formalised easily in this way using simply typed lambda calculus with β-reduction and η-expansion as substitution calculus, as we will show.

higher-order term rewriting. From this formalisation of term rewriting, a natural way to classify term rewriting systems, becomes apparent: classify them according to the logic employed. Since we parametrise over the logic, not restricting attention to first order but allowing for any 'suitable', e.g. higher-order, logic, we can also handle higher-order term rewriting.

weak orthogonality. Orthogonality of two rewrite rules expresses that applications of those rules to a term always operate on different parts of the term. Weak orthogonality is a weaker assumption than orthogonality, because the rules may operate on the same part of a term, but in that case, both applications should result in exactly the same term.

weak orthogonality implies confluence. It is well-known that orthogonality implies confluence for many classes of term rewriting systems ([CR36, Ros73, Klo80, Raa93, Nip93]). Basically, two methods are used to prove this. The first method known as 'confluence via developments', is due to Church and Rosser [CR36], and employed in the first three papers above. The second method, due to Tait and Martin-Löf (see [Bar84]) is known as 'confluence via parallel reductions', and employed in the last two papers. In the case of weak-orthogonality (and generalisations thereof), confluence has been proved only for TRSs and was an open problem ([DJK93]) for CRSs and HRSs.

weak orthogonality implies confluence: the higher-order case. In this paper we prove confluence for the class of all weakly orthogonal higher-order term rewriting systems, for which the substitution calculus satisfies some, more or less natural conditions. This generalises all the known results. First, because confluence was only shown to hold for (admittedly large) subclasses of orthogonal term rewriting systems. Second, because confluence was only shown to hold for weakly orthogonal term rewriting systems, so far, not for either CRSs or HRSs.

We prove confluence both via 'developments' as well as via 'parallel reductions'. The confluence by developments proof works by a reduction to strong normalisation of cut-elimination of the employed logic. The confluence by parallel reductions proof works by proof transformations. Both methods differ substantially from the known methods.

organisation of the paper. First we illustrate our definition of term rewriting by presenting some examples. Then we give our formal definition of higher-order term rewriting systems (HORSs) and show how some common formats of term rewriting fit into this definition. Next, we motivate and present conditions on the substitution calculus allowing to derive the confluence by weak orthogonality result. The two confluence proofs are the topics of the next sections and the paper concludes with the conclusion.

A short version of the present paper has appeared as [OR94].

2. Use of a Substitution Calculus

In this section we illustrate the intended use of a substitution calculus by considering two examples. First we consider the term rewriting system

$$\begin{array}{rcl} x+0 & \rightarrow & 0 \\ x+S(y) & \rightarrow & S(x+y) \end{array}$$

We will use abbreviations of $S^n(0)$ whenever convenient. The first rule can be applied to the term $8+0$. In the usual definition of rewriting this is seen be remarking that $8+0=(x+0)^\theta$ with $\theta(x)=8$. In our definition of rewriting the substitution of 8 for x will be performed by the substitution calculus. It is quite natural to have as substitution calculus λ-calculus with β-reduction, the prime example of a calculus implementing substitution. If the substitution calculus is to act on the variable x, then we must change the rule in such a way that x will be of object-level, instead of of meta-level as in the rule in the usual format. With a λ-calculus as substitution calculus, this is done by turning x into a *bound variable.* We write _._ for abstraction and concatenation for application. The rules given above then take the following form:

$$\begin{array}{rcl} x.(x+0) & \rightarrow & x \\ x.y.(x+S(y)) & \rightarrow & x.y.(S(x+y)) \end{array}$$

Remember now that a rewrite step consists of a conversion in the substitution calculus, followed by a replacement of the left-hand side by the right-hand side, followed by a conversion in the substitution calculus. So a term M is rewritten to N if

$$M \leftrightarrow^*_{\mathcal{SC}} C[l] \rightarrow C[r] \leftrightarrow^*_{\mathcal{SC}} N$$

With simply typed λ-calculus with β-reduction and (restricted) η-expansion as substitution calculus, we obtain the following computation:

$$\begin{array}{rcl} (8+1)+1 & \leftarrow_\beta & \{y.((8+1)+S(y))\}\{0\} \\ & \leftarrow_\beta & \{x.y.(x+S(y))\}\{8+1\}\{0\} \end{array}$$

$$\begin{array}{cl}
\rightarrow & \{x.y.S(x+y)\}\{8+1\}\{0\} \\
\rightarrow_\beta & \{y.S((8+1)+y)\}\{0\} \\
\rightarrow_\beta & S((8+1)+0) \\
\leftarrow_\beta & S(\{x.(x+0)\}\{8+1\}) \\
\rightarrow & S(\{x.x\}\{8+1\}) \\
\rightarrow_\beta & S(8+1) \\
\leftarrow_\beta & S(\{y.(8+S(y))\}\{0\}) \\
\leftarrow_\beta & S(\{x.y.(x+S(y))\}\{8\}\{0\}) \\
\rightarrow & S(\{x.y.S(x+y)\}\{8\}\{0\}) \\
\rightarrow_\beta & S(\{y.S(8+y)\}\{0\}) \\
\rightarrow_\beta & S(S(8+0)) \\
\leftarrow_\beta & S(S(\{x.(x+0)\}\{8\})) \\
\rightarrow & S(S(\{x.x\}\{8\})) \\
\rightarrow_\beta & S(S(8))
\end{array}$$

Note that the replacement step is safe because left- and right-hand side of a rule are closed.

In the second example we consider a rewriting system involving bound variables. As usual things become more complicated in the presence of bound variables. The example concerns a rule for calculating the derivative of the sum of two arbitrary functions. Informally, this rule can be given as follows:

$$d_x(f(x)+g(x)) \rightarrow d_x(f(x))+d_x(g(x))$$

where f and g stand for arbitrary functions of one variable. The rule applies to the expression $d_x(x^2+5x)$, which is rewritten to $d_x(f(x))+d_x(g(x))$. This is the case because $d_x(x^2+5x) = (d_x(f(x)+g(x)))^\theta$ with $\theta(f) = x \mapsto x^2$ and $\theta(g) = x \mapsto 5x$. Not surprisingly, λ-calculus can take care of substitution also in this example. We now consider the rule and the rewrite step in our format, with again simply typed λ-calculus with β-reduction and (restricted) η-expansion as substitution calculus. First, the functions $x \mapsto x^2$ and $x \mapsto 5x$ are now denoted as $x.x^2$ and $x.5x$. Second, f and g become bound variables in our representation of the rewrite rule:

$$fg.d(y.(fy+gy)) \rightarrow fg.(z.(d(y.fy)z+d(y.gy)z))$$

The computation of $d_x(x^2+5x)$ is now as follows:

$$\begin{array}{rc}
d(y.y^2+5y) & \leftarrow_\beta \\
d(y.\{x.x^2\}\{y\}+\{x.5x\}\{y\}) & \twoheadleftarrow_\beta \\
\{fg.d(y.fy+gy)\}\{x.x^2\}\{x.5x\} & \rightarrow \\
\{fg.(z.(d(y.fy)z+d(y.gy)z))\}\{x.x^2\}\{x.5x\} & \twoheadrightarrow_\beta \\
z.(d(y.x.\{x^2\}\{y\})z+d(\{x.5x\}\{y\})z) & \rightarrow_\beta \\
z.(d(y.y^2)z+d(y.5y)z) &
\end{array}$$

Two remarks seem appropriate. First, like usual, we work modulo α-equivalence. Second, in both examples the recipe *conversion-replacement-conversion* is in fact used in the form *expansion-replacement-reduction.* This is not just good luck. In this paper we will be interested in rewriting of expressions that do not contain redexes for the substitution calculus. Moreover the substitution calculus is required to be complete. In that case, rewriting is *expansion-replacement-reduction.*

3. Higher-Order Rewriting Systems: syntax

In this section we give the definition of a Higher-Order Rewriting System. A Higher-Order Rewriting System is defined as a triple consisting of an alphabet, a substitution calculus and a set of rewrite rules: $\mathcal{H} = (\mathcal{A}, \mathcal{SC}, \mathcal{R})$.

The alphabet contains an operator for applications, one for abstraction and further nullary symbols. The substitution calculus has an associated rewrite relation, denoted by $\rightarrow_{\mathcal{SC}}$, on the set of expressions over the alphabet. This rewrite relation is to be thought of as implementing substitution. The rewrite rules determine the behaviour of a subset of symbols called the *defined symbols.*

We first take a closer look at the alphabet. The alphabet of each Higher-Order Rewriting System is supposed to contain an operator for application and an operator for abstraction.

Definition 3.1 An *alphabet* $\mathcal{A}$ of a Higher-Order Rewriting System consists of:
- a symbol Ap for the application operation,
- a symbol $\_.\_$ for abstraction,
- symbols $x\ y\ z\ \ldots$ for variables; among them are special symbols $\Box_1, \Box_2, \ldots$ for distinguished variables called *holes*,
- symbols $U\ V\ W\ \ldots$ for substitution operators,
- symbols $F\ G\ H\ \ldots$ for rewrite or defined operators.

The set of variables is denoted by Var. The set of symbols for substitution operators is denoted by $\mathcal{O}_{\mathcal{SC}}$. The set of symbols for rewrite operators is denoted by $\mathcal{O}_{\mathcal{R}}$. The union of $\mathcal{O}_{\mathcal{SC}}$ and $\mathcal{O}_{\mathcal{R}}$ is denoted by $\mathcal{O}$. We use $a, a', a'', \ldots$ to denote an arbitrary element of $\mathcal{O}$.

The substitution operators are used by the substitution calculus to implement substitution. The rewrite operators are given an operational semantics by the rewrite rules. The sets $\mathcal{O}_{\mathcal{SC}}$ and $\mathcal{O}_{\mathcal{R}}$ are supposed to be disjoint.

Expressions over $\mathcal{A}$ are called *preterms.*

Definition 3.2 The set $\mathsf{PreTerms}$ of *preterms* is defined as the least set satisfying:

(1) $x \in \mathsf{PreTerms}$ for every variable $x \in \mathsf{Var}$,
(2) $a \in \mathsf{PreTerms}$ for every operator $a \in \mathcal{O}$,
(3) if $M_0 \in \mathsf{PreTerms}$ and $M_1 \in \mathsf{PreTerms}$, then $\mathsf{Ap}(M_0, M_1) \in \mathsf{PreTerms}$,
(4) if $M \in \mathsf{PreTerms}$ and $x \in \mathsf{Var}$ then $x.M \in \mathsf{PreTerms}$.

A variable x occurs *free* in a preterm M if it occurs not in the scope of an abstraction $x.\_$, and it occurs *bound* otherwise. The set of variables that occur free in a preterm M is denoted by $\mathsf{FVar}(M)$, and the set of variables that occur bound in M is denoted by $\mathsf{BVar}(M)$. By the Variable Convention, one may assume $\mathsf{FVar}(M) \cap \mathsf{BVar}(M) = \emptyset$. If all variables occur bound in a preterm, then the preterm is said to be *closed*. It is convention not to bind over variables that are holes.

Ths substitution symbols are the substitution operators and the bound variables. The rewrite symbols are the free variables and the rewrite operators. Note that one is usually interested in what happens to rewrite operators during substitution. Note that it might be cleaner to make a syntactic distinction between free and bound variables, because then the definition of substitution symbols and of rewrite symbols is independent of the terms we are working with. This set-up is chosen in [Oos94].

Notation 3.3 We write $M_0 M_1$ for $\mathsf{Ap}(M_0, M_1)$. We write $x_1 \ldots x_n.M$ for $x_1 . \ldots . x_n . M$.

A *precontext* is defined as a preterm in which all occurrences of holes are made explicit. If the holes occurring in a precontext are among $\Box_1, \ldots, \Box_n$, then it is called an n-ary precontext, and it is denoted by $C[, \ldots,]$. A unary precontext is denoted by $C[\,]$. For a unary precontext we usually don't make the index of the hole occurring in it explicit. The result of replacing occurrences of $\Box_1, \ldots, \Box_n$ by preterms $M_1, \ldots, M_n$ is denoted by $C[M_1, \ldots, M_n]$. We suppose that if a hole is replaced by a preterm, the result is a well-formed preterm. An n-ary context is said to be *linear* if every hole $\Box_i$ (for $i = 1, \ldots, n$) occurs exactly once in it.

A *position* is a finite word over $\{0, 1\}$. Positions are denoted by ϕ, ψ, χ. The set $\{0, 1\}^*$ of positions is denoted by Pos. The empty word over $\{0, 1\}^*$ is denoted by ϵ. It is the neutral element for the concatenation operation, which is denoted by $\cdot$. Concatenation is associative. On Pos a prefix ordering denoted by $\leq$ is defined as follows: $\phi \leq \psi$ if and only if there exists a ϕ' such that $\phi \cdot \phi' = \psi$. In that case ϕ is called a *prefix* of ψ. If for $\phi, \psi \in \mathsf{Pos}$, ϕ is not a prefix of ψ and ψ is not a prefix of ϕ, then ϕ and ψ are said to be *disjoint*.

Definition 3.4 Let M be a preterm. The set of positions of M, $\mathsf{Pos}(M)$, the head-symbol of M, $\mathsf{top}(M)$, and the subterm of M at position ϕ, $\phi\backslash t$ are defined by induction on the structure of M as follows:

- if $M = x$, then
$$\begin{aligned} \mathsf{Pos}(M) &= \{\epsilon\} \\ \mathsf{top}(M) &= x \\ \epsilon\backslash M &= x \end{aligned}$$
- if $M = a$ with $a \in \mathcal{O}$, then
$$\begin{aligned} \mathsf{Pos}(M) &= \{\epsilon\} \\ \mathsf{top}(M) &= a \\ \epsilon\backslash M &= a \end{aligned}$$
- if $M = M_0 M_1$, then
$$\begin{aligned} \mathsf{Pos}(M) &= \{\epsilon\} \cup \{0 \cdot \phi \mid \phi \in \mathsf{Pos}(M_0)\} \cup \{1 \cdot \phi \mid \phi \in \mathsf{Pos}(M_1)\} \\ \mathsf{top}(M) &= \mathsf{Ap} \\ \epsilon\backslash M &= M \\ 0 \cdot \phi\backslash t &= \phi\backslash M_0 \end{aligned}$$

$$1 \cdot \phi \backslash t \quad = \quad \phi \backslash M_1$$

- if $M = x.M_0$, then

$$\begin{array}{rcl} \mathsf{Pos}(M) & = & \{\epsilon\} \cup \{0 \cdot \phi \mid \phi \in \mathsf{Pos}(M_0)\} \\ \mathsf{top}(M) & = & x. \\ \epsilon \backslash M & = & M \\ 0 \cdot \phi \backslash M & = & \phi \backslash M' \end{array}$$

If $\phi_1 = \psi \cdot 0 \cdot \phi_1'$ and $\phi_2 = \psi \cdot 1 \cdot \phi_2'$, then ϕ_1 is *on the left of* ϕ_2. We also say that the symbol at ϕ_1 is on the left of the symbol at ϕ_2.

We now consider the properties a decent substitution calculus should have in order to deserve the name.

A substitution calculus is meant to implement substitution. One would like that calculating a substitution yields a result, and moreover, that this result is unique. This is guaranteed by requiring the substitution calculus to be complete, that is, confluent and terminating.

If some calculations in the substitution calculus concerning some closed term M are done, we want to be able to use these calculations for a larger term having M as subterm. For hygienic reasons it is required that rewriting in the substitution calculus preserves closedness of a term. Further we require that if there is a conversion in the substitution calculus between closed terms, $M \leftrightarrow^*_{\mathcal{SC}} M'$, then there is the same conversion in a context: $C[M] \leftrightarrow^*_{\mathcal{SC}} C[M']$.

In the same spirit, if we have a conversion in the substitution calculus between two terms $C[\ ] \leftrightarrow^*_{\mathcal{SC}} C'[\ ]$ where $[\ ]$ denotes a hole which is possibly present in $C'[\ ]$, we can replace the hole by a closed term. Between the results of the replacement there still is a conversion: $C[M] \leftrightarrow^*_{\mathcal{SC}} C'[M]$.

Finally one remark: for the moment, we ignore typing problems and we assume the preterms that are considered to be well-formed. For example, if the substitution calculus is simply typed λ-calculus, we assume all terms to be simply typable.

The rewrite relation of the substitution calculus is denoted by $\rightarrow_{\mathcal{SC}}$.

The requirements on the substitution calculus discussed above are listed in the next definition.

Definition 3.5 The rewrite relation of a substitution calculus $\mathcal{SC}$ must satisfy the following requirements:

(1) (completeness)
The rewrite rules of a substitution calculus generate a confluent and terminating rewrite relation on the set of expressions over $\mathcal{A}$.

(2) (closed under closed)
If M is closed and $M \rightarrow M'$, then M' is closed.

(3) (closed under contexts)
The conversion relation $\leftrightarrow^*_{\mathcal{SC}}$ generated by the rewrite rules of a substitution calculus $\mathcal{SC}$ is closed under contexts, i.e. if $M \leftrightarrow^*_{\mathcal{SC}} M'$ is a conversion between closed terms, then $C[M] \leftrightarrow^*_{\mathcal{SC}} C[M']$.

(4) (closed under substitutions)
The conversion relation $\leftrightarrow^*_{\mathcal{SC}}$ generated by the rewrite rules of a substitution calculus is closed under substitution, i.e. if $C[\,] \leftrightarrow^*_{\mathcal{SC}} C'[\,]$ then $C[M] \leftrightarrow^*_{\mathcal{SC}} C'[M]$.

The convertibility relation of the substitution calculus is an equivalence relation on the set of preterms. Rewriting in a Higher-Order Rewriting System will be defined modulo the convertibility relation of the substitution calculus. By completeness of the substitution calculus, each equivalence class has a unique representative, which is found by reducing any member of the equivalence class to $\mathcal{SC}$-normal form. Mostly we are interested in the representatives of the equivalence classes, that do not contain redexes for the substitution calculus.

Definition 3.6 A preterm that is in normal form with respect to the substitution calculus is a *term*. The set of terms is denoted by Terms.

All notions defined for preterms persist for terms, delete if necessary the prefix *pre*.

Now the moment is there to discuss the rewrite rules of a Higher-Order Rewriting System.

Definition 3.7 A *rewrite rule* of a Higher-Order Rewriting System is a pair (l, r) of closed terms with the same outermost abstractions in the same order. Usually we write $l \to r$ for (l, r).

As usual, the rewrite rules induce a rewrite relation. We define the rewrite relation on the set of terms. The idea is that there is a rewrite step $M \to N$ if M equals modulo the substitution calculus the left-hand side of some rewrite rule in a context, that is, $M \leftrightarrow^*_{\mathcal{SC}} C[l]$, and N equals modulo the substitution calculus the right-hand side of the same rewrite rule in the same context, that is, $C[r] \leftrightarrow^*_{\mathcal{SC}} N$. Since M and N are *terms* (not preterms) and the substitution calculus is complete, this idea can be simplified. For the first conversion one can take an expansion and for the second conversion one can take a reduction.

Definition 3.8 A term M rewrites to a term N, notation $M \to N$, if there is a unary context $C[\,]$ and a rewrite rule $l \to r$ such that $M \,_{\mathcal{SC}}\!\twoheadleftarrow C[l]$ and $C[r] \twoheadrightarrow_{\mathcal{SC}} N$.

Notation 3.9 The transitive closure of $\to$ is denoted by $\twoheadrightarrow^+$, and its reflexive-transitive closure by $\twoheadrightarrow$.

The definition of a Higher-Order Rewriting System is now completed. We conclude this section by making some remarks.

In the definition of a rewrite rule and a rewrite step, some restrictions seem to have been imposed: a rewrite rule is a pair of *terms*, not a pair of preterms, and the rewrite relation is defined using a *context*, not a precontext. That these are no real restrictions is due to the last three requirements on the substitution calculus. The proofs can be found in [Oos94].

Further, in the definition of the rewrite relation the context is *unary*. It is possible to formulate requirements on the substitution calculus that guarantee the rewrite relation defined using a unary context to be as expressive as the rewrite relation defined using an arbitrary context. This is not done in the present paper. The interested reader is referred to [Oos94].

Finally, in this paper we restrict attention to rewriting on the set of terms. There certainly are good reasons to consider also rewriting on the set of preterms. It is for instance very natural to introduce sharing by means of the substitution calculus. This matter has our concern but it is beyond the scope of the present paper.

4. Examples of Higher-Order Rewriting Systems

In this section we represent some well-known rewriting systems as a Higher-Order Rewriting Systems. In all the examples the substitution calculus is λ-calculus with β-reduction and η-expansion. This illustrates the expressive power of Higher-Order Rewriting Systems.

4.1 Term Rewriting Systems.

Every term rewriting system is a Higher-Order Rewriting System. We illustrate this fact by considering two examples.

The term rewriting system for Combinatory Logic. Consider the term rewriting system describing Combinatory Logic:

$$\begin{array}{rcl} Ix & \to & x \\ Kxy & \to & x \\ Sxyz & \to & xz(yz) \end{array}$$

This is the usual representation of the term rewriting system describing Combinatory Logic. In fact, MN is an abbreviation for $@(M, N)$ with @ a binary operator for application. The symbols I, K and S denote nullary operators. For the representation of Combinatory Logic as a Higher-Order Rewriting System, we consider the rewrite rules in full detail:

$$\begin{array}{rcl} @(I, x) & \to & x \\ @(@(K, x), y) & \to & x \\ @(@(@(S, x), y), z) & \to & @(@(x, z), @(y, z)) \end{array}$$

Combinatory Logic as a Higher-Order Rewriting System. We shall now present this system as an Higher-Order Rewriting System. The set of defined symbols consists of $@ : 0 \to 0 \to 0, I : 0, K : 0$ and $S : 0$. There are no symbols for substitution operators. Using this alphabet, we can represent every term of Combinatory Logic as a term in CL. For instance, $@(I, x)$ is written as $\mathsf{Ap}(\mathsf{Ap}(@, I), x)$ and $@(@(K, I), S)$ is written as $\mathsf{Ap}(\mathsf{Ap}(K, I), S)$. Note that we can build many terms that do not correspond to a 'real' term in Combinatory Logic, like for instance $\mathsf{Ap}(I, x)$. This is in general the case when representing an existing system as a Higher-Order Rewriting System.

The free variables in the rules of the term rewriting system representing Combinatory Logic are turned into object variables. This is done by turning the left- and the right- hand side of the rules into closed expressions. The rewrite rules then take the following form:

$$\begin{aligned} x.@Ix &\rightarrow x.x \\ x.y.@(@Kx)y &\rightarrow x.y.x \\ x.y.z.@(@(@Sx)y)z &\rightarrow x.y.z.@(@xy)(@yz) \end{aligned}$$

An example of a well-known rewrite sequence is the following:

$$\begin{array}{rl} @(@(@SI)I)(@(@SI)I) & \twoheadleftarrow_{\beta} \\ \{x.y.z.@(@(@Sx)y)z\}\{I\}\{I\}\{@(@SI)I\} & \rightarrow \\ \{x.y.z.@(@xz)(@yz)\}\{I\}\{I\}\{@(@SI)I\} & \twoheadrightarrow_{\beta} \\ @(@I(@(@SI)I))(@I(@(@SI)I)) & \leftarrow_{\beta} \\ @(\{x.@Ix\}\{@(@SI)I\})(@I(@(@SI)I)) & \rightarrow \\ @(\{x.x\}\{@(@SI)I\})(@I(@(@SI)I)) & \rightarrow_{\beta} \\ @(@(@SI)I)(@I(@(@SI)I)) & \leftarrow_{\beta} \\ @(@(@SI)I)(\{x.@Ix\}\{@(@SI)I\}) & \rightarrow \\ @(@(@SI)I)(\{x.x\}\{@(@SI)I\}) & \rightarrow_{\beta} \\ @(@(@SI)I)(@(@SI)I) & \end{array}$$

The term rewriting system for parallel or Also term rewriting systems that are in functional format can be presented as a Higher-Order Rewriting System. As an example we consider the following term rewriting system for parallel or:

$$\begin{aligned} \mathtt{por}(\mathtt{tt},x) &\rightarrow \mathtt{tt} \\ \mathtt{por}(x,\mathtt{tt}) &\rightarrow \mathtt{tt} \\ \mathtt{por}(\mathtt{ff},\mathtt{ff}) &\rightarrow \mathtt{ff} \end{aligned}$$

The alphabet of this term rewriting system consists of a binary symbol `por` and the nullary symbols `tt` and `ff`.

Parallel or as a Higher-Order Rewriting System. The alphabet of the Higher-Order Rewriting System that is associated to the term rewriting system describing parallel or consists of two constants `tt` and `ff` of type 0 and one constant `por` of type $0 \rightarrow 0 \rightarrow 0$. The rewrite rules are as follows:

$$\begin{aligned} x.\mathtt{por}(\mathtt{tt})(x) &\rightarrow x.\mathtt{tt} \\ x.\mathtt{por}(x)(\mathtt{tt}) &\rightarrow x.\mathtt{tt} \\ \mathtt{por}(\mathtt{ff})(\mathtt{ff}) &\rightarrow \mathtt{ff} \end{aligned}$$

We have the following computation:

$$\begin{array}{rl} \mathtt{por}(\mathtt{por}(\mathtt{ff})(\mathtt{tt}))(\mathtt{por}(\mathtt{ff})(\mathtt{ff})) & \rightarrow \\ \mathtt{por}(\mathtt{por}(\mathtt{ff})(\mathtt{tt}))(\mathtt{ff}) & \leftarrow_{\beta} \\ \mathtt{por}(\{x.\mathtt{por}(x)(\mathtt{tt})\}\{\mathtt{ff}\})(\mathtt{ff}) & \rightarrow \end{array}$$

$$\begin{array}{rl} \texttt{por}(\{x.\texttt{tt}\}\{\texttt{ff}\})(\texttt{ff}) & \rightarrow_\beta \\ \texttt{por}(\texttt{tt})(\texttt{ff}) & \leftarrow_\beta \\ \{x.\texttt{por}(\texttt{tt})(x)\}\{\texttt{ff}\} & \rightarrow \\ \{x.\texttt{tt}\}\{\texttt{ff}\} & \rightarrow_\beta \\ \texttt{tt} & \end{array}$$

4.2 λ-calculus.

A prime example of a Higher-Order Rewriting System is of course λ-calculus. In this example we present λ-calculus with β- and η-reduction as a Higher-Order Rewriting System.

λ-calculus. Traditionally the rewrite rules are given as follows:

$$\begin{array}{rll} (\lambda x.M)N & \rightarrow_{\texttt{beta}} & M[x := N] \\ \lambda x.Mx & \rightarrow_{\texttt{eta}} & M \quad \text{if } x \text{ doesn't occur free in } M \end{array}$$

λ-calculus as a Higher-Order Rewriting System. The alphabet of the Higher-Order Rewriting System representation of $\lambda\beta\eta$-calculus contains the following symbols for operators:

$$\begin{array}{lcl} \texttt{app} & : & 0 \rightarrow (0 \rightarrow 0) \\ \texttt{abs} & : & (0 \rightarrow 0) \rightarrow 0 \end{array}$$

Then, for instance MN is represented as $\texttt{app}MN$ and $\lambda x.M$ as $\texttt{abs}(x.M)$. The rewrite rules for β- and η-reduction are as follows:

$$\begin{array}{rll} z.z'.\texttt{app}(\texttt{abs}(x.zx))(z') & \rightarrow_{\texttt{beta}} & z.z'.zz' \\ z.\texttt{abs}(x.\texttt{app}zx) & \rightarrow_{\texttt{eta}} & z.z \end{array}$$

Note that the side-condition for the $\texttt{eta}$-rule is not necessary. In an attempt to minimise confusion we note that the rewrite relations in the substitution calculus is denoted as $\rightarrow_\beta$ and $\rightarrow_{\overline{\eta}}$, whereas the rewrite relations of the object calculus is written as $\rightarrow_{\texttt{beta}}$ and $\rightarrow_{\texttt{eta}}$.

We give a beginning of the reduction sequence of the term Ω:

$$\begin{array}{rl} \texttt{app}(\texttt{abs}(x.\texttt{app}xx))(\texttt{abs}(x.\texttt{app}xx)) & {}_\beta\!\twoheadleftarrow \\ \{z.z'.\texttt{app}(\texttt{abs}(x.zx))(z')\}\{x'.\texttt{app}x'x'\}\{\texttt{abs}(x.\texttt{app}xx)\} & \rightarrow_{\texttt{beta}} \\ \{z.z'.zz'\}\{x'.\texttt{app}x'x'\}\{\texttt{abs}(x.\texttt{app}xx)\} & \twoheadrightarrow_\beta \\ \{x'.\texttt{app}x'x'\}\{\texttt{abs}(x.\texttt{app}xx)\} & \rightarrow_\beta \\ \texttt{app}(\texttt{abs}(x.\texttt{app}xx))(\texttt{abs}(x.\texttt{app}xx)) & \end{array}$$

Another example of a rewriting sequence:

$$\begin{array}{rl} \texttt{app}(\texttt{abs}(y.\texttt{abs}(x.\texttt{app}yx)))(\texttt{abs}(v.\texttt{app}uv)) & {}_\beta\!\twoheadleftarrow \\ \{z.z'.\texttt{app}(\texttt{abs}x.zx)(z')\}\{y.\texttt{abs}(x.\texttt{app}yx)\}\{\texttt{abs}(v.\texttt{app}uv)\} & \rightarrow_{\texttt{beta}} \\ \{z.z'.zz'\}\{y.\texttt{abs}(x.\texttt{app}yx)\}\{\texttt{abs}(v.\texttt{app}uv)\} & \twoheadrightarrow_\beta \end{array}$$

$$\begin{array}{rl}
\{y.\mathtt{abs}(x.\mathtt{app}yx)\}\{\mathtt{abs}(v.\mathtt{app}uv)\} & \rightarrow_\beta \\
\mathtt{abs}(x.\mathtt{app}(\mathtt{abs}v.\mathtt{app}uv)(x)) & {}_\beta\!\leftarrow \\
\{z.\mathtt{abs}(x.\mathtt{app}zx)\}\{\mathtt{abs}(v.\mathtt{app}uv)\} & \rightarrow_{\mathtt{eta}} \\
\{z.z\}\{\mathtt{abs}(v.\mathtt{app}uv)\} & \rightarrow_\beta \\
\mathtt{abs}(v.\mathtt{app}uv) &
\end{array}$$

4.3 Interaction Systems.

We present Interaction Systems as Higher-Order Rewriting Systems. Interaction Systems form a class of higher-order rewriting systems that has been defined by Asperti and Laneve [AL92]. They form a subclass of the class of Combinatory Reduction Systems (see next subsection).

Interaction Systems. We start by recalling briefly the definition of an Interaction System.

An Interaction System is a pair $<\Sigma, \mathcal{R}>$ of a signature Σ and a set of rewrite rules $\mathcal{R}$. The signature Σ consists of

- a denumerable set of variables written as $x\ y\ z\ \ldots$,
- a set of forms written as $f\ g\ h\ \ldots$, each equipped with a fixed arity.

The alphabet $\mathcal{A}$ of an Interaction system $<\Sigma, \mathcal{R}>$ consists of

- symbols in Σ,
- a symbol $\_.\_$ for abstraction over variables,
- symbols $X\ Y\ Z\ \ldots$ for metavariables,
- for every n a symbol $[\_/\_, \ldots, \_/\_]$ for metasubstitution, with n occurrences of $\_/\_$.

Note that $t[t_1/x_1, \ldots, t_n/x_n]$ denotes the result of replacing x_i by t_i in t for $i = 1, \ldots, n$. The set of forms is divided into two disjoint sets Σ^+ and Σ^-, the first one containing forms that act as a *constructor* and the second one containing forms that act as a *destructor*. Each form has an arity, which is a finite sequence of natural numbers. The length of the sequence specifies the number of arguments a form is supposed to get. If the arity of some form f is $k_1 \ldots k_n$, then the ith argument is supposed to start with k_i abstractions. All destructors have an arity of the form $0k_2 \ldots k_n$.

The set $\mathcal{T}$ of *expressions* is defined inductively as follows:

- every variable x is an expression,
- if $f \in \Sigma$ is a form of arity $k_1 \ldots k_n$ and $t_1, \ldots, t_n$ are expressions, then $f(x_{11}.\ldots.x_{1k_1}.t_1, \ldots, x_{n1}.\ldots.x_{nk_n}.t_n)$ is an expression.

Often we abbreviate $x_1.\ldots.x_n.t$ by $\vec{x_n}.t$. The notion of free and bound variable is as usual. Expressions that are equal up to a renaming of bound variables are identified.

A *metaexpression* is an expression in which possibly metavariables and metasubstitutions occur.

Rewrite rules generate a rewrite relation on the set of expressions. A rewrite rule is a pair of metaexpressions often written as $l \rightarrow r$.

The left-hand side of a rewrite rule must satisfy:

- it is of the form $f_d(f_c(\vec{x_{l_1}}.X_1, \ldots, \vec{x_{l_m}}.X_m), \vec{x_{k_2}}.Y_2, \ldots, \vec{x_{k_n}}.Y_n)$ with $f_c \in \Sigma^+, f_d \in \Sigma^-$,
- all metavariables are different,
- there are no occurrences of metasubstitutions.

The right-hand side of a rewrite rule must satisfy

- it is a closed metaexpression,
- all metavariables occurring in it occur also in the left-hand side.

A right-hand side contains possibly metasubstitutions of the form $X[t_1/x_1, \ldots, t_n/x_n]$.

The set of rewrite rules $\mathcal{R}$ satisfies the property that for every pair consisting of a constructor and a destructor there is at most one rewriting rule.

The rewrite relation $\rightarrow$ is defined as follows: $t \rightarrow t'$ if $t = C[l^\theta]$ and $s = C[r^\theta]$ for a rewriting rule $l \rightarrow r$, a context $C[\,]$ and an assignment θ. Contexts are defined as usual. An assignment assigns expressions to metavariables.

An example of an Interaction System is λ-calculus. There are two forms: @ of arity 00 for application and λ of arity 1 for λ-abstraction. The rule for β-reduction then takes the following form:

$$@(\lambda(x.X), Y) \rightarrow X[Y/x]$$

Interaction Systems as Higher-Order Rewriting Systems. We now associate a Higher-Order Rewriting System to an Interaction System $< \Sigma, \mathcal{R} >$.

First we associate to an arity of the form $k_1 \ldots k_n$ a simple type built from 0 and $\rightarrow$. Define k^* inductively as follows:

$$\begin{array}{rcl} 0^* & = & 0 \\ (n+1)^* & = & 0 \rightarrow n^* \end{array}$$

To an arity $k_1 \ldots k_n$ we then associate the type $k_1^* \rightarrow \ldots \rightarrow k_n^* \rightarrow 0$.

The alphabet of the Higher-Order Rewriting System associated to an Interaction System $< \Sigma, \mathcal{R} >$ consists of the following:

- symbols $x\ y\ z\ \ldots$ for typed variables,
- a symbol $\_.\_$ for abstraction over variables,
- a symbol Ap for application,
- for every form f of arity $k_1 \ldots k_n$ in Σ, we have a symbol f of type $k_1^* \rightarrow \ldots \rightarrow k_n^* \rightarrow 0$ for an operator.

As usual we write $t_1 t_2$ for $\mathsf{Ap}(t_1, t_2)$.

Now we translate the expressions of the Interaction System $< \Sigma, \mathcal{R} >$ into terms of the Higher-Order Rewriting System. The definition is by induction on the structure of an expression. We write t^* for the translation of an expression t.

- a variable x is translated into a variable x of type 0,
- an expression $f(\vec{x_{k_1}}.t_1, \ldots, \vec{x_{k_n}}.t_n)$ is translated into $f(\vec{x_{k_1}}.t_1^*) \ldots (\vec{x_{k_n}}.t_n^*)$.

Note that the translation of an expression of an Interaction System is a term of type 0.

Now we come to the point of translating the rewrite rules. The first thing to be done is turning the metavariables into object variables, and abstract over them. Next we have to take care of substitution. In the left-hand side, we replace each subexpression of the form $x_1. \ldots x_n.X$ into a subexpression $x_1 \ldots x_n.xx_1 \ldots x_n$. Here, x is a variable of type $0 \to \ldots \to 0 \to 0$ ($n+1$ times a zero). It is abstracted over on the outside of the left-hand side. This is sufficient to translate left-hand side of rewrite rules.

In the right-hand side, we replace subexpressions of the form $X[t_1/x_1, \ldots, t_k/x_k]$ by a subexpression $x(t_1) \ldots (t_n)$. Here, $t_i = x_i$ of x_i doesn't occur in the metasubstitution. Again, x is a variable of the right type that is abstracted over on the outside of the right-hand side. This is sufficient for translating right-hand sides of rewrite rules.

It is now easy to see that if an expression t is in fact $C[l^\theta]$, then its translation t^* equals modulo the substitution calculus the translation of l is some context. It is then almost immediate that the set of translated rewrite rules induces the right rewrite relation.

4.4 Combinatory Reduction Systems.

In this example we consider the class of Combinatory Reduction Systems defined by Klop [Klo80]. It forms a generalisation of the class of Contraction Schemes introduced by Aczel [Acz78].

Combinatory Reduction Systems. First we will highlight the particular points of the definition of a Combinatory Reduction System. We will follow the definition of Combinatory Reduction Systems as given in [KOR93]. The main difference between this definition and the original one in [Klo80] is that it employs the functional format, whereas the original presentation is in applicative format. For a detailed account the reader may wish to consult [KOR93]. Next, we represent a particular Combinatory Reduction System, the one describing or-elimination in natural deduction, as a Higher-Order Rewriting System.

A Combinatory Reduction System is a pair consisting of an alphabet and a set of rewrite rules. The alphabet consists of

- variables, written as $x\ y\ z \ldots$,
- metavariables, each with a fixed arity, written as Z_i^k, where k is the arity of Z_i^k,
- function symbols, each with a fixed arity,
- an operator for abstraction over variables, written as $[\cdot]\cdot$,
- improper symbols '(', ')' and ','.

Metaterms and terms are distinguished. Metaterms are expressions built from the symbols in the alphabet in the usual way. Terms are metaterms that do not contain any occurrence of a metavariable. In this way there is on a syntactical level a distinction between the objects that actually interest us, the terms, and 'metaobjects', the metaterms, that can be used to express a relation on the set of terms. The typical way to use metaterms is in rewrite rules. The metavariables represent the 'holes' that must be instantiated in order to obtain a rewrite step.

The β-reduction rule of λ-calculus is in the Combinatory Reduction System format written as

$$@(\lambda([x]Z(z)), Z') \to Z(Z')$$

A rewrite rule of a Combinatory Reduction System is a pair of metaterms, written as $l \to r$. A rewrite rule must satisfy some restrictions we will not mention here.

As usual, the rewrite rules induce a rewrite relation on the set of terms. Extracting the rewrite relation from the rewrite rules is a rather delicate business in Combinatory Reduction Systems. The basic idea is that an instance of a left- or right-hand side of a rule is obtained by first replacing each metavariable by a special kind of λ-term and then performing a development of all special β-redexes created by this replacement.

We will explain this in some more detail. In order to define valuations we must first consider the so-called *substitute*. This will be the 'special λ-term' mentioned above.

An n-ary substitute is an expression of the form $\underline{\lambda}(x_1, \ldots, x_n).M$, with M a term and $x_1, \ldots, x_n$ different variables. An n-ary substitute $\underline{\lambda}(x_1, \ldots, x_n).M$ can be applied to an n-tuple of terms $(M_1, \ldots M_n)$. This results in a simultaneous substitution of M_i for x_i for $i = 1, \ldots, n$:

$$(\underline{\lambda}(x_1, \ldots, x_n).M)(N_1, \ldots, N_n) = M[x_1 \mapsto N_1 \ldots x_n \mapsto N_n]$$

A *valuation* σ is a map assigning an n-ary substitute to an n-ary metavariable:

$$\sigma(Z) = \underline{\lambda}(x_1, \ldots, x_n).M$$

A valuation is extended to a mapping from metaterms to terms in the following way:

$$\begin{array}{rcl} x^\sigma & = & x \\ ([x]M)^\sigma & = & [x]M^\sigma \\ F(M_1, \ldots, M_n)^\sigma & = & F(M_1^\sigma, \ldots M_n^\sigma) \\ Z(M_1, \ldots, M_n)^\sigma & = & \sigma(Z)(M_1^\sigma, \ldots, M_n^\sigma) \end{array}$$

We suppose unintended bindings like in $([x]Z)^\sigma$ where $\sigma(Z) = x$ to be ruled out by the variable convention.

A rewrite step is now defined in the usual way: is $l \to r$ is a rewrite rule, σ a valuation and $C[\,]$ a context, then $C[l^\sigma]$ rewrites to $C[r^\sigma]$.

The Combinatory Reduction System for elimination introduction as a Higher-Order Rewriting System. We will now consider the representation of a particular Combinatory Reduction System as a Higher-Order Rewriting System, but first let us make some remarks about the canonical translation. It seems tempting to translate the symbol $[\cdot]$ for abstraction in a Combinatory Reduction System straightforwardly into the symbol for abstraction $\cdot.\cdot$ of a

Higher-Order Rewriting System. However, this causes typing problems. Suppose for instance that the alphabet of some Combinatory Reduction System contains a unary operator denoted by F. Then both $F(x)$ and $F([x]x)$ are perfectly legal as terms in the Combinatory Reduction System. But it cannot be the case that both Fx and $F(x.x)$ are simply typable.

The solution of this problem is as follows. Like in the case of term rewriting, an operator F of arity n is translated into F of type $0 \to \ldots \to 0 \to 0$ with $n+1$ times a zero. Like in the translation of a Combinatory Reduction System into a Higher-order Rewrite System, we add an operator $\texttt{abs} : (0 \to 0) \to 0$ that collapses a functional type. Intuitively, the type 0 can be thought of as the set of all terms. A subterm of the form $[x]t$ in a Combinatory Reduction System is then represented as $\texttt{abs}(x.t^*)$ (with t^* the representation of t) in a Higher-Order Rewriting System.

So the translation of a Combinatory Reduction System into a Higher-Order Rewriting System requires actually an encoding of untyped λ-calculus in simply typed λ-calculus. Exactly the same is going on in the representation of a Combinatory Reduction System as a Higher-order Rewrite System as defined by Nipkow. This has been reported in [OR93].

Now we will represent one particular Combinatory Reduction System as a Higher-Order Rewriting System. The Combinatory Reduction System we will consider concerns rules taken from proof theory for the elimination of the disjunction. In natural deduction, rules for elimination of $\vee$ is as follows:

$$\dfrac{\dfrac{\begin{matrix}\vdots\\ A\end{matrix}}{A \vee B} \quad \begin{matrix}[A]\\ \vdots\\ C\end{matrix} \quad \begin{matrix}[B]\\ \vdots\\ C\end{matrix}}{C} \;\to\; \begin{matrix}\vdots\\ A\\ \vdots\\ C\end{matrix} \qquad\qquad \dfrac{\dfrac{\begin{matrix}\vdots\\ B\end{matrix}}{A \vee B} \quad \begin{matrix}[A]\\ \vdots\\ C\end{matrix} \quad \begin{matrix}[B]\\ \vdots\\ C\end{matrix}}{C} \;\to\; \begin{matrix}\vdots\\ B\\ \vdots\\ C\end{matrix}$$

These rules can be written in the formalism of Combinatory Reduction Systems. They then take the following form:

$$\begin{array}{lcl}\texttt{el}(\texttt{inl}(Z),[x]Z_0(x),[y]Z_1(y)) & \to & Z_0(Z)\\ \texttt{el}(\texttt{inr}(Z),[x]Z_0(x),[y]Z_1(y)) & \to & Z_1(Z)\end{array}$$

So the alphabet of this Combinatory Reduction System consists of two unary function symbol `inl` and `inr` (for introduction of disjunction) and a ternary function symbol `el` (for elimination of disjunction).

In fact, the Combinatory Reduction System above models the conversion rules concerning disjunction only in a typed setting. Typed Combinatory Reduction Systems are not officially introduced, but the essentials for this particular example are in the following:

- if t is a term of type α, then $\texttt{inl}(t)$ is a term of type $\alpha \vee \beta$,
- if t is a term of type β, then $\texttt{inr}(t)$ is a term of type $\alpha \vee \beta$,

- if s and t are terms of type γ, u a term of type $\alpha \vee \beta$, and x and y are variables of type α and β, then $\mathtt{el}(u, [x]s, [y]t)$ is a term of type γ.

An example of a rewrite step is the following. We take the assignment σ defined as follows:

$$\begin{array}{rcl} Z & \mapsto & \mathtt{inr}(u) \\ Z_0 & \mapsto & \underline{\lambda}(x).\mathtt{el}(x, [u]u, [u']u) \\ Z_1 & \mapsto & \underline{\lambda}(y).\mathtt{inr}(y), [u]u', [u']u') \end{array}$$

Then we have:

$$\begin{array}{rc} \mathtt{el}(\mathtt{inl}(\mathtt{inr}(u)), [x]\mathtt{el}(\mathtt{inl}(x), [u]u, [u']u), [y]\mathtt{el}(\mathtt{inr}(y), [u]u, [u']u)) & = \\ \mathtt{el}(\mathtt{inl}(\mathtt{inr}(u)), [x]\underline{\lambda}(x').\mathtt{el}(\mathtt{inl}(x'), [u]u, [u']u)(x), [y]\underline{\lambda}(y').\mathtt{el}(\mathtt{inr}(y'), [u]u, [u']u)(y) & = \\ \mathtt{el}(\mathtt{inl}(Z), [x]Z_0(x), [y]Z_1(y))^\sigma & \to \\ Z_0(Z)^\sigma & = \\ \underline{\lambda}(x').\mathtt{el}(\mathtt{inl}(x'), [u]u, [u']u)(\mathtt{inr}(u)) & = \\ \mathtt{el}(\mathtt{inl}(\mathtt{inr}(u)), [u]u, [u']u) & \end{array}$$

In the format of Higher-Order Rewriting Systems, the system is written as

$$\begin{array}{rcl} z.z_0.z_1.\mathtt{el}(\mathtt{inl}z)(x.z_0x)(y.z_1y) & \to & z.z_0.z_1.z_0z \\ z.z_0.z_1.\mathtt{el}(\mathtt{inr}z)(x.z_0x)(y.z_1y) & \to & z.z_0.z_1.z_1z \end{array}$$

Like in the other examples, we take simply typed λ-calculus with β-reduction and (restricted) η-expansion as substitution calculus. The rewrite step mentioned above is simulated in the setting of Higher-Order Rewriting Systems as follows:

$$\begin{array}{rc} \mathtt{el}(\mathtt{inl}(\mathtt{inr}u))(x.\mathtt{el}(\mathtt{inl}x)(u.u)(u'.u))(y.\mathtt{el}(\mathtt{inr}y)(u.u)(u'.u)) & {}_\beta\twoheadleftarrow \\ \{z.z_0.z_1.\mathtt{el}(\mathtt{inl}z)(x.z_0x)(y.z_1y)\}\{\mathtt{inr}u\}\{x'.\mathtt{el}(\mathtt{inl}x')(u.u)(u'.u)\}\{y'.\mathtt{el}(\mathtt{inr}y')(u.u)(u'.u)\} & \to \\ \{z.z_0.z_1.z_0z\}\{\mathtt{inr}u\}\{x'.\mathtt{el}(\mathtt{inl}x')(u.u)(u'.u)\}\{y'.\mathtt{el}(\mathtt{inr}y')(u.u)(u'.u)\} & \twoheadrightarrow_\beta \\ \{x'.\mathtt{el}(\mathtt{inl}x')(u.u)(u'.u)\}\{\mathtt{inr}u\} & \to_\beta \\ \mathtt{el}(\mathtt{inl}(\mathtt{inr}u))(u.u)(u'.u) & \end{array}$$

4.5 Expression Reduction Systems.

Khasidashvili introduced a framework for higher-order rewriting under the name of Expression Reduction Systems. The definition of Expression Reduction Systems was introduced around 1986. An early reference is [Kha90]. In other publications they are sometimes also called Combinatory Reduction Systems. The development of Expression Reduction Systems has been influenced by work by Pkhakadze. Expression Reduction Systems are quite similar to Combinatory Reduction Systems as introduced by Klop, but independently developed.

Expression Reduction Systems. First we shortly recall the basics of the definition of an Expression Reduction System. We use the definition as given in [Kha94]. An Expression Reduction System is a pair (Σ, R) consisting of an alphabet and a set of rewrite rules.

Definition 4.1 The *alphabet* of an Expression Reduction Systems consists of

- object metavariables written as $a, a', a'', \ldots$,
- term metavariables written as $A, A', A'', \ldots$,
- variables written as $x, y, z, \ldots$,
- function symbols with a fixed arity k written as $f, g, h, \ldots$,
- quantifier symbols with a fixed arity (m, n), where $m \neq 0$ and $n \neq 0$, written as $\sigma, \rho, \tau, \ldots$,
- symbols $(\_/\_, \ldots, \_/\_)$ for metasubstitutions.

The arity of a function symbols prescribes the number of argument it is supposed to have, like in term rewriting systems. The first component of the arity of a quantifier symbol prescribes how many variables it binds. The second component indicates how many arguments it is supposed to have. An example of a quantifier symbol is λ of λ-calculus. Its arity is $(1, 1)$.

Like in Combinatory Reduction Systems, *terms* and *metaterms* are distinguished.

Definition 4.2 The set of metaterms is the smallest set satisfying the following:

- a variable x is a metaterm,
- an object metavariable a is a metaterm,
- a term metavariable A is a metaterm,
- if f is a function symbol of arity n and $t_1, \ldots, t_n$ are metaterms, then $f(t_1, \ldots, t_n)$ is a metaterm,
- if σ is a quantifier symbol of arity (m, n), and $b_1, \ldots, b_m$ are variables or object metavariables and $t_1, \ldots, t_n$ are metaterms then $\sigma b_1 \ldots b_m(t_1, \ldots, t_n)$ are metaterms,
- if $a_1, \ldots, a_n$ are object metavariables and $t, t_1. \ldots, t_n$ are metaterms, then $(t_1/a_1, \ldots, t_n/a_n)t$ is a metaterm.

The construct $(t_1/a_1, \ldots, t_n/a_n)$ in the last clause of the previous definition is called a *metasubstitution.* A metaterm without metasubstitutions is a *simple metaterm.* A metaterm without any occurrence of object metavariables of term metavariables is a *term.* In $\sigma b_1 ... b_n$ and in $(t_1/b_1, \ldots, t_n/b_n)$, the variables or object metavariables $b_1, \ldots, b_n$ are called *binding variables.* It is easier to understand how things work if we first look at the definition of a rewrite rule in an Expression Reduction System.

Definition 4.3 A *rewrite rule* of an Expression Reduction System is a pair of metaterms usually written as $l \rightarrow r$ satisfying the following conditions:

1. l is a simple metaterm which first symbol is a function symbol or a quantifier symbol,
2. l and r do not contain variables,
3. occurrences of object metavariables in l and in r are bound,
4. term metavariables occurring in r occur also in l,

Note that r may contain occurrences of an object metavariable that doesn't occur in l. Such an object metavariable is called an *additional object metavariable.*

As usual, the rewrite rules induce a rewrite relation on the set of terms. We have that s is rewritten to t, notation $s \to t$, if $s = C[l^\theta]$ and $t = C[r^\theta]$. Here $C[\,]$ is a context, and θ is an assignment due to some restrictions discussed below.

The definition of a rewrite rule in an Expression Reduction System is very liberal with respect to binding of variables. For instance, the pathological rule

$$f(A) \to (c/a)A$$

is perfectly legal. Here a is an additional object metavariable. By restricting the ways such a rule may be used the rewrite relation is prevented from becoming pathological.

An *assignment* is a mapping that maps object metavariables to variables and term metavariables to terms.

Definition 4.4 An assignment is *admissible* for a rewrite rule $l \to r$ if the following is satisfied:

If one occurrence of $\theta(A)$ in $l^\theta \to r^\theta$ is in the scope of a binding variable $\theta(a)$, then all occurrences of $\theta(A)$ are in the scope of the binding variable $\theta(a)$.

In the example of the pathological rule $f(A) \to (c/a)A$ above, the instance $f(x) \to c$ is for instance not allowed.

The β- and η-reduction rules of λ-calculus are in the format of Expression Reduction Systems written as

$$\begin{array}{rcl} \mathsf{Ap}(\lambda a(A), B) & \to & (B/a)A \\ \lambda a(\mathsf{Ap}(A, a)) & \to & A \end{array}$$

An instance $(s/x)t$ of the right-hand side denotes the term t in which each free occurrence of x has been replaced by s.

Expression Reduction Systems as Higher-Order Rewriting Systems. We sketch the translation of an Expression Reduction System into a Higher-Order Rewriting System with λ-calculus with β-reduction and restricted η-expansion as a substitution calculus.

First variables, function symbols and quantifier symbols are translated. A variable is translated into a variable of type 0. A function symbol f of arity k is translated into a rewrite operator of type $0 \to \ldots \to 0 \to 0$ with $k+1$ times a 0. Let σ be a quantifier symbol of arity (m, n). Define m^* as follows:

$$\begin{array}{rcl} 0^* & = & 0 \\ (m+1)^* & = & 0 \to m^* \end{array}$$

The translation of σ is then a rewrite operator of type $m^* \to \ldots \to m^* \to 0$ with n times m^*.

The translation t^* of a term t is as follows:

- $x^* = x$,

- $f(t_1, \ldots, t_n)^* = f(t_1^*) \ldots (t_n^*)$,
- $\sigma x_1 \ldots x_m(t_1, \ldots, t_n) = \sigma(x_1 \ldots x_m.t_1^*) \ldots (x_1 \ldots x_m.t_n^*)$.

Remark 4.5 In Expression Reduction Systems to each quantifier symbol a *scope indicator* is associated. It indicates in which arguments the binding variables actually bind. We consider a simplified version of Expression Reduction Systems without scope indicator. Then, binding variables of a quantifier symbol bind in all arguments of that quantifier symbol.

As far as rules are concerned, we consider here a translation of a modified version the rewrite rules of an Expression Reduction System. Note that if a term metavariable A is in the scope of a binding (object meta)variable a, this binding may only play a role in an actual instance of the rewrite rule if *all* occurrences of A are in the scope of the binding variable a. Therefore, we choose to translate a modified version of the rewrite rules where we forget about bindings that never play a role in an actual instance of the rule.

The translation of a modified rule is then as follows. Let t be a metaterm that is the left- or right-hand side of a rewrite rule. We first associate to t a term t^* as follows:

- an object metavariable is translated into a variable of type 0,
- the translation of $f(t_1, \ldots, t_n) == f(t^*) \ldots (t_n^*)$,
- $\sigma a_1 \ldots a_m(t_1, \ldots, t_n)^* = \sigma(x_{a_1} \ldots x_{a_m}.t_1^*) \ldots (x_{a_1} \ldots x_{a_m}.t_n^*)$
- a metasubstitution $(t_1/a_1, \ldots, t_n/a_n)t$ is translated into $t^*(t_1^*) \ldots (t_n^*)$,
- a term metavariable A that is in the scope of binding variables $a_1, \ldots a_n$ of quantifier symbols is translated into $z_A x_{a_1} \ldots x_{a_n}$ where z_A is a variable of type $0 \rightarrow \ldots \rightarrow 0 \rightarrow 0$ with $n+1$ times a 0,
- a term metavariable A in a subterm of the form $(t_1/a_1, \ldots, t_n/a_n)A$ is translated into $z_A(t_1^*) \ldots (t_n^*)$.

Next we take the closure $z_1 \ldots z_n.l^*$ of the left-hand side. The translation of a rule $l \rightarrow r$ is then $z_1 \ldots z_n.l^* \rightarrow z_1 \ldots z_n.r^*$.

Note that things, and in particular the modified version of a rewrite rule of an Expression Reduction System are not sufficiently formalised yet. Work certainly remains to be done here, and this sketch is only meant to be a first step.

For instance, we should take care that the translation of a metasubstitution is well-typed. This is only the case if metasubstitutions apply to metaterms. It is possible to modify rewrite rules of an Expression Reduction System (without changing the rewrite relation) in such a way that this holds, but this is one of the things that remain to be done in a formal way.

Then, of course, it remains to prove that the translation is correct. That is, we have to prove that if $t \rightarrow s$ in an Expression Reduction System, then $t^* \rightarrow s^*$ in the associated Higher-Order Rewriting System. We do not give the proof in detail here, but just consider some translations of rewrite rules.

The translation of $f(A) \to (c/a)A$ is $z.fz \to z.z$. An admissible assignment θ in the Expression Reduction System may not assign to A a term containing free occurrences of $\theta(a)$. All other instances are easily seen to be simulated in the associated Higher-Order Rewriting System.

The translation of $\sigma a(A) \to f(\sigma a(A), A)$ is $z.\sigma Z \to z.f(\sigma z)(z)$. Again, admissible assignments θ may not assign to A a term containing free occurrences of $\theta(a)$.

The translation of $\mathsf{Ap}(\lambda a(A), B) \to (B/a)A$ is $z_A z_B.\mathsf{Ap}(\lambda(x_a.z_A x_A))(z_B) \to z_A z_B.z_A z_B$, as it should be.

4.6 Higher-order Rewrite Systems.

In this example we consider a class of higher-order rewriting systems introduced by Nipkow: the Higher-order Rewrite Systems [Nip91].

Higher-order Rewrite Systems. We first recall the definition of a Higher-order Rewrite System; since Higher-Order Rewriting Systems are very similar to them we can be really quick here.

Expressions of a Higher-order Rewrite System are built from simply typed variables, abstraction and application and simply typed constants as in simply typed λ-calculus. The expressions we are interested in are the ones in $\beta\overline{\eta}$-normal form. They are called terms. A context is a term with one occurrence of a hole. A substitution is the homomorphic extension of a type-preserving mapping from variables to terms. A rewrite rule is a pair of terms written as $l \to r$ satisfying some restrictions of which we mention only two here:

- l and r are terms of the same base-type,
- l satisfies the so-called pattern-condition, i.e. every occurrence of a free variable x is in a subterm of the form $x(t_1)\ldots(t_n)$, such that $t_1{\downarrow_\eta}, \ldots, t_n{\downarrow_\eta}$ is a list of distinct bound variables.

The Higher-order Rewrite System for 'mini-scoping' as a Higher-Order Rewriting System. As an example of a Higher-order Rewrite System, we consider the system 'mini-scoping' that pushes quantifiers inwards. It is taken from [Nip91]. There are two base types, `term` and `form`. The system contains the following constants:

$$\begin{aligned} \wedge, \vee &: \mathtt{form} \to \mathtt{form} \to \mathtt{form} \\ \forall &: (\mathtt{term} \to \mathtt{form}) \to \mathtt{form} \end{aligned}$$

The rewrite rules are as follows, where $P, Q : \mathtt{form}$ and $P', Q' : \mathtt{term} \to \mathtt{form}$ are free variables:

$$\begin{aligned} \forall(x.P) &\to P \\ \forall(x.((P'x) \wedge (Q'x))) &\to (\forall P') \wedge (\forall Q') \\ \forall(x.((P'x) \vee Q)) &\to (\forall P') \vee Q \\ \forall(x.(P \vee (Q'x))) &\to P \vee (\forall Q') \end{aligned}$$

An example of a rewrite sequence is:

$$\forall(x.(\forall(y.a) \vee (bx \vee b'))) \to$$

$$\begin{array}{rl}
\forall(y.a) \vee \forall(z.(bz \vee b')) & \rightarrow \\
a \vee \forall(z.(bz \vee b')) & \rightarrow \\
a \vee (\forall(b) \vee b') &
\end{array}$$

We now represent this Higher-order Rewrite System as a Higher-Order Rewriting System. The substitution calculus is simply typed λ-calculus. The only thing that should be done is turning taking the closure of the rules. We then obtain the following Higher-Order Rewriting System:

$$\begin{array}{rcl}
P.\forall(x.P) & \rightarrow & P.P \\
P'.Q'.(\forall(x.(P'x \wedge Q'x))) & \rightarrow & P'.Q'.((\forall P') \wedge (\forall Q')) \\
P'.Q.(\forall(x.(P'x \vee Q))) & \rightarrow & P'.Q.((\forall P') \vee Q) \\
P.Q'.(\forall(x.(P \vee Q'x))) & \rightarrow & P.Q'.(P \vee (\forall Q'))
\end{array}$$

The rewrite sequence above is then obtained as follows:

$$\begin{array}{rl}
\forall(x.(\forall(y.a) \vee (bx \vee b'))) & \twoheadleftarrow_{\beta} \\
\{P.Q'.(\forall(x.(P \vee Q'x)))\}\{\forall(y.a)\}\{z.(bz \vee b')\} & \rightarrow \\
\{P.Q'.P \vee (\forall Q')\}\{\forall(y.a)\}\{z.(bz \vee b')\} & \twoheadrightarrow_{\beta} \\
\forall(y.a) \vee \forall(z.(bz \vee b')) & \leftarrow_{\beta} \\
\{P.\forall(x.P)\}\{a\} \vee \forall(z.(bz \vee b')) & \rightarrow \\
\{P.P\}\{a\} \vee \forall(z.(bz \vee b')) & \rightarrow_{\beta} \\
a \vee \forall(z.(bz \vee b')) & \twoheadleftarrow \\
a \vee \{P'.Q.\forall(z.((P'z) \vee Q))\}\{x.bx\}\{b'\} & \rightarrow \\
a \vee \{P'.Q.(\forall P') \vee Q\}\{x.bx\}\{b'\} & \twoheadleftarrow_{\beta} \\
a \vee (\forall(x.bx)) \vee b') &
\end{array}$$

5. Weak Orthogonality

Orthogonality of two computations means that the two computations are independent of each other. If computation is modelled by a rewriting system then one is usually not interested in independency of steps but in independency of rules. Two rewrite rules are orthogonal to each other if always if they both can be applied to a certain term, they use different 'resources' of this term. A rewriting system is said to be orthogonal if each pair of rules is. Traditionally, one imposes orthogonal behaviour on a rewriting system by requiring all rules to be left-linear and by requiring each pair of rules to be non-ambiguous.

Two rewrite rules are said to be weakly orthogonal to each other if whenever they can both be applied to a certain term using (partly) the same resources, the result of applying the one rule is the same as the result of applying the other rule.

Under the restriction of orthogonality, confluence has been proven for Combinatory Reduction Systems [Klo80, Raa93] and for Higher-order Rewrite Systems [Nip93]. In the next two sections we give two proofs of confluence for weakly orthogonal Higher-Order Rewriting Systems. This extends already existing results because the requirement of orthogonality is relaxed to weak orthogonality and because the class of Higher-Order Rewriting Systems covers all systems for which a confluence proof has been already given. proofs have been given for so far. It solves a problem which was raised in [DJK93, Problem 61].

Since our format of rewriting differs from the usual one, the reader won't find the familiar definition of orthogonality in this text. We try however to make our presentation the least shocking as possible. In this section we first discuss orthogonality and then weak orthogonality. The definition of orthogonality concerns on the one hand the substitution calculus and on the other hand the rewrite rules.

5.1 Orthogonality

The substitution calculus. We first consider the part of the definition of orthogonality that concerns the substitution calculus.

In rewriting, one is often interested in tracing what happens to symbols, or rather to positions of symbols. What happens to a position in a term M during a rewrite sequence $M \twoheadrightarrow N$ is described by means of a descendant relation, relating positions of M to positions of N. In Higher-Order Rewriting Systems, we will be interested in what happens to free variables and defined symbols during rewrite sequences. Since the rewrite relation of a Higher-Order Rewriting System is defined via the rewrite relation of its substitution calculus, it is natural to define a descendant relation for a Higher-Order Rewriting System via the descendant relation of its substitution calculus.

Therefore we add to the requirements on the substitution calculus that it should have a descendant relation. We first consider the definition of such a descendant relation.

Definition 5.1

1. A descendant relation of the substitution calculus maps a step $u : M \rightarrow_{\mathcal{SC}} N$ to a relation $\underrightarrow{\lfloor u \rfloor}_{\mathcal{SC}}$ between positions of M and positions of N. If $u : M \rightarrow_{\mathcal{SC}} N$ is a rewrite step and $\phi \in \mathsf{Pos}(M)$, $\psi \in \mathsf{Pos}(N)$, then by $\phi \underrightarrow{\lfloor u \rfloor} \psi_{\mathcal{SC}}$ is meant that the position ϕ in M descends to the position ψ in N by the step $u : M \rightarrow_{\mathcal{SC}} N$.
2. The descendant relation is extended straightforwardly to arbitrary rewrite sequences (and conversions) by defining $\underrightarrow{\lfloor u \cdot u_1 \cdot \ldots \cdot u_n \rfloor}_{\mathcal{SC}} = \underrightarrow{\lfloor u \rfloor}_{\mathcal{SC}} \underrightarrow{\lfloor u_1 \cdot \ldots \cdot u_n \rfloor}_{\mathcal{SC}}$ and $\underrightarrow{\lfloor u^{-1} \rfloor}_{\mathcal{SC}} = \underrightarrow{\lfloor u \rfloor}^{-1}_{\mathcal{SC}}$. The descendant relation of the empty rewrite sequence is the identity.

Not every descendant relation is useful for tracing interesting symbols during a rewrite sequence. We impose some natural restrictions on the descendant relation of the substitution calculus. The restrictions, imposing 'naturality' are given in the next definition.

One can also consider a descendant relation and some natural restrictions on it in a more abstract setting where the objects have no visible structure. We are not interested in abstract rewriting in the present paper, but the interested reader is referred to [Oos94].

Definition 5.2 Let $\underrightarrow{\lfloor\ \rfloor}_{\mathcal{SC}}$ be the descendant relation of the substitution calculus $\mathcal{SC}$. It is said to be *natural* if the following holds:

1. Let $C[\,]$ be a unary context with $u : C[\,] \to_{\mathcal{SC}} D[\,]$. Let the step u' be obtained by replacing the hole by a closed term M: $u' : C[M] \to_{\mathcal{SC}} D[M]$. Then the positions of M in $C[M]$ are related to the positions of M in $D[M]$ via the positions of the hole, and the positions of $C[\,]$ in $C[M]$ are related to the positions of $D[\,]$ in $D[M]$ via $\underrightarrow{\lfloor u \rfloor}_{\mathcal{SC}}$.

   That is, for $\phi \in \mathsf{Pos}(C[\,])$ and $\psi \in \mathsf{Pos}(D[\,])$, we have

   $$\phi \underrightarrow{\lfloor u' \rfloor} \psi \;\text{ if }\; \phi \underrightarrow{\lfloor u \rfloor} \psi$$

   For $\chi \in \mathsf{Pos}(M)$, ϕ' the position of the hole in $C[\,]$ and ψ' a position of the hole in $D[\,]$, we have

   $$\phi' ; \chi \underrightarrow{\lfloor u' \rfloor} \psi' ; \chi$$

2. For two reductions to $\mathcal{SC}$-normal form $d_1 : M \twoheadrightarrow_{\mathcal{SC}} M'$ and $d_2 : M \twoheadrightarrow_{\mathcal{SC}} M'$ we have $\underrightarrow{\lfloor d_1 \rfloor}_{\mathcal{SC}} = \underrightarrow{\lfloor d_2 \rfloor}_{\mathcal{SC}}$.

The rewrite rules. We now consider the definition of orthogonality as far as it concerns the rewrite rules.

To start with, we want that for a left-hand side l of some rule, the $\mathcal{SC}$-normal form of $C[l]$ contains always a trace of l. More precisely, we require a left-hand side l to have a special position that has exactly one descendant in $C[l]{\downarrow_{\mathcal{SC}}}$ for every context $C[\,]$. This special position will be called the head-position of l.

Second, we require the following. If there is an expansion $M_{\mathcal{SC}} \twoheadleftarrow C[l]$ such that some position ϕ in M originates from the head-position of l in $C[l]$, then $C[\,]$ is unique. The expansion itself is of course not necessarily unique.

The first requirement on the form of a rewrite rule is given in the following definition.

Definition 5.3 A rewrite rule $l \to r$ is said to be *head-defined* if there is a unique position ϕ of l, called the *head-position*, that satisfied the following.

1. For every linear context $C[\,]$, the position ϕ has exactly one descendant in $C[l]{\downarrow_{\mathcal{SC}}}$,

2. for every term M and every position ψ in M, the linear context $C[\,]$ such that ψ originates from ϕ in $C[l]$ via $M_{\mathcal{SC}} \twoheadleftarrow C[l]$ is unique.

Note that in every 'reasonable' rewriting system all rules have a head-symbol. In first-order term rewriting, it is the leftmost symbol of the left-hand side. In Combinatory Reduction Systems, it is also the leftmost symbol of the left-hand side of a rule.

In a Higher-Order Rewriting System with simply typed λ-calculus with β-reduction and (restricted) η-expansion as substitution calculus, the head-symbol of a rewrite rule is the leftmost defined symbol of the left-hand side.

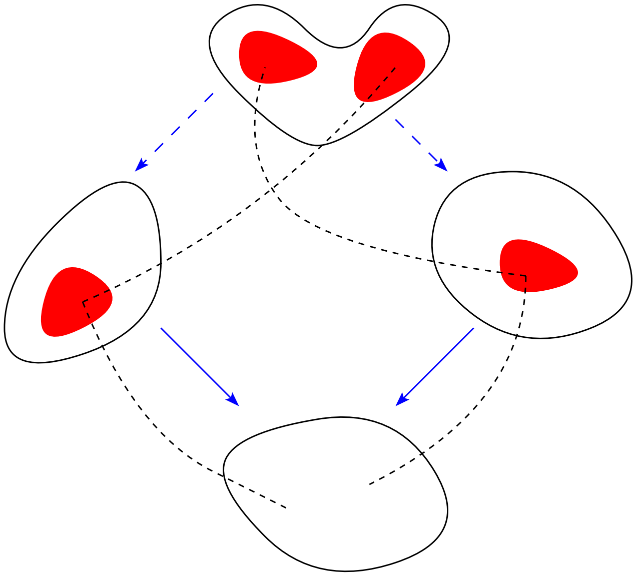
In general, the definition of a head-symbol depends on the substitution calculus and its descendant relation.

Note that it is often the case that a rewrite rule is required to have a head-symbol by definition.

The second requirement on a rewrite rule is that its left-hand side is *linear*, which is formulated as follows: if l is $x_1 \ldots x_n.l_0$, all variables $x_1, \ldots x_n$ occur exactly once in l_0. A somewhat more sophisticated definition of linearity is given in [Oos94]. A rule is said to be *left-linear* if its left-hand side is linear.

Finally, we consider a adaptation of the concept of 'non-ambiguity'. The idea of non-ambiguity is that if two rules can be applied to a term they use different parts of the term. This idea can be formalised using expansions.

Definition 5.4 Two rewrite rules $l \rightarrow r$ and $l' \rightarrow r'$ are said to be *non-ambiguous* or *simultaneous* if the following holds. Let a term M contain a redex for a rule $l \rightarrow r$ and one for a rule $l' \rightarrow r'$. Then there are expansions $M \twoheadleftarrow_{\mathcal{SC}} C[l]$ and $M \twoheadleftarrow_{\mathcal{SC}} C'[l']$. We require that both left-hand sides can be made explicit together in a (binary) context. That is, there is an expansion $M \twoheadleftarrow_{\mathcal{SC}} D[l, l']$. If both redexes occur on the same position this is only required if the two rules are different.



a pair of non-ambiguous or simultaneous rewrite steps

We recapitulate the above in the following definition.

Definition 5.5 A Higher-Order Rewriting System $\mathcal{H} = (\mathcal{A}, \mathcal{SC}, \mathcal{R})$ is *orthogonal* if the following is satisfied.

1. $\mathcal{SC}$ has a natural descendant relation,
2. every rule in $\mathcal{R}$ is head-defined,
3. every rule in $\mathcal{R}$ is left-linear,
4. every pair of rules of $\mathcal{R}$ is non-ambiguous or simultaneous.

5.2 Weak Orthogonality

The difference between orthogonality and weak orthogonality only lies in the point of non-ambiguity.

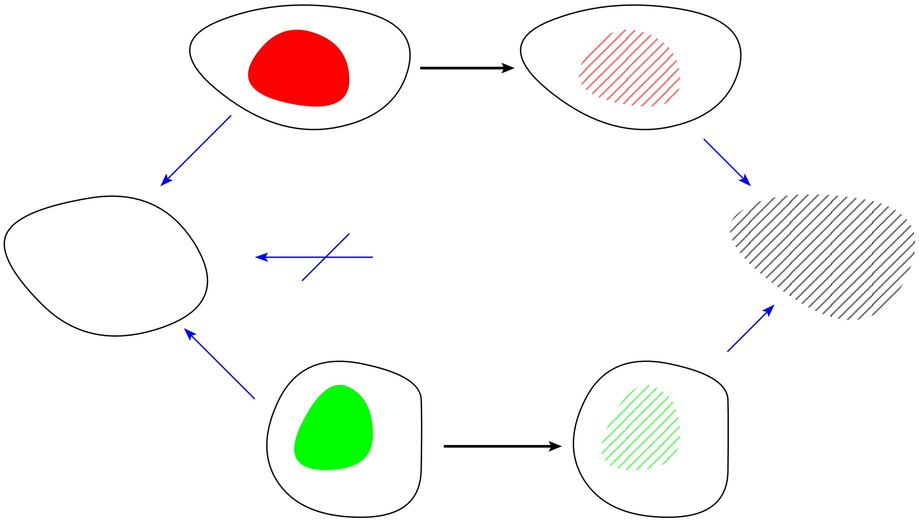
The requirement that two rules cannot operate on the same part of a term is relaxed to requiring that in case they do, both applications should yield exactly the same result.

Definition 5.6 Two rewrite rules $l \to r$ and $l' \to r'$ are said to be *weakly non-ambiguous* or *weakly simultaneous* if the following holds.

If we have $M_{\mathcal{SC}} \twoheadleftarrow C[l]$ and $M_{\mathcal{SC}} \twoheadleftarrow C'[l']$, and it is not the case that $M_{\mathcal{SC}} \twoheadleftarrow D[l, l']$, then $C[r]{\downarrow_{\mathcal{SC}}} = C'[r']{\downarrow_{\mathcal{SC}}}$.

Definition 5.7 A Higher-Order Rewriting System $\mathcal{H} = (\mathcal{A}, \mathcal{SC}, \mathcal{R})$ is *weakly orthogonal* if

1. $\mathcal{SC}$ has a natural descendant relation,
2. every rule in $\mathcal{R}$ is head-defined,
3. every rule in $\mathcal{R}$ is left-linear,
4. every pair of rules of $\mathcal{R}$ is weakly non-ambiguous.



a pair of weakly non-ambiguous rewrite steps

Now that we know what a weakly orthogonal Higher-Order Rewriting System is, we embark on the two confluence proofs.

6. A confluence proof by developments

In this subsection we prove all weakly orthogonal Higher-Order Rewriting Systems to be confluent by extending the method of ‘confluence by developments’ to the weakly orthogonal case. Before formalising the proof, we first present the proof idea.

A classical way to prove confluence for orthogonal rewriting systems is via the Finite Developments theorem. It states that rewriting all the redexes which are present ‘simultaneously’ in an initial term, in any order, is finite, always results in the same term, and induces the same descendant relation. This implies confluence if any set of redexes is indeed simultaneous.

If a rewriting system is orthogonal, then any set of redexes present in a term is simultaneous. Orthogonality in fact consists of three parts. First, distinct actions consume distinct resources ('consistency'). Second, actions may interact as long as this interaction is finitary ('finiteness'). 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 three conditions correspond to Axiom 0 in [GLM92].

The standard 'long' proof to show that orthogonal systems are confluent is via the parallel moves lemma ([HL91]). That is, one can construct the following diagram

$$\begin{tikzcd}
M_0 \arrow[r, "u_0"'] \arrow[d, "v"] & M_1 \arrow[r, "u_1"'] \arrow[d, dashed, "v"] & M_2 \arrow[r, two heads] \arrow[d, dashed, "v"] & M_n \arrow[d, dashed, "v"] \\
N_0 \arrow[r, dashed, two heads, "u_0"'] & N_1 \arrow[r, dashed, two heads, "u_1"'] & N_2 \arrow[r, dashed, two heads] & N_n
\end{tikzcd}$$

in which in $N_i \twoheadrightarrow N_{i+1} \twoheadleftarrow M_{i+1}$ only descendants of the rewrite steps on the opposite side are contracted. The essence of this construction is, that there exists for each term M_i a set of simultaneous redexes $\mathcal{U}_i$ in M_i, such that there exist complete developments $d : M_i \twoheadrightarrow M_{i+1} \twoheadrightarrow N_{i+1}$ and $d' : M_i \twoheadrightarrow N_i \twoheadrightarrow N_{i+1}$ 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 $V := \{v \mid \exists u \in \mathcal{U}_i . u \underrightarrow{\lfloor u_i \rfloor} v\}$ of residuals of $\mathcal{U}_i$ in M_{i+1}. But then we know by weak orthogonality, that there exists some step $u' \in V$ doing exactly the same as u_{i+1}, hence by starting with this step u', we obtain a complete development of V which 'goes through' M_{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. Moreover, one needs that if the redex u_{i+1} does not overlap with any redex in V, then the set $V \cup \{u_{i+1}\}$ is simultaneous again.

After having explained the idea informally, we will formalise it now.

Definition 6.1 Let $u : M \to N$ be a rewrite step, consisting of the expansion $e : M \,{}^{!}\!\twoheadleftarrow C[l]$, the replacement step $C[l \to r] : C[l] \to C[r]$ and the reduction $d : C[r] \twoheadrightarrow^{!} N$. The descendant relation induced by u is defined by $\underrightarrow{\lfloor u \rfloor} := \underrightarrow{\lfloor e \rfloor}; \underrightarrow{\lfloor C[l \to r] \rfloor}; \underrightarrow{\lfloor d \rfloor}$, where ; denotes relation composition. Descendants of redexes are defined via the descendant of their head-symbol.

Definition 6.2 Let $\mathcal{U} = \{u_1, \ldots, u_n\}$ be a set of redexes in a term M, where $u_i = (\phi_i, l_i \to r_i)$ is a redex at position ϕ_i in M with respect to rule $l_i \to r_i$.

a A rewrite sequence d starting from M is a *$\mathcal{U}$-development* if only descendants of redexes in $\mathcal{U}$ are reduced along d. It is *complete* if it ends in a term not containing any descendants of $\mathcal{U}$.

b The set $\mathcal{U}$ of redexes is called *simultaneous* if $d' : M \twoheadleftarrow C[l_1, \ldots, l_n]$, the head-symbol of l_i descends to ϕ_i along d', with $C[, \ldots ,]$ an n-ary linear context.

We first prove FD for simultaneous sets of redexes and then show that in an orthogonal Higher-Order Rewriting System, every set of redexes in a term is simultaneous.

Lemma 6.3 (Finite Developments) *Complete developments of a simultaneous set of redexes in a Higher-Order Rewrite Systems are finite, end in the same term and all have the same descendant relation.*

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

a First, one proves 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*. In a diagram:

$$\begin{tikzcd}
D[L,l] \arrow[rrr, "{D[L,l \to r]}"'] \arrow[dd, two heads, "g"'] \arrow[dr, two heads, "{h[l]}"'] & & & D[L,r] \arrow[dd, two heads, "f"] \arrow[dl, two heads, "{h[r]}"] \\
& C[l] \arrow[r, "{C[l \to r]}"'] \arrow[dl, two heads, "e"] & C[r] \arrow[dr, two heads, "d"'] & \\
M \arrow[rrr] & & & M'
\end{tikzcd}$$

By simultaneity of $\mathcal{U}$, one can construct the extraction g. Then one constructs the linear expansion $h[l] : C[l] \twoheadleftarrow_{\mathcal{SC}} D[L,l]$, and the linear reduction $h[r] : D[L,r] \twoheadrightarrow_{\mathcal{SC}} C[r]$ (note that we define $h[l]$ to be an expansion, while $h[r]$ 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, $\lfloor \underrightarrow{g; D[L,l \to r]; f} \rfloor = \lfloor \underrightarrow{e; C[l \to r]; d} \rfloor$. This follows by some easy calculations.

b Then, one gives a measure on 'abstracted rewrite steps' and shows 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*. More precisely, let $\mathcal{U}'$ be the set of descendants of $\mathcal{U}$ along u. We will construct an extraction $g' : M' \twoheadleftarrow_{\mathcal{SC}} D'[L']$ of $\mathcal{U}'$ from M', which is smaller than g, in the following sense: $(D[R,r], n) \to^{+}_{\mathcal{SC}} \times_{lex} > (D'[R'], n')$, where n, n' are the number of holes in $D[\,]$ and $D'[\,]$. The construction of g' is shown in the following diagram

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

Here $h'[\,]$ is a reduction from $D[\,,r]$ to its $\mathcal{SC}$-normal form $E[\,]$, reducing the $\mathcal{SC}$-redexes created by plugging in the right-hand side r in the context $D[\,,\,]$. Because r might be non-linear (only left-hand sides were required to be linear), $E[\,]$ might be a non-linear context. Now take $D'[\,]$ to be the *linearisation* of $E[\,]$, i.e. a linear context such that the positions of the holes in $E[\,]$ and $D'[\,]$ are the same, hence $E[L] \equiv D'[L']$ for some appropriate L'. By closure of reduction under substitution, the reduction $h'[L]$ can be constructed and by completeness there exists an expansion g' from M' to $E[L] \equiv D'[L']$. One then shows that the expansion g' is an extraction of $\mathcal{U}'$ from M' into $D'[\,]$. 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 $h'[R] : D[R,r] \twoheadrightarrow_{\mathcal{SC}} E[R] \equiv D'[R']$. The extraction g' can only be not smaller than g if $h'[R]$ is an empty reduction, but then $D'[\,] \equiv D'[\,,r]$ which has one hole less than $D'[\,,\,]$.

c Finally, combining the Envelop lemma with the Develop Lemma, one shows that every complete development of $\mathcal{U}$ from M to N can be simulated by a simultaneous extraction of $\mathcal{U}$ from M into some context $D[\,]$, followed by a sequence of replacement steps from $D[L,l]$ to $D[R,r]$, followed by a reduction to N. This is shown in the following diagram:

$$\begin{tikzcd}
D[L,l] \arrow[rrr, "{D[L,l \to r]}"'] \arrow[dd, "g"] \arrow[dr, "{h[l]}"'] & & & D[L,r] \arrow[rrr, two heads, "{D[L \to R,r]}"'] \arrow[dl, "{h[r]h'[L]}"] \arrow[dr] & & & D[R,r] \arrow[dl, "{h'[R]}"] \arrow[dd, "f"] \\
& C[l] \arrow[r, "{C[l \to r]}"'] \arrow[dl, "e"'] & C[r] \arrow[dr, "d"] & & E[L] \arrow[r, two heads, "{E[L \to R]}"'] \arrow[dl, "g'"'] & E[R] \arrow[dr, "f'"] & \\
M \arrow[rrr] & & & M' \arrow[rrr, two heads] & & & N
\end{tikzcd}$$

Every complete $\mathcal{U}$-development ends in the term N, and the descendant relation is the one induced by $g; D[L \to R, l \to r]; f$, that is, the one induced by following the 'outside' of the diagram.

□

Showing that every set of redexes in an orthogonal Higher-Order Rewriting System is simultaneous can be reduced to showing that every pair of redexes is simultaneous by the following lemma.

Lemma 6.4 *A Higher-Order Rewriting System is simultaneous if and only if it is pairwise simultaneous.*

Now one can show that orthogonal Higher-Order Rewriting Systems are pairwise simultaneous by reducing this property further to non-ambiguity, and state the following theorem.

Theorem 6.5 *Every orthogonal Higher-Order Rewriting System is confluent.*

Here, we are interested in proving confluence for weakly orthogonal systems. In such systems distinct redexes are not simultaneous if they are ambiguous. However, instead of parallel simultaneity the following two properties suffice, as was shown above.

a Simultaneity of a set of redexes is preserved by rewriting,

b If a redex u is simultaneous with each redex in $\mathcal{U}$, then $\mathcal{U} \cup \{u\}$ is simultaneous.

The first item follows easily from the proof of the Develop Lemma. The second item follows from a property called *cubicity*.

A Higher-Order Rewriting System is said to be *cubic*, if every triple of pairwise simultaneous redexes is simultaneous.

Lemma 6.6 *Every Higher-Order Rewriting System is cubic.*

Proof. Consider a triple $\mathcal{U} := \{u_1, u_2, u_3\}$ 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 one redex must nest the other, so without loss of generality, there are four cases to consider:
a All three redexes are disjoint,
b u_1 nests u_2, which in turn nests u_3,
c u_1 nests both u_2 and u_3 which are disjoint,
d u_1 nests u_2 and both are disjoint from u_3.
In each of these cases, first performing the $\mathcal{SC}$-steps in e_3, then the ones in e_2 and finally the steps in e_1 gives a simultaneous extraction of $\mathcal{U}$ into some ternary context. □

The next theorem states that every weakly orthogonal Higher-Order Rewriting System is confluent.

Theorem 6.7 *Every weakly orthogonal Higher-Order Rewriting System is confluent.*

Proof. By the preceding lemma, it suffices to prove that cubicity implies that if u is pairwise simultaneous with each redex in $\mathcal{U}$, then $\mathcal{U} \cup \{u\}$ is simultaneous. One proves, by induction on the size of the set $\mathcal{V} := \mathcal{U} \cup \{u\}$ of simultaneous redexes, that there exists a simultaneous extraction of $\mathcal{V}$ from M, using cubicity to ensure that origins of simultaneous redexes are simultaneous again. □

Next we show that weakly orthogonal combinations of left-linear confluent Higher-Order Rewriting Systems (hence of term rewriting systems, Combinatory Reduction Systems and Higher-order Rewrite Systems) are confluent, thereby solving a problem which was raised by the first author in [DJK93, Problem 62].

Theorem 6.8 *Let $\mathcal{H}, \mathcal{I}$ be left-linear confluent Higher-Order Rewriting Systems on the same alphabet having sets of rules $\mathcal{R}$ and $\mathcal{S}$. The union $\mathcal{H} \cup \mathcal{I}$ obtained by taking $\mathcal{R} \cup \mathcal{S}$ as set of rules, is confluent if the rules of $\mathcal{R}$ are weakly ambiguous with respect to those in $\mathcal{S}$.*

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.

$$\begin{tikzcd}
{} \arrow[r, two heads, "\mathcal{I}"'] \arrow[d, two heads, "\mathcal{H}"] & {} \arrow[d, dashed, two heads, "\mathcal{H}"] \\
{} \arrow[r, dashed, two heads, "\mathcal{I}"'] & {}
\end{tikzcd}$$

Using the 'commutativity' variant of the Strip Lemma (cf.[Bar84]), it suffices to show that for any set $\mathcal{U}$ of simultaneous $\mathcal{R}$-redexes we can construct the following diagram.

$$\begin{tikzcd}
M \arrow[r, "v"'] \arrow[d, two heads, "\mathcal{U}"] & P \arrow[d, dashed, two heads, "\mathcal{U}'"] \\
N \arrow[r, dashed, two heads, "\mathcal{V}"'] & Q
\end{tikzcd}$$

where v is an $\mathcal{I}$-step, $\mathcal{U}'$ is a set of simultaneous $\mathcal{R}$-redexes, $M \twoheadrightarrow_{\mathcal{U}} N$ is a complete development of $\mathcal{U}$ and $P \twoheadrightarrow_{\mathcal{U}'} Q$ is a complete development of $\mathcal{U}'$. There are two cases to consider:

a If v is simultaneous with each step in $\mathcal{U}$, one shows by reasoning analogous to the preceding theorem that first extracting v from M by some extraction e gives a set of pairwise simultaneous $\mathcal{R}$-redexes $\{\mathcal{U} \underrightarrow{\lfloor e \rfloor}\}$, which is simultaneous by cubicity and hence the set $\mathcal{U} \cup \{v\}$ is simultaneous. By (the proof of) the Develop Lemma we know that performing the step v preserves simultaneity, so we can take $\mathcal{U}' := \{\mathcal{U} \underrightarrow{\lfloor v \rfloor}\}$.

b If v is not simultaneous with some step $u \in \mathcal{U}$, then we can take $\mathcal{U}' := \{\mathcal{U} \underrightarrow{\lfloor u \rfloor}\}$.

Finally, the diagram can be completed by an application of the Finite Developments theorem. To start the induction in the Strip Lemma, we observe that if $\mathcal{U}$ consists of just one step, it is simultaneous. □

7. A confluence proof à la Tait and Martin-Löf

The proof method we employ is due to Tait and Martin-Löf. It is as follows. First we define a relation $\Rrightarrow$ on Terms such that its transitive closure equals reduction. Then we prove the diamond property for $\Rrightarrow$. That is, we prove that for any terms M, N, P such that $M \Rrightarrow N$ and $M \Rrightarrow P$ a term Q exists, satisfying $N \Rrightarrow Q$ and $P \Rrightarrow Q$. Before embarking on the proof, we first need some auxiliary results concerning substitution.

Substitution. We will use the following results concerning substitution.

An *elco* is a context $E[,\ldots,]$ consisting of symbols of the substitution calculus and holes. If we are concerned with the replacement of one particular hole by a term M, and the occurrences of the other holes have already been replaced by terms, then we write $E[\,]$ and say that $E[\,]$ is an elco for M.

Proposition 7.1 *Let M_0 and M_1 be terms with $M_0 \twoheadrightarrow M_0'$ and $M_1 \twoheadrightarrow M_1'$. If $M_0M_1 = E[P_1,\ldots,P_n]$, for some elco, then $M_0'M_1' = E[P_1',\ldots,P_n']$ with $P_1 \twoheadrightarrow P_1',\ldots,P_n \twoheadrightarrow P_n'$.*

Proposition 7.2 *Let $E[\,]$ be an elco for a term M. Suppose $E[\,] \twoheadrightarrow E'[\,]$ and $M \twoheadrightarrow M'$. Then $E[M]{\downarrow}_{\mathcal{SC}} \twoheadrightarrow E'[M']{\downarrow}_{\mathcal{SC}}$.*

Proof. The proof proceeds by induction on the length of the maximal reduction of $E[M]$ to $\mathcal{SC}$-normal form.

base step. In the base step, the reduction of $E[M]$ to $\mathcal{SC}$-normal form takes zero steps. We prove by induction on the length of the rewrite sequence $M \twoheadrightarrow M'$ that $E[M] \twoheadrightarrow E'[M']$. If $M = M'$, then $E[M] \twoheadrightarrow E'[M]$ follows by induction on the length of the rewrite sequence $E[\,] \twoheadrightarrow E'[\,]$. That is, we prove that $E[M] \twoheadrightarrow E'[M]$ if $E[\,] \rightarrow E'[\,]$. If $M \rightarrow N \twoheadrightarrow M'$, then we prove $E[M] \twoheadrightarrow E[N]$. By induction hypothesis of the induction on the length of the rewrite sequence $M \twoheadrightarrow M'$, we have $E[N] \twoheadrightarrow E'[M']$. Together, we have $E[M] \twoheadrightarrow E[N] \twoheadrightarrow E'[M']$. The statement $E[M] \twoheadrightarrow E[N]$ is proven by induction on the structure of $E[\,]$.

induction step. In the induction step, we suppose that the reduction of $E[M]$ to $\mathcal{SC}$-normal form takes more than zero steps. The induction step is proven by induction on the length of the rewrite sequence $M \twoheadrightarrow M'$. If $M = M'$, then we prove by induction on the length of $E[\ ] \twoheadrightarrow E'[\ ]$ that $E[M]\downarrow_{\mathcal{SC}} \twoheadrightarrow E'[M]\downarrow_{\mathcal{SC}}$. That is, we prove that $E[M]\downarrow_{\mathcal{SC}} \twoheadrightarrow E'[M]\downarrow_{\mathcal{SC}}$ if $E[\ ] \to E'[\ ]$. If $M \to N \twoheadrightarrow M'$, then we prove $E[M]\downarrow_{\mathcal{SC}} \twoheadrightarrow E[N]\downarrow_{\mathcal{SC}}$. By induction hypothesis of the induction on the length of the rewrite sequence $M \twoheadrightarrow M'$, we have $E[N]\downarrow_{\mathcal{SC}} \twoheadrightarrow E'[M']\downarrow_{\mathcal{SC}}$. Together, we have $E[M]\downarrow_{\mathcal{SC}} \twoheadrightarrow E[N]\downarrow_{\mathcal{SC}} \twoheadrightarrow E'[M']\downarrow_{\mathcal{SC}}$. We prove $E[M]\downarrow_{\mathcal{SC}} \twoheadrightarrow E[N]\downarrow_{\mathcal{SC}}$ by induction on the structure of $E[\,]$.

- If $E[\,]$ is a hole then it is clear.
- If $E[\,] = x.E_0[\,]$, then the statement follows from the induction hypothesis of the induction on the structure of $E[\,]$.
- If $E[\,]$ is an application we suppose without loss of generality that the hole occurs in the left part, i.e. $E[\,] = E_0[\,]E_1$. By induction hypothesis, we have $E_0[M]\downarrow_{\mathcal{SC}} \twoheadrightarrow E_0[N]\downarrow_{\mathcal{SC}}$. If $E_0[M]\downarrow_{\mathcal{SC}}E_1$ is a term, we are done. Otherwise, it is of the form $E^*[P_1, \ldots, P_n]$. By Proposition 7.1 we know that $E_0[N]\downarrow_{\mathcal{SC}}E_1$ is of the form $E^*[P'_1, \ldots, P'_n]$ with $P_1 \twoheadrightarrow P'_1, \ldots, P_n \twoheadrightarrow P'_n$. By induction hypothesis of the induction on the length of the maximal reduction of $E[M]$ to $\mathcal{SC}$-normal form, we have $E^*[P_1, \ldots, P_n] \twoheadrightarrow E^*[P'_1, \ldots, P'_n]$. This yields $(E_0[M]E_1)\downarrow_{\mathcal{SC}} \twoheadrightarrow (E_0[N]E_1)\downarrow_{\mathcal{SC}}$.

□

Corollary 7.3 *Let M be a term with $M \twoheadrightarrow M'$. Let $C[\ ]$ be a context with $C[\ ] \twoheadrightarrow C'[\ ]$. Then $C[M]\downarrow_{\mathcal{SC}} \twoheadrightarrow C'[M']$.*

Proof. By induction on the structure of $C[\,]$, the base case being the previous proposition. □

The Proof. Now we can give the proof of confluence of weakly orthogonal Higher-Order Rewriting Systems.

First the definition of $\Rightarrow$ is given.

Definition 7.4 A relation $\Rightarrow$ on Terms is defined as follows:
(1) $x \Rightarrow x$ for every variable $x \in \mathsf{Var}$,
(2) $a \Rightarrow a$ for every operator $a \in \mathcal{O}$,
(3) if $M \Rightarrow M'$ then $x.M \Rightarrow x.M'$,
(4) if $M_0 \Rightarrow M'_0$ and $M_1 \Rightarrow M'_1$, then $M_0M_1 \Rightarrow M'_0M'_1$,
(5) if $l \to r$ is a rewrite rule and $E[\,]$ is an elco for l such that $E[\,] \Rightarrow E'[\,]$, then $E[l]\downarrow_{\mathcal{SC}} \Rightarrow E'[r]\downarrow_{\mathcal{SC}}$.

The first step of the confluence proof is easy.

Proposition 7.5 *The transitive closure of $\Rightarrow$ equals reduction.*

Proof. Suppose $M \to M'$ by some rewrite rule $l \to r$. Then $M\ {}_{\mathcal{SC}}\!\twoheadleftarrow C[l]$ and $C[r] \twoheadrightarrow_{\mathcal{SC}} M'$ for a context $C[\,]$. We prove by induction on the structure of $C[\,]$ that $C[l]\downarrow_{\mathcal{SC}} \Rightarrow C[r]\downarrow_{\mathcal{SC}}$. If $C[\,]$ is an elco for l, then $C[l]\downarrow_{\mathcal{SC}} \Rightarrow C[r]\downarrow_{\mathcal{SC}}$, since by reflexivity of $\Rightarrow$ we have $C[\,] \Rightarrow C[\,]$. The other cases follow from context-compatibility of $\Rightarrow$.

On the other hand, suppose $M \Rightarrow M'$. We prove $M \twoheadrightarrow M'$ by induction on the derivation of $M \Rightarrow M'$. If $M \Rightarrow M'$ is not due to the last clause of the definition of $\Rightarrow$, then it is obvious. If $M \Rightarrow M'$ is in fact due to the last clause of the definition of $\Rightarrow$, then $M = C[l]\downarrow_{\mathcal{SC}}$ and $C'[r]\downarrow_{\mathcal{SC}} = M'$ with $C[\,] \Rightarrow C'[\,]$. By induction hypothesis, $C[\,] \twoheadrightarrow C'[\,]$. By Proposition 7.2 we have $M = C[l]\downarrow_{\mathcal{SC}} \twoheadrightarrow C'[r]\downarrow_{\mathcal{SC}} = M'$. □

For the proof of the diamond property we need a result concerning the interaction between substitution and parallel rewriting, and a Coherence Lemma.

PROPOSITION 7.6 *Let M_0 and M_1 be terms with $M_0 \Rightarrow M_0'$ and $M_1 \Rightarrow M_1'$. If $M_0M_1 = E[P_1, \ldots, P_n]$ then $M_0'M_1' = E[P_1', \ldots, P_n']$ with $P_1 \Rightarrow P_1', \ldots, P_n \Rightarrow P_n'$.*

PROPOSITION 7.7 *Let $E[\,]$ be an elco for M. Suppose $E[\,] \Rightarrow E'[\,]$ and $M \Rightarrow M'$. Then $E[M]{\downarrow_{\mathcal{SC}}} \Rightarrow E'[M']{\downarrow_{\mathcal{SC}}}$.*

PROOF. The proof proceeds by induction on the maximal length of the reduction of $E[M]$ to $\mathcal{SC}$-normal form.

base step. In the base case, the reduction of $E[M]$ to $\mathcal{SC}$-normal form takes zero steps. The statement is proven by induction on the derivation of $E[\,] \Rightarrow E'[\,]$.

induction step. Consider for the induction step that the reduction of $E[M]$ to $\mathcal{SC}$-normal takes more than zero steps. The proof of the induction step proceeds by induction on the derivation of $E[\,] \Rightarrow E'[\,]$.

(1) If $E[\,] \Rightarrow E'[\,]$ is $x \Rightarrow x$ for a variable $x \in \mathsf{Var}$, then it is trivial.

(2) If $E[\,] \Rightarrow E'[\,]$ is $a \Rightarrow a$ for an operator $a \in \mathcal{O}$, then it is also trivial.

(3) If $E[\,] \Rightarrow E'[\,]$ is $x.E_0[\,] \Rightarrow x.E_0'[\,]$ with $E_0[\,] \Rightarrow E_0'[\,]$, then the statement follows from the induction hypothesis of the induction on the derivation of $E[\,] \Rightarrow E'[\,]$.

(4) Suppose $E[\,] \Rightarrow E'[\,]$ is due to the fourth clause of the definition of $\Rightarrow$. Without loss of generality we assume that the hole in $E[\,]$ occurs in the left part of the application, so $E[\,] = E_0[\,]E_1$. By induction hypothesis of the induction on the derivation of $E[\,] \Rightarrow E'[\,]$, we have $E_0[M]{\downarrow_{\mathcal{SC}}} \Rightarrow E_0'[M]{\downarrow_{\mathcal{SC}}}$ and $E_1 \Rightarrow E_1'$.

If $E_0[M]{\downarrow_{\mathcal{SC}}}E_1$ is a term, then we are done.

If $E_0[M]{\downarrow_{\mathcal{SC}}}E_1$ is not a term, then it a redex for the substitution calculus of the form $E[P_1, \ldots, P_n]$. By Proposition 7.6 we know that $E_0'[M']{\downarrow_{\mathcal{SC}}}E_1' = E[P_1', \ldots, P_n']$ with $P_1 \Rightarrow P_1', \ldots, P_n \Rightarrow P_n'$. The reduction of $E[P_1, \ldots, P_n]$ to $\mathcal{SC}$-normal form takes less steps than the one of $E_0[M]E_1$ to $\mathcal{SC}$-normal form. By induction hypothesis of the induction on the length of the maximal reduction to $\mathcal{SC}$-normal form, we have $E[P_1, \ldots, P_n]{\downarrow_{\mathcal{SC}}} \Rightarrow E[P_1', \ldots, P_n']{\downarrow_{\mathcal{SC}}}$. This yields $(E_0[M]E_1){\downarrow_{\mathcal{SC}}} \Rightarrow (E_0'[M]E_1'){\downarrow_{\mathcal{SC}}}$.

(5) Suppose $E[\,] \Rightarrow E'[\,]$ is due to the last clause of the definition of $\Rightarrow$. Then $E[\,] = C[l]{\downarrow_{\mathcal{SC}}}$ and $E'[\,] = C'[r]{\downarrow_{\mathcal{SC}}}$ for a rewrite rule $l \to r$ and an elco $C[\,]$ for l with $C[\,] \Rightarrow C'[\,]$. $C[\,]$ is of the form $D[\Box_1, \Box_2]$ where $\Box_1$ is to be replaced by M and $\Box_2$ by l, and $C'[\,]$ is of the form $D'[\Box_1, \Box_2]$ with $D[\Box_1, \Box_2] \Rightarrow D'[\Box_1, \Box_2]$. By induction hypothesis of the induction on the derivation of $E[\,] \Rightarrow E'[\,]$, we have $D[M, \Box_2]{\downarrow_{\mathcal{SC}}} \Rightarrow D'[M', \Box_2]{\downarrow_{\mathcal{SC}}}$. This yields

$$\begin{array}{rcl} E[M]{\downarrow_{\mathcal{SC}}} & = & D[M, l]{\downarrow_{\mathcal{SC}}} \\ & \Rightarrow & D'[M', r]{\downarrow_{\mathcal{SC}}} \\ & = & E'[M']{\downarrow_{\mathcal{SC}}} \end{array}$$

□

COROLLARY 7.8 *Let M be a closed term with $M \Rightarrow M'$. Let $C[\,]$ be a context with $C[\,] \Rightarrow C'[\,]$. Then $C[M] \Rightarrow C'[M']$.*

Proof. The proof proceeds by induction on the structure of $C[\ ]$. The base case is the previous proposition.

Lemma 7.9 (Coherence Lemma) *Let $l \to r$ be a rewrite rule. Let $M = M_0M_1 = E[l]{\downarrow_{\mathcal{SC}}}$ for an elco $E[\,]$ for l. Suppose $M_0 \Rightarrow N_0$ and $M_1 \Rightarrow N_1$. Suppose in $M_0 \Rightarrow N_0$ or in $M_1 \Rightarrow N_1$ a redex that is critical for $l \to r$ is contracted. Then $M' = E[r]{\downarrow_{\mathcal{SC}}} \Rightarrow M_0M_1$.*

$$\begin{tikzcd}
M = M_0M_1 \arrow[r] \arrow[d, Rightarrow] & M' \arrow[dl, Rightarrow] \\
N_0N_1 &
\end{tikzcd}$$

Proof. If M contains two disjoint redexes that are critical for $l \to r$, then by weak orthogonality $M = M'$. Then the statement trivially holds.

So suppose all redexes in M that are critical for $l \to r$ are nested. Suppose they all occur in M_0. Let the smallest (with respect to the subterm-relation) redex that is critical for $l \to r$ and that is contracted in $M_0 \Rightarrow N_0$ be an instance of $g \to d$. So for some context $C[\,]$ we have $M_0 = C[g]{\downarrow_{\mathcal{SC}}}$. By weak orthogonality, we have $M' = E[r]{\downarrow_{\mathcal{SC}}} = C[d]{\downarrow_{\mathcal{SC}}}M_1$. By hypothesis, we have $M_1 \Rightarrow N_1$. We prove $C[d]{\downarrow_{\mathcal{SC}}} \Rightarrow N_1$, then the statement follows by application of the fourth clause of the definition of $\Rightarrow$. By induction on the structure of $C[\ ]$ one proves the following: if $C[g]{\downarrow_{\mathcal{SC}}} \Rightarrow N_0$ and if in this derivation an instance of g is contracted, then $C[d]{\downarrow_{\mathcal{SC}}} \Rightarrow N_0$. $\Box$

Theorem 7.10 *The relation $\Rightarrow$ satisfies the diamond property.*

Proof. Suppose $M \Rightarrow N$ and $M \Rightarrow P$. We prove a Q exists with $N \Rightarrow Q$ and $P \Rightarrow Q$ by either considering 'easier' derivations of $M \Rightarrow N$ or of $M \Rightarrow P$, where 'easier' means that there are less applications of the last clause of the definition of $\Rightarrow$, or by considering subderivations of $M \Rightarrow N$ and of $M \Rightarrow P$. Let $\mathsf{C}(M \Rightarrow N)$ be the number of applications of the last clause of the definition of $\Rightarrow$ in the derivation $M \Rightarrow N$. Let $\mathsf{L}(M \Rightarrow N)$ be the length of the derivation of $M \Rightarrow N$. The proof proceeds by induction on $(\mathsf{C}(M \Rightarrow N) + \mathsf{C}(M \Rightarrow P), \mathsf{L}(M \Rightarrow N) + \mathsf{L}(M \Rightarrow P))$, lexicographically ordered. We call $\mathsf{C}(M \Rightarrow N) + \mathsf{C}(M \Rightarrow P)$ the complexity of the diversion $M \Rightarrow N$ and $M \Rightarrow P$.

(1) If $M \Rightarrow N$ is $x \Rightarrow x$ for some $x \in \mathsf{Var}$, then $P = x$. Define $Q := x$.

(2) If $M \Rightarrow N$ is $a \Rightarrow a$ for some $a \in \mathcal{O}$, then define $Q := P$.

(3) If $M \Rightarrow N$ is $x.M_0 \Rightarrow x.N_0$ with $M_0 \Rightarrow N_0$, then P is of the form $x.P_0$ with $M_0 \Rightarrow P_0$. By induction hypothesis a Q_0 exists satisfying $N_0 \Rightarrow Q_0$ and $P_0 \Rightarrow Q_0$. Define $Q := x.Q_0$.

(4) If $M \Rightarrow N$ is $M_0M_1 \Rightarrow N_0N_1$ with $M_0 \Rightarrow N_0$ and $M_1 \Rightarrow N_1$, then there are two possibilities for the last step of the derivation of $M \Rightarrow P$.

If $M \Rightarrow P$ is $M_0M_1 \Rightarrow P_0P_1$, then by induction hypothesis Q_0 and Q_1 exist with $N_0 \Rightarrow Q_0$, $P_0 \Rightarrow Q_0$, $N_1 \Rightarrow Q_1$ and $P_1 \Rightarrow Q_1$. Define $Q := Q_0Q_1$.

If $M \Rightarrow P$ is due to the last clause of the definition of $\Rightarrow$, then $M = C[l]{\downarrow_{\mathcal{SC}}}$ and $P = C'[r]{\downarrow_{\mathcal{SC}}}$ for some rewrite rule $l \to r$ and an elco $C[\,]$ for l with $C[\,] \Rightarrow C'[\,]$.

- If in $M_0 \Rightarrow N_0$ nor in $M_1 \Rightarrow N_1$ a redex that is critical for $l \to r$ is contracted, then N_0N_1 is of the form $C''[l]{\downarrow_{\mathcal{SC}}}$ for some elco $C''[\,]$ for l with $C[\,] \Rightarrow C''[\,]$. By induction hypothesis, an elco $D[\ ]$ for l exists with $C'[\ ] \Rightarrow D[\ ]$ and $C''[\ ] \Rightarrow D[\ ]$. Define

$Q = D[r]{\downarrow}_{\mathcal{SC}}$. Then we have the following:

$$\begin{tikzcd}
M = C[l]{\downarrow}_{\mathcal{SC}} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & C'[r]{\downarrow}_{\mathcal{SC}} = P \arrow[d, Rightarrow] \\
N = C''[l]{\downarrow}_{\mathcal{SC}} \arrow[r, Rightarrow] & D[r]{\downarrow}_{\mathcal{SC}} = Q
\end{tikzcd}$$

- If in $M_0 \Rightarrow N_0$ or in $M_1 \Rightarrow N_1$ a redex critical for $l \to r$ is contracted, then we distinguish two possibilities.

  If M contains two disjoint redexes that are critical for $l \to r$ then by weak orthogonality $C[l]{\downarrow}_{\mathcal{SC}} = C[r]{\downarrow}_{\mathcal{SC}}$. The complexity of the diversion $C[r]{\downarrow}_{\mathcal{SC}} \Rightarrow N$ and $C[r]{\downarrow}_{\mathcal{SC}} \Rightarrow C'[r]{\downarrow}_{\mathcal{SC}}$ is less than the one of the diversion $C[l]{\downarrow}_{\mathcal{SC}} \Rightarrow N$ and $C[l]{\downarrow}_{\mathcal{SC}} \Rightarrow C'[r]{\downarrow}_{\mathcal{SC}}$. So by induction hypothesis, a Q exists with $N \Rightarrow Q$ and $P = C'[r]{\downarrow}_{\mathcal{SC}} \Rightarrow Q$. So we have

$$\begin{tikzcd}
M = C[l]{\downarrow}_{\mathcal{SC}} \arrow[rr, Rightarrow] \arrow[dd, Rightarrow] \arrow[dr, equal] & & C'[r]{\downarrow}_{\mathcal{SC}} = P \arrow[dd, Rightarrow] \\
& C[r]{\downarrow}_{\mathcal{SC}} \arrow[ur, Rightarrow] \arrow[dl, Rightarrow] & \\
N = N_0N_1 \arrow[rr, Rightarrow] & & Q
\end{tikzcd}$$

  So suppose all redexes in M that are critical for $l \to r$ are nested and suppose at least one of them is contracted in $M_0 \Rightarrow N_0$. Suppose the largest redex that is contracted in $M_0 \Rightarrow N_0$ and that is critical for $l \to r$ is an instance of $g \to d$. So $M_0 = D_0[g]{\downarrow}_{\mathcal{SC}}$ with $D_0[\,]$ a context with an elco for g as subcontext, and N is of the form $D_0'[d]{\downarrow}_{\mathcal{SC}}N_1$ with $D_0[\,] \Rightarrow D_0'[\,]$. Let $M' = C'[l]{\downarrow}_{\mathcal{SC}}$. Note that M' is of the form $M_0'M_1'$.

  If in $M = C[l]{\downarrow}_{\mathcal{SC}} \Rightarrow C'[l]{\downarrow}_{\mathcal{SC}} = M' = M_0'M_1'$ no redex critical for $g \to d$ is contracted, then M_0' is of the form $D_0''[g]{\downarrow}_{\mathcal{SC}}$ with $D_0[\,] \Rightarrow D_0''[\,]$. By induction hypothesis a context $E_0[\,]$ exists with $D_0'[\,] \Rightarrow E_0[\,]$ and $D_0''[\,] \Rightarrow E_0[\,]$. Also by induction hypothesis a Q_1 exists with $M_1' \Rightarrow Q$ and $N_1 \Rightarrow Q_1$. Define $Q_0 = E_0[d]{\downarrow}_{\mathcal{SC}}$ and let $Q = Q_0Q_1$. Now we have $M' = M_0'M_1' \Rightarrow E_0[d]{\downarrow}_{\mathcal{SC}}Q_1$ with $M_0' \Rightarrow E_0[d]{\downarrow}_{\mathcal{SC}}$ and $M_1' \Rightarrow Q_1$. Further, $M' = C'[l]{\downarrow}_{\mathcal{SC}} \to C'[r]{\downarrow}_{\mathcal{SC}} = N$. In $M_0' \Rightarrow E_0[d]{\downarrow}_{\mathcal{SC}}$ a redex that is critical in $l \to r$ has been contracted, namely an instance of $g \to d$. Therefore, by the Coherence Lemma (Lemma 7.9), we have $C'[r]{\downarrow}_{\mathcal{SC}} \Rightarrow E_0[d]{\downarrow}_{\mathcal{SC}}Q_1$. That is, we have $N \Rightarrow Q$ and $P \Rightarrow Q$. In a picture:

$$\begin{tikzcd}
C[l]{\downarrow}_{\mathcal{SC}} = D_0[g]{\downarrow}_{\mathcal{SC}}M_1 \arrow[rr, Rightarrow] \arrow[dd, Rightarrow] \arrow[dr, Rightarrow] & & C'[r]{\downarrow}_{\mathcal{SC}} = P \arrow[dd, Rightarrow] \\
& D_0''[g]{\downarrow}_{\mathcal{SC}}M_1' = C'[l]{\downarrow}_{\mathcal{SC}} \arrow[ur] \arrow[dr, Rightarrow] & \\
N = D_0'[d]{\downarrow}_{\mathcal{SC}}N_1 \arrow[rr, Rightarrow] & & E_0[d]{\downarrow}_{\mathcal{SC}}Q_1
\end{tikzcd}$$

  If in $C[l]{\downarrow}_{\mathcal{SC}} \Rightarrow C'[l]{\downarrow}_{\mathcal{SC}}$ a redex critical for $g \to d$ is contracted, then we consider two possibilities.

  If there are two disjoint redexes in M that are critical for $g \to d$, then we have $D_0[g]{\downarrow}_{\mathcal{SC}} = D_0[d]{\downarrow}_{\mathcal{SC}}$, so $M = D_0[d]{\downarrow}_{\mathcal{SC}}M_1$. We have $D_0[d]{\downarrow}_{\mathcal{SC}}M_1 \Rightarrow D_0'[d]{\downarrow}_{\mathcal{SC}}N_1 = N$ and $D_0[d]{\downarrow}_{\mathcal{SC}}M_1 \Rightarrow P$. The complexity of that diversion is strictly less than that of

$M = D_0[g]{\downarrow_{\mathcal{SC}}}M_1 \Rightarrow N$ and $M = D_0[g]{\downarrow_{\mathcal{SC}}}M_1 \Rightarrow P$. By induction hypothesis, a Q exists with $N \Rightarrow Q$ and $P \Rightarrow Q$. In a picture:

$$\begin{tikzcd}
M = C[l]{\downarrow_{\mathcal{SC}}} \arrow[rr, Rightarrow] \arrow[dd, Rightarrow] \arrow[dr, phantom, sloped, "\equiv"] & & C'[r]{\downarrow_{\mathcal{SC}}} = P \arrow[dd, Rightarrow] \\
& D_0[d]{\downarrow_{\mathcal{SC}}}M_1 \arrow[ur, Rightarrow] \arrow[dl, Rightarrow] & \\
N = D_0'[d]{\downarrow_{\mathcal{SC}}}N_1 \arrow[rr, Rightarrow] & & Q
\end{tikzcd}$$

Suppose next that all redexes in $C[l]{\downarrow_{\mathcal{SC}}}$ that are critical for $g \to d$ are nested and suppose that at least one of them is contracted in $C[l]{\downarrow_{\mathcal{SC}}} \Rightarrow C'[l]{\downarrow_{\mathcal{SC}}}$. Let the largest one of them be an instance of $g' \to d'$. So $M_0 = E_0[g']{\downarrow_{\mathcal{SC}}}$ for some context $E_0[\,]$ having an elco for g' as subcontext. This instance of $g' \to d'$ is not critical for $l \to r$. So there exists an elco $C^*[\,]$ for l with $C[\,] \Rightarrow C^*[\,] \Rightarrow C'[\,]$ such that $C^*[l]{\downarrow_{\mathcal{SC}}} = E_0[d']{\downarrow_{\mathcal{SC}}}M_1$. By weak orthogonality, $E_0[d']{\downarrow_{\mathcal{SC}}} = D_0[d]{\downarrow_{\mathcal{SC}}}$. Let $M' = C^*[l]{\downarrow_{\mathcal{SC}}} = E_0[d']{\downarrow_{\mathcal{SC}}}M_1$. We have $M' = D_0[d]{\downarrow_{\mathcal{SC}}}M_1 \Rightarrow D_0'[d]{\downarrow_{\mathcal{SC}}}N_1 = N$ and $M' = C^*[l]{\downarrow_{\mathcal{SC}}} \Rightarrow C'[r]{\downarrow_{\mathcal{SC}}} = P$. The complexity of this derivation is strictly less than the one of $M = D_0[g]{\downarrow_{\mathcal{SC}}}M_1 \Rightarrow D_0'[d]{\downarrow_{\mathcal{SC}}}N_1 = N$ and $M = C[l]{\downarrow_{\mathcal{SC}}} \Rightarrow C'[l]{\downarrow_{\mathcal{SC}}} = P$. So by induction hypothesis, a Q exists with $N \Rightarrow Q$ and $P \Rightarrow Q$. We have

$$\begin{tikzcd}
E_0[d']{\downarrow_{\mathcal{SC}}}M_1 = C^*[l]{\downarrow_{\mathcal{SC}}} \arrow[rr, Rightarrow] \arrow[dd, Rightarrow] \arrow[dr, phantom, sloped, "\equiv"] & & C'[r]{\downarrow_{\mathcal{SC}}} = P \arrow[dd, Rightarrow] \\
& D_0[d]{\downarrow_{\mathcal{SC}}}M_1 \arrow[ur, Rightarrow] \arrow[dl, Rightarrow] & \\
N = D_0'[d]{\downarrow_{\mathcal{SC}}}N_1 \arrow[rr, Rightarrow] & & Q
\end{tikzcd}$$

(5) Suppose $M \Rightarrow N$ is due to the last clause of the definition of $\Rightarrow$. Then $M = C[l]{\downarrow_{\mathcal{SC}}}$ and $N = C'[r]{\downarrow_{\mathcal{SC}}}$ for some rewrite rule $l \to r$ and an elco $C[\,]$ for l.

If $M \Rightarrow P$ is $M_0M_1 \Rightarrow P_0P_1$ with $M_0 \Rightarrow P_0$ and $M_1 \Rightarrow P_1$ then we proceed similarly to the previous case.

So suppose $M \Rightarrow P$ is also due to the last clause of the definition of $\Rightarrow$. Then $M = D[g]{\downarrow_{\mathcal{SC}}}$ and $P = D'[d]{\downarrow_{\mathcal{SC}}}$ for a rewrite rule $g \to d$ and an elco $D[\,]$ for g. If in $C[l]{\downarrow_{\mathcal{SC}}} \Rightarrow C'[l]{\downarrow_{\mathcal{SC}}}$ no redex critical for $g \to d$ is contracted, then $C'[l]{\downarrow_{\mathcal{SC}}} = D''[g]{\downarrow_{\mathcal{SC}}}$ for some elco $D''[\,]$ for g with $D[\,] \Rightarrow D''[\,]$. By weak orthogonality, we have $C'[r]{\downarrow_{\mathcal{SC}}} = D''[d]{\downarrow_{\mathcal{SC}}}$. By induction hypothesis, an elco $E[\,]$ for g exists with $D'[\,] \Rightarrow E[\,]$ and $D''[\,] \Rightarrow E[\,]$. We have $N = C'[r]{\downarrow_{\mathcal{SC}}} = D''[d]{\downarrow_{\mathcal{SC}}} \Rightarrow E[d]{\downarrow_{\mathcal{SC}}}$ and $P = D'[d]{\downarrow_{\mathcal{SC}}} \Rightarrow E[d]{\downarrow_{\mathcal{SC}}}$. So take $Q := E[d]{\downarrow_{\mathcal{SC}}}$. In a picture:

$$\begin{tikzcd}
C[l]{\downarrow_{\mathcal{SC}}} = D[g]{\downarrow_{\mathcal{SC}}} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & D'[d]{\downarrow_{\mathcal{SC}}} = P \arrow[d, Rightarrow] \\
N = C'[r]{\downarrow_{\mathcal{SC}}} = D''[d]{\downarrow_{\mathcal{SC}}} \arrow[r, Rightarrow] & E[d]{\downarrow_{\mathcal{SC}}} = Q
\end{tikzcd}$$

The case that in $D[g]\!\downarrow_{\mathcal{SC}} \Rightarrow D'[g]\!\downarrow_{\mathcal{SC}}$ no redex critical for $l \to r$ is contracted is similar.

Finally, suppose in $C[l]\!\downarrow_{\mathcal{SC}} \Rightarrow C'[l]\!\downarrow_{\mathcal{SC}}$ a redex critical for $g \to d$ is contracted and in $D[g]\!\downarrow_{\mathcal{SC}} \Rightarrow D'[g]\!\downarrow_{\mathcal{SC}}$ a redex critical for $l \to r$ is contracted. Then by weak orthogonality $C[r]\!\downarrow_{\mathcal{SC}} = C[l]\!\downarrow_{\mathcal{SC}} = D[g]\!\downarrow_{\mathcal{SC}} = D[d]\!\downarrow_{\mathcal{SC}}$. We have $M = D[d]\!\downarrow_{\mathcal{SC}} \Rightarrow D'[d]\!\downarrow_{\mathcal{SC}} = N$ and $M = C[r]\!\downarrow_{\mathcal{SC}} \Rightarrow C'[r]\!\downarrow_{\mathcal{SC}} = P$. The complexity of this diversion is strictly less than the one of $M = D[g]\!\downarrow_{\mathcal{SC}} \Rightarrow D'[d]\!\downarrow_{\mathcal{SC}}$ and $M = C[l]\!\downarrow_{\mathcal{SC}} \Rightarrow C'[r]\!\downarrow_{\mathcal{SC}}$. By induction hypothesis, a Q exists with $C'[r]\!\downarrow_{\mathcal{SC}} = N \Rightarrow Q$ and $D'[d]\!\downarrow_{\mathcal{SC}} = P \Rightarrow Q$.

$$\begin{tikzcd}
C[l]\!\downarrow_{\mathcal{SC}} = D[g]\!\downarrow_{\mathcal{SC}} \arrow[rr, Rightarrow] \arrow[dd, Rightarrow] \arrow[dr, equal] & & D'[d] \arrow[dd, Rightarrow] \\
& C[r]\!\downarrow_{\mathcal{SC}} = D[d]\!\downarrow_{\mathcal{SC}} \arrow[ur, Rightarrow] \arrow[dl, Rightarrow] & \\
C'[r]\!\downarrow_{\mathcal{SC}} \arrow[rr, Rightarrow] & & Q
\end{tikzcd}$$

□

Corollary 7.11 *All weakly orthogonal Higher-Order Rewriting Systems are confluent.*

8. Conclusion

In this paper we have presented a general confluence by (weak) orthogonality result for the class of higher-order term rewriting systems. This result generalises known results for special classes of rewriting systems such as TRSs, CRSs and HRSs, via a uniform presentation preserving their common features and parametrising over their differences. The uniform presentation is based on the analogy

$$\textit{rewriting} \quad = \quad \textit{substitution} + \textit{rules}$$

or more tentatively,

$$\textit{rewriting} \quad = \quad \textit{logic} + \textit{rules}$$

Accordingly, one can classify properties of rewriting systems into *logical* properties, which depend on the logic, and *rewrite* properties which depend on the actual rewrite rules. Then, (weak) orthogonality can be viewed as a sufficient condition on the rewrite rules allowing to reduce the rewrite property of confluence to a logical property.

Since, in this paper, we aimed at the development of theory for *term* rewriting systems we have restricted attention to a formalisation of the proofs of the logic as (λ)-terms. Moreover, we have restricted attention to the propositional intuitionistic logic of application.

In future work we will consider rewriting systems having a graphical notation for substitution (the proofs of the logic): i.e. graph rewriting systems. As a first problem we set out to investigate the rewrite property of *optimality* of rewriting as defined by Lévy [Lév78]. Although optimal implementations using graph rewriting do exist both for the lambda calculus ([Lam90, Kat90]) and for the more general class of Interaction Systems ([AL92]), we think our approach can shed new light on the subject matter. In this light, the work so far can be characterised as stating conditions on the form of the rewrite rules allowing to reduce optimality from a rewrite property to a logical property, much in the same way as orthogonality can be viewed as a sufficient condition on the rewrite rules allowing to reduce the rewrite property of confluence to a logical property.

9. Acknowledgements

We are grateful for inspiring discussions with Zurab Khasidashvili, Jan Willem Klop, Aart Middeldorp and Fer-Jan de Vries.

References

[Acz78] Peter Aczel. A general Church-Rosser theorem. Technical report, University of Manchester, July 1978.

[AL92] A. Asperti and C. Laneve. Interaction systems I: the theory of optimal reductions. Technical Report 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*. North-Holland Publishing Company, revised edition, 1984. (Second printing 1985).

[CR36] Alonzo Church and J.B. Rosser. Some properties of conversion. *Transactions of the American Mathematical Society*, 39:472–482, 1936.

[DJ89] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Formal Methods and Semantics, Handbook of Theoretical Computer Science, Volume B*, chapter 6, pages 243–320. MIT Press, 1989.

[DJK93] Nachum Dershowitz, Jean-Pierre Jouannaud, and Jan Willem Klop. More problems in rewriting. Pp. 468–487 of LNCS 690, Proceedings of 5th RTA, 1993.

[GLM92] Georges Gonthier, Jean-Jacques Lévy, and Paul-André Melliès. An abstract standardisation theorem. In *Proceedings of 7th LICS*, pages 72–81, Santa Cruz, California, 1992. IEEE Computer Society Press.

[HL91] Gérard Huet and Jean-Jacques Lévy. Computations in orthogonal rewriting systems, I. Ch. 11 of Computational Logic: Essays in Honor of Alan Robinson, 1991.

[Kat90] V. Kathail. *Optimal Interpreters for lambda-calculus based functional languages.* PhD thesis, MIT, 1990.

[Kha90] Z.O. Khasidashvili. Expression reduction systems. In *Proceedings of I. Vekua Institute of Applied Mathematics*, volume 36, pages 200–220, Tbilisi, 1990.

[Kha94] Zurab Khasidashvili. The longest perpetual reductions in orthogonal expression reduction systems. In *Proceedings of the third International Symposium on Logical Foundations of Computer Science*, pages 191–203, 1994.

[Klo80] J.W. Klop. *Combinatory Reduction Systems.* Mathematical Centre Tracts Nr. 127. CWI, Amsterdam, 1980. PhD Thesis.

[Klo92] J.W. Klop. Term rewriting systems. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science, Volume II.* Oxford University Press, 1992.

[KOR93] J.W. Klop, V. van Oostrom, and F. van Raamsdonk. Combinatory reduction systems, introduction and survey. *Theoretical Computer Science*, 121:279–308, 1993.

[Lam90] John Lamping. An algorithm for optimal lambda calculus reduction. In *Proceedings of the 17th ACM Conference on Principles of Programming Languages*, pages 16–30, 1990.

[Lév78] Jean-Jacques Lévy. *Réductions correctes et optimales dans le lambda-calcul.* PhD thesis, Université de Paris VII, 1978.

[Nip91] T. Nipkow. Higher-order critical pairs. In *Proceedings of the sixth annual IEEE Symposium on Logic in Computer Science*, pages 342–349, 1991.

[Nip93] T. Nipkow. Orthogonal Higher-Order Rewrite Systems are Confluent. In *Proceedings of the International Conference on Typed Lambda Calculi and Application*, pages 306–317, 1993.

[Oos94] Vincent van Oostrom. *Confluence for Abstract and Higher-Order Rewriting.* PhD thesis, Vrije Universiteit, Amsterdam, March 1994.

[OR93] V. van Oostrom and F. van Raamsdonk. Comparing combinatory reduction systems and higher-order rewrite systems. Technical Report CS-R9361, CWI, 1993. Extended abstract in Proceedings of HOA'93.

[OR94] Vincent van Oostrom and Femke van Raamsdonk. Weak orthogonality implies confluence: The higher-order case. In *Proceedings of the third International Symposium on Logical Foundations of Computer Science*, pages 379–392, 1994.

[Raa93] F. van Raamsdonk. Confluence and superdevelopments. In C. Kirchner, editor, *Proceedings of the 5th International Conference on Rewrite Techniques and Applications*, 1993.

[Ros73] Barry K. Rosen. Tree-manipulating systems and Church-Rosser theorems. *Journal of the Association for Computing Machinery*, 20(1):160–187, January 1973.