Weak orthogonality implies confluence: the higher-order case

Vincent van Oostrom[*] and Femke van Raamsdonk[**]

Abstract

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

1 Introduction

This paper deals with higher-order rewriting. There seems to be no generally accepted definition of 'higher-order'. As far as rewriting is concerned, it seems reasonable to consider systems with rules describing how to transform objects, i.e. expressions over a first-order language to be 'first-order'. This corresponds to what is done in (first-order) term rewriting systems. A rewriting system is then to be called 'higher-order' if it may contain rules describing transformations of functions, not only of objects. In order to be able to write down rules describing transformations of functions, one should be able to denote functions by terms. After Church, functions are written as a term by means of an operator for abstraction. In the case of Church's λ-calculus, this operator is denoted by λ. For instance, the term denoting the function $f : x \mapsto x^2$ is $\lambda x.x^2$. So, for being 'higher-order' a rewriting system should contain an operator for abstraction such that functions can be denoted by terms. In this paper, we will use _._ to denote the abstraction operator for higher-order rewriting systems. As soon as one has abstraction, there is the need to express application of a function to an argument. In the case of λ-calculus, this is done by an binary application operator, which usually is denoted by juxtaposition of two terms. E.g. in λ-calculus, $(\lambda x.x^2)2$ denotes the application of the function $f : x \mapsto x^2$ to the argument 2. In this paper we choose for the applicative style, like in λ-calculus. Note that instead of having an application operator it is also possible to provide all function symbols and variables for functions with a fixed arity, i.e. the number of arguments to which they should be applied. Various formats of higher-order rewriting have been considered recently, which all have the possibility of expressing 'abstraction' and 'application'. We mention for instance the Contraction Schemes defined by Aczel, the Combinatory Reduction Systems defined by Klop, the Higher-order Rewrite Systems defined by Nipkow and the Higher-Order Term Rewriting Systems introduced by Wolfram. In spite of the common basic features, there are some subtle differences between these rewriting systems, which makes it difficult to compare their expressive power and makes it sometimes impossible to apply a result obtained for the one class to systems of another class.

The main difference between these systems is in the way the application of a function to an argument is evaluated, (see [OR93]), i.e. the way in which rules are instantiated. In general, explicitly or implicitly some language is used for instantiating rewrite rules. This language is a parameter of a Higher-Order Rewriting System, and is called a *substitution calculus*. Thus we attempt by introducing Higher-Order Rewriting Systems to unify the existing theory of higher-order rewriting by preserving the common basic features and by parametrizing over the

[*] Department of Mathematics and Computer Science, Vrije Universiteit, De Boelelaan 1081a, 1081 HV Amsterdam, email: oostrom@cs.vu.nl

[**] CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands, email: femke@cwi.nl, supported by NWO/SION project 612-316-606

differences. In addition, having a substitution calculus as a parameter of a rewriting system could lead to a better understanding of substitution in rewriting.

Intuitively, the rewrite rules of the substitution calculus describe the behaviour of abstraction, and the rewrite rules of the Higher-Order Rewriting System describe the behaviour of the operators that are typical for the Higher-Order Rewriting System. Most often the substitution calculus is a λ-calculus with β-reduction. The operators in the rewrite rule of the substitution calculus must be in the alphabet of the Higher-Order Rewriting System, otherwise the substitution calculus has no 'grip' on the terms of the Higher-Order Rewriting System. Further, performing a substitution should yield a unique result. This is ensured by requiring the substitution calculus to be confluent and terminating. Some more conditions are imposed on the rewrite relation of the substitution calculus in order to ensure a smooth interaction with the rules of the Higher-Order Rewrite System.

Now we will explain informally how to use a substitution calculus. Consider the following rewrite rule, calculating the derivative of the sum of two arbitrary functions f and g.

$$d(x.(fx+gx)) \rightarrow y.((d(x.fx))y + (d(x.gx))y)$$

We take the prime example of a substitution calculus, simply typed λ-calculus with β-reduction and η-expansion, as substitution calculus. A rewrite step due to this rule is obtained by instantiating the variables f and g. In Higher-Order Rewriting Systems, this is not done by a valuation, on a metalevel, but by means of the substitution calculus. The substitution calculus is used to substitute terms for bound variables. So, f and g must be turned into bound variables. Abstracting over f and g yields:

$$f.g.(d(x.fx+gx)) \rightarrow f.g.(y.((d(x.fx))y + (d(x.gx))y))$$

Suppose we want to instantiate f by $x.x^2$ and g by $x.5x$ (denoting the functions $x \mapsto x^2$ and $x \mapsto 5x$). We can build an expression M, having the left-hand side of the rule and $x.x^2$ and $x.5x$ as subexpressions, such that it can be reduced by β-reduction to the intended instance of the left-hand side: $M = (f.g.d(x.fx+gx))(x.x^2)(x.5x) \twoheadrightarrow_\beta d(x.(x^2+5x)) = N$. This instance of the left-hand side then can be rewritten to 'the same' instance of the right-hand side, which is obtained by replacing the left-hand side in M by the right-hand side and reducing to β-normal from: $M' = (f.g.(y.((d(x.fx))y + (d(x.gx)y))))(x.x^2)(x.5x) \twoheadrightarrow_\beta y.(d(x.x^2)y + d(x.5x)y) = N'$. We then have a rewrite step $N \rightarrow N'$. So rewriting in a Higher-Order Rewriting System takes place modulo the substitution calculus. We have now seen that a rewrite rule is instantiated by placing it in a suitable context which has the terms by which the variables are to be instantiated as subterms, and reducing it to normal form in the substitution calculus. Conversely, a term can be rewritten by a rewrite rule $l \rightarrow r$, if it can be expanded in the substitution calculus to a term having l as a subterm. By replacing l by r and reducing the result to normal form in the substitution calculus, we obtain a rewrite step. In the previous example, $d(x.(x^3+2x))$ can be rewritten to $y.(d(x.x^3)y + d(x.2x)y)$, since we have $d(x.(x^3+2x)) \ {}_\beta\!\twoheadleftarrow f.g.d(x.fx+gx)(x.x^3)(x.2x) \rightarrow f.g.(y.((d(x.fx))y + (d(x.gx)y)))(x.x^3)(x.2x) \twoheadrightarrow_\beta y.(d(x.x^3)y + d(x.2x)y)$.

After introducing Higher-Order Rewriting Systems, we prove all weakly orthogonal Higher-Order Rewriting Systems to be confluent. This extends the confluence results that have been obtained so far. First because confluence for higher-order rewriting so far has been obtained under the restriction of orthogonality. In orthogonal systems there are no critical pairs, whereas in weakly orthogonal systems the presence of trivial critical pairs is allowed. The proofs for the orthogonal case are done in two ways: by a method due to Tait and Martin-Löf and via developments. We prove confluence for weakly orthogonal higher-order rewriting in both ways. Second, our work generalizes work on confluence of higher-order rewriting since the class of Higher-Order Rewriting Systems contains all recently studied formats of higher-order rewriting. This is the case because of the parametrization over the substitution calculus. For a presentation of term rewriting systems, Higher-order Rewrite Systems and Combinatory Reduction Systems as Higher-Order Rewriting Systems, see [Oos94].. Complete proofs of the results in this paper can be found in [Oos94] and in a forthcoming CWI-technical report with the same title as the present paper. As applications of the main result of this paper we mention the following. All weakly confluent Higher-Order Rewriting Systems as defined by Nipkow are confluent. All weakly orthogonal Combinatory Reduction Systems as defined by Klop are confluent. Further, this yields a direct proof of (the well-known) confluence of system F with $\beta\eta$-reduction.

2 Higher-Order Rewriting Systems: syntax

In this section we give the definition of a Higher-Order Rewriting System. For an extensive account on higher-order rewriting we refer to [Oos94]. A Higher-Order Rewriting System (HORS) 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 abstraction, by means of which functions can be expressed as terms. We choose for the applicative style, where besides the symbols for the abstraction operator and the application operator, all other symbols are nullary. There are two kinds of operator symbols: symbols for substitution operators and symbols for rewrite or defined operators. The defined operators are the ones whose operational semantics is given by the rewrite rules of the Higher-Order Rewrite System; they are the 'real' operators. The substitution operators are used only by the rules of the substitution calculus.

DEFINITION 2.1 An *alphabet* $\mathcal{A}$ of a Higher-Order Rewriting System consists of:

- symbols $x\ y\ z\ \ldots$ for variables; among them are special symbols $\Box_1, \Box_2, \ldots$ for distinguished variables called holes in Var,
- a symbol Ap for the application operation,
- a symbol $\_.\_$ for abstraction,
- symbols $U\ V\ W\ \ldots$ for substitution operators in $\mathcal{O}_{\mathcal{SC}}$,
- symbols $F\ G\ H\ \ldots$ for rewrite or defined operators in $\mathcal{O}_{\mathcal{R}}$.

The union of $\mathcal{O}_{\mathcal{R}}$ and $\mathcal{O}_{\mathcal{SC}}$ is denoted by $\mathcal{O}$ and its elements are denoted by a. The alphabet of the substitution calculus, denoted by $\mathcal{A}_{\mathcal{SC}}$, consists of the symbols for variables, term forming symbols for application and abstraction, and of the symbols in $\mathcal{O}_{\mathcal{SC}}$. The alphabet $\mathcal{A}$ in fact consists of on the one hand symbols for the substitution calculus, for which, as we will see later on, rules are defined that give the semantics of abstraction (and its interaction with application), and on the other hand symbols for the Higher-Order Rewriting System, the semantics of which is given by the rewrite rules of the Higher-Order Rewriting System itself. Expressions over $\mathcal{A}$ are called *preterms*.

DEFINITION 2.2 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 is not in the scope of an abstraction $x.\_$ and *bound* otherwise. The set of variables that occur free in a preterm M is denoted by $\mathcal{F}\mathsf{Var}(M)$, and the set of variables that occur bound in t is denoted by $\mathcal{B}\mathsf{Var}(M)$. By the Variable Convention, one may assume $\mathcal{F}\mathsf{Var}(M) \cap \mathcal{B}\mathsf{Var}(M) = \emptyset$. If all variables occur bound in a preterm, then the preterm is said to be *closed*.

NOTATION 2.3 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 closed preterms $M_1, \ldots, M_n$ is denoted by $C[M_1, \ldots, M_n]$. An n-ary context is said to be *linear* if every hole $\Box_i$ (for $i = 1, \ldots, n$) occurs exactly once in it.

Positions are defined in a standard way, we denote them by ϕ, ψ, χ. The set of positions $\mathsf{Pos}(M)$ of a preterm M and the symbol of M at position ϕ, denoted by $\phi\backslash M$ are defined as usual.

As remarked above, two sets of rules operate on the set of preterms. The rewrite rules that give the semantics of the defined symbols will be defined below. First we will consider requirements the substitution calculus should satisfy. 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.

If a preterm contains redexes for the substitution calculus, then substitution is not yet carried out fully. One would like that calculating the result of substituting yields a result, and moreover that this result is unique. This is guaranteed by requiring the substitution calculus to be complete, i.e. confluent and terminating. As we will see later on, rewriting in a Higher-Order Rewriting System is performed modulo the substitution calculus. Completeness of the substitution calculus yields that every equivalence class of preterms has a unique representative which is found by reducing any member of the equivalence class to normal form with respect to the substitution calculus.

Further, there are two natural requirements on the convertibility relation of the substitution calculus, namely that it is closed under contexts and closed under substitution. A technical motivation for them will be given after defining rewrite rules for Higher-Order Rewriting Systems.

Finally, there is a requirement on the substitution calculus concerning its descendant relation. 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 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 a substitution calculus should have a descendant relation. For keeping the definition of a Higher-Order Rewriting System as general as possible, the form of the rules of a substitution calculus is not specified. Nevertheless one should have some information concerning their behaviour. Part of that is obtained by imposing two requirements on the descendant relation of a substitution calculus: we will require it to be 'natural'. First we define what is a descendant relation of a substitution calculus.

A descendant relation maps a step $u : M \rightarrow N$ to a relation $\lfloor u \rfloor$ between positions of M and positions of N. The descendant relation is extended straightforwardly to rewrite sequences of arbitrary length and to conversions.

DEFINITION 2.4 Let $\lfloor\ \rfloor_{\mathcal{SC}}$ 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 such that $u : C[\ ] \rightarrow_{\mathcal{SC}} D[\ ]$. Let the step u' be obtained by replacing the hole by a closed term M: $u' : C[M] \rightarrow_{\mathcal{SC}} D[M]$. 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 $D[M]$. That is, for $\phi \in \mathsf{Pos}(C[\,])$ and $\psi \in \mathsf{Pos}(D[\,])$, we have

$$\phi \lfloor u' \rfloor \psi$$

and for $\chi \in \mathsf{Pos}(M)$, ϕ' a position of the hole in $C[\,]$ and ψ' a position of the hole in $D[\,]$, we have $\phi'; \chi \lfloor u' \rfloor \psi'; \chi$.

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 $\lfloor d_1 \rfloor = \lfloor d_2 \rfloor$.

Both the definition of descendant relation and the one of 'naturality' can be given in more general setting, but this is beyond the scope of the present paper. See however [Oos94]. In the following definition the requirements on the substitution calculus that are discussed hitherto are listed.

DEFINITION 2.5 The rewrite rules of a substitution calculus 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 $\mathcal{A}$. The rewrite relation of the substitution calculus is denoted by $\rightarrow_{\mathcal{SC}}$.

2 (closure 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 $M \leftrightarrow^*_{\mathcal{SC}} M'$ then $C[M] \leftrightarrow^*_{\mathcal{SC}} C[M']$.

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

4 (naturality)
A substitution calculus $\mathcal{SC}$ must have a natural descendant relation. It is denoted by $\sqcup\!\!\!\rightarrow_{\mathcal{SC}}$.

EXAMPLE 2.6 The prime example of a substitution calculus is simply typed λ-calculus with β-reduction and η-expansion (see [Wol93]), here denoted by $\lambda^{\rightarrow}_{\overline{\eta}}$. This calculus satisfies all the requirements of a substitution calculus. It is the substitution calculus of Nipkow's Higher-Order Rewriting Systems ([Nip91]) and Wolfram's Higher-Order Term Rewriting Systemss ([Wol93]). Another example of a substitution calculus is λ-calculus with double β-developments, that is, λ-calculus where reductions consist of two consecutive developments of β-redexes. It is the substitution calculus of Klop's Combinatory Reduction Systems ([Klo80]).

Often we are interested in preterms in which the substitution is carried out completely, that is, in preterms that do not contain redexes for the substitution calculus.

DEFINITION 2.7 A preterm that is in normal form with respect to the substitution calculus is a *term*. The set of terms is denoted by Terms.

By completeness of the substitution calculus, each equivalence class of preterms contains a unique term, that is the representative of the equivalence class. All notions defined for preterms persist for terms, delete if necessary the prefix *pre*.

We will now clarify what is meant by well-formedness of a preterm. The substitution calculus is meant to substitute terms for bound variables. Let $x_1 \ldots x_n.M, N_1, \ldots, N_n$ be closed terms. In order to substitute $N_1, \ldots, N_n$ for $x_1, \ldots, x_n$ in M, one should build a preterm having $x_1 \ldots x_n.M$ and $N_1, \ldots, N_n$ as subterms, such that it reduces in the substitution calculus to $M[x_1 := N_1 \ldots x_n := N_n]$. Note that this isn't necessarily a $\mathcal{SC}$-normal form, since substituting N_i for x_i might yield new redexes for the substitution calculus. Only symbols in $\mathcal{A}_{\mathcal{SC}}$ should be used to glue the terms $x_1 \ldots x_n.M, N_1, \ldots, N_n$ together. That is, we build a $n+1$-ary linear context $C[, \ldots,]$ that consists solely of symbols in $\mathcal{A}_{\mathcal{SC}}$, such that $C[x_1 \ldots x_n.M, N_1, \ldots, N_n] \twoheadrightarrow_{\mathcal{SC}} M[x_1 := N_1 \ldots x_n := N_n]$. The unary linear context $C[\Box, N_1, \ldots, N_n]$ is called an *elco* (short for *elementary context*) for $x_1 \ldots x_n.M$. For the well-formedness of preterms, we require that whenever a preterm has a subterm of the form $\mathsf{elco}[x_1 \ldots x_n.M, N_1, \ldots, N_n]$, N_i is of the form $y_1 \ldots y_m.P$ if and only if x_i occurs in M in a subterm of the form $\mathsf{elco}[x_i, Q_1, \ldots, Q_m]$. If $\mathsf{elco}[x_1 \ldots x_n.M, N_1, \ldots, N_n]{\downarrow_{\mathcal{SC}}} = M'$, then M' is said to be an *instance* of M.

Now we come to the point where the rewrite rules of a Higher-Order Rewriting System can be defined. Usually in rewriting, rewrite rules contain free variables. These are assigned a value to by an assignment or valuation in order to obtain a rewrite step. So instantiating a left- or right-hand side is done on a metalevel. In Higher-Order Rewriting Systems, rewrite rules are instantiated by the substitution calculus. Therefore, the variables that are to be instantiated must be abstracted over 'on the outside'. Consider for instance the term rewriting rule $Fx \rightarrow Gx$. In the format of Higher-Order Rewriting Systems, with simply typed λ-calculus as substitution calculus, this rule is given by $x.Fx \rightarrow x.Gx$. The outermost abstractions can be thought of as universal quantifiers. The left- and right-hand side of a rule must have the same outermost abstractions. This guarantees that an elco for the left-hand side is also an elco for the right-hand side.

A common accepted restriction in rewriting is that it should not be possible to introduce arbitrary terms by rewriting. So usually one requires that all free variables in the right-hand side of a rule occur in the left-hand side as well. Rules of a Higher-Order Rewriting System do not contain any free variables at all, so the restriction in this form doesn't make any sense. Instead, for a rule of the form $x_1 \ldots x_n.l_0 \rightarrow x_1 \ldots x_n.r_0$, one requires that x_i occurs free in l_0 if it occurs free in r_0.

Next, in term rewriting the left-hand side of a rule is not allowed to be a variable, since otherwise an arbitrary term could be rewritten. A left-hand side must have a pattern, and if a term matches this pattern it can be rewritten by the rule in question. A symbol of the pattern can be traced in order to see what happens during a rewrite sequence. Rewriting in a Higher-Order Rewriting System takes place modulo the substitution calculus, so if patterns are to be traced they should be 'rigid' for the substitution calculus. We require the left-hand side of a rule to be of the form $x_1 \ldots x_n.Fl_1 \ldots l_m$ with F a defined symbol. Then every instance of the left-hand side is of the form $Fl'_1 \ldots l'_m$.

Finally, there is a condition on the 'well-formedness' of the rewrite rules. Let a rewrite rule have the form $x_1 \ldots x_n.l_0 \rightarrow x_1 \ldots x_n.r_0$. We require that x_i occurs in l_0 in a subterm of the form $\mathsf{elco}[x_i, y_1, \ldots, y_m]$ with $y_1, \ldots y_m$ ($m \geq 0$) bound in l_0, and that x_i then occur in r_0 in a subterm of the form $\mathsf{elco}[x_i, u_1, \ldots, u_m]$ with $u_1, \ldots, u_m$ arbitrary terms. The requirements on the form of the rewrite rules of a Higher-Order Rewriting System are listed in the following definition.

DEFINITION 2.8 A *rewrite rule* of a Higher-Order Rewriting System is a pair (l, r) of terms, usually written as $l \rightarrow r$, satisfying the following requirements:

1 l and r are closed terms, with the same outermost abstractions, say $l = x_1 \ldots x_n.l_0$ and $r = x_1 \ldots x_n.r_0$,

2 l_0 is of the form $Fl_1 \ldots l_m$, with F a defined symbol which is called the *head-symbol* of $l \rightarrow r$,

3 if x_i occurs free in r_0 then x_i occurs free in l_0,

4 for all $i \in \{1, \ldots, n\}$, x_i occurs in l_0 in a subterm of the form $\mathsf{elco}[x_i, y_1, \ldots, y_m]$ with y_j bound in l_0 for all $j \in \{1, \ldots, m\}$ and x_i occurs in r_0 in a subterm of the form $\mathsf{elco}[x_i, u_1, \ldots, u_n]$ with u_j arbitrary terms.

This definition of rewrite rule corresponds to the usual one for term rewriting systems and the ones given by Klop for Combinatory Reduction Systems [Klo80] and Nipkow for Higher-Order Rewriting Systems [Nip93]. Note that left- and right-hand sides are terms, so they do not contain redexes for the substitution calculus. It might be interesting to relax the last condition, but we do not pursue that idea in the present paper.

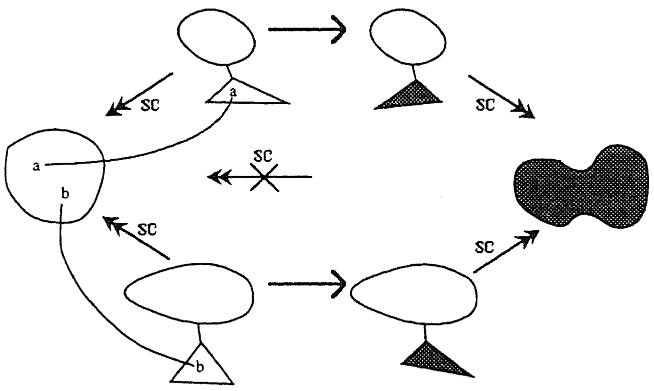
Now we come to the definition of the rewrite relation induced by the rewrite rules of the Higher-Order Rewriting System. Usually in rewriting, a term can be rewritten if it equals an instance of the left-hand side of some rule in some context $C[\ ]$. It can then be rewritten to that instance of the right-hand side of the rewrite rule in some context $C[\ ]$: $C[l^\sigma] \rightarrow C[r^\sigma]$. In Higher-Order Rewriting Systems, an instance of a left- or right-hand side of some rule is obtained by placing it in an elco and reducing So, a term can be rewritten by a rule $l \rightarrow r$ if it can be expanded by the substitution calculus to a context with l in it. This context should have an elco for l as subcontext. Then, by replacing l by r and reducing the so-obtained term to $\mathcal{SC}$-normal form, the result of the rewrite step is obtained. So a rewrite step $M \rightarrow N$ is built up as follows: $M \ {}_{\mathcal{SC}}\twoheadleftarrow C[l] \rightarrow C[r] \twoheadrightarrow^!_{\mathcal{SC}} N$. The step $C[l] \rightarrow C[r]$ is called a *replacement step.*

DEFINITION 2.9 Let $l \rightarrow r$ be a rewrite rule. A term M rewrites to a term N, or M is rewritten to N, notation $M \rightarrow N$, if $M \, {}_{SC}\!\twoheadleftarrow C[l]$ and $C[r] \twoheadrightarrow_{SC} N$, with $C[\,]$ a unary linear context that has an elco for l as a subcontext. Let ϕ be the position of the head-symbol of l in $C[l]\downarrow_{SC}$. The pair $(\phi, l \rightarrow r)$ is called a *redex*. The transitive closure of $\rightarrow$ is denoted by $\twoheadrightarrow^{+}$, and its reflexive-transitive closure by $\twoheadrightarrow$.

Note that the interaction between $C[\,]$ and l (or r) takes place in the subcontext of $C[\,]$ that is an elco for l. The term elco$[l, u_1, \ldots, u_n]\downarrow_{SC}$ with $l = x_1 \ldots x_n.l_0$ corresponds to l_0^{σ} with $\sigma = \{x_1 \mapsto u_1, \ldots, x_n \mapsto u_n\}$. In the definition, one uses a context, and not a precontext. Moreover, the context has a single occurrence of $\Box$, and rewrite rules are pairs of terms, not of preterms. That these are not real restrictions is proven in [Oos94]. In the proofs the fact that the convertibility relation of the substitution calculus is closed under substitution and under contexts is used. The definition of a Higher-Order Rewriting System is now completed. We proceed by giving the definitions of orthogonal and weakly orthogonal.

DEFINITION 2.10
- A term M is *linear* if every variable that occurs free in it M, occurs exactly once in it.
- A rewrite rule $l \rightarrow r$ with $l = x_1 \ldots x_n.l_0$ and $r = x_1 \ldots x_n.r_0$ is *left-linear* if l_0 is a linear term.
- A Higher-Order Rewriting System is *left-linear* if all its rewrite rules are left-linear.

DEFINITION 2.11
- Consider two rewrite rules $l \rightarrow r$ and $g \rightarrow d$. Let $C[\,]$ be an elco for $l \rightarrow r$. If $D[\,]$ is a context having an elco for $g \rightarrow d$ as subcontext, such that $C[l]\downarrow_{SC} = D[g]\downarrow_{SC}$ and the head-symbol of g in $D[g]\downarrow_{SC}$ is a defined symbol of l in $C[l]\downarrow_{SC}$, then $l \rightarrow r$ and $g \rightarrow d$ are said to be *ambiguous*. Let ϕ be the head-symbol of $l \rightarrow r$ in $C[l]\downarrow_{SC}$ and let ψ be the head-symbol of g in $g \rightarrow d$ in $D[g]\downarrow_{SC}$. The redex $(\psi, g \rightarrow d)$ is said to be *critical* for $(\phi, l \rightarrow r)$. If the positions are clear from the context, we say that $g \rightarrow d$ is critical for $l \rightarrow r$.
- Two rewrite rules $l \rightarrow r$ and $g \rightarrow d$ are *weakly ambiguous* if they are ambiguous with $C[l]\downarrow_{SC} = D[g]\downarrow_{SC}$ as in the previous clause, such that $C[r]\downarrow_{SC} = D[d]\downarrow_{SC}$.
- A Higher-Order Rewriting System is *ambiguous* if there is a pair of ambiguous rewrite rules.
- A Higher-Order Rewriting System is *non-ambiguous* if it is not ambiguous.
- A Higher-Order Rewriting System is *weakly non-ambiguous* if every pair of ambiguous rewrite rules is weakly ambiguous.



a pair of weakly ambiguous rewrite steps

DEFINITION 2.12 A Higher-Order Rewriting System is *orthogonal* if it is left-linear and non-ambiguous. A Higher-Order Rewriting System is *weakly orthogonal* if it is left-linear and weakly non-ambiguous.

3 Confluence for weakly orthogonal Higher-Order Rewriting Systems

In this section we prove twice that all weakly orthogonal Higher-Order Rewriting Systems are confluent.

3.1 A proof à la Tait and Martin-Löf

The proof method we employ is due to Tait and Martin-Löf. We define a relation $\Rightarrow$ for 'parallel rewriting' On Terms, such that its transitive closure equals rewriting. Then we prove the diamond property for $\Rightarrow$. That is, we prove that for any terms M, N, P such that $M \Rightarrow N$ and $M \Rightarrow P$ a term Q exists, satisfying $N \Rightarrow Q$ and $P \Rightarrow Q$. Confluence of rewriting is then an easy corollary. First the definition of $\Rightarrow$ is given.

DEFINITION 3.1 A relation $\Rightarrow$ on Terms is defined as follows:
1 $x \Rightarrow x$ for every variable x,
2 $a \Rightarrow a$ for every operator a,
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 $M_0M_1 \Rightarrow M_0'M_1'$.
5 if $l \to r$ is a rewrite rule and $C[\,]$ is an elco for l, such that $C[\,] \Rightarrow C'[\,]$, then $C[l]\downarrow_{SC} \Rightarrow C'[r]\downarrow_{SC}$.

The first step of the confluence proof is easy.

PROPOSITION 3.2 *The transitive closure of $\Rightarrow$ equals rewriting.*

For the proof of the diamond property we need a result concerning the interaction between substitution and parallel rewriting, and a Coherence Lemma. We cite them here without proof.

PROPOSITION 3.3
1 *Let* $\mathsf{elco}[\square, P_1, \ldots, P_n] \Rightarrow \mathsf{elco}[\square, P_1', \ldots, P_n']$ *and* $M = x_1 \ldots x_n.M_0 \Rightarrow M'$.
Then $\mathsf{elco}[M, P_1, \ldots, P_n]\downarrow_{SC} \Rightarrow \mathsf{elco}[M', P_1', \ldots, P_n']\downarrow_{SC}$.
2 *Let $C[\,]$ be a context with an elco for the term M as subcontext. Suppose $C[\,] \Rightarrow C'[\,]$ and $M \Rightarrow M'$. Then $C[M]\downarrow_{SC} \Rightarrow C'[M']\downarrow_{SC}$.*

PROOF. The proof of the first part proceeds by induction on the maximal length of the reduction of $\mathsf{elco}[M, P_1, \ldots, P_n]$ to SC-normal form. The second part is a corollary of the first part. □

LEMMA 3.4 (Coherence.) *Suppose $M = M_0M_1 = \mathsf{elco}[l, P_1, \ldots, P_n]\downarrow_{SC}$ with l the left-hand side of some rewrite rule $l \to r$. Suppose $M_0 \Rightarrow N_0$ and $M_1 \Rightarrow N_1$. If 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* $\mathsf{elco}[r, P_1, \ldots, P_n]\downarrow_{SC} \Rightarrow N_0N_1$.

THEOREM 3.5 *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. The base case is trivial. We only mention the difficult cases of the proof of the induction step.
1 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 $M = C[l]\downarrow_{SC}$ and $P = C'[r]\downarrow_{SC}$ for some rewrite rule $l \to r$ and an elco $C[\,]$ for l.

- 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 $C''[l]\downarrow_{SC}$ for some elco $C''[\,]$ for l. we have the following.

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

- 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 $C[l]\downarrow_{SC} = C[r]\downarrow_{SC}$. Then we have

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

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 $M_0 = D_0[g]\downarrow_{SC}$ with $D_0[\,]$ a context with an elco for g as subcontext, and N is of the form $D_0'[d]\downarrow_{SC}N_1$ with $D_0[\,] \Rightarrow D_0'[\,]$. If in $C[l]\downarrow_{SC} \Rightarrow C'[l]\downarrow_{SC} =: M_0'M_1'$ no redex critical for $g \to d$ is contracted, then M_0' is of the form $D_0''[g]\downarrow_{SC}$. We have

$$\begin{tikzcd}
C[l]\downarrow_{SC} = D_0[g]\downarrow_{SC}s_2 \arrow[rr, Rightarrow] \arrow[dd, Rightarrow] \arrow[dr, Rightarrow] & & C'[r]\downarrow_{SC} = P \arrow[dd, Rightarrow] \\
& D_0''[g]\downarrow_{SC}M_1' = C'[l]\downarrow_{SC} \arrow[ur] \arrow[dr, Rightarrow] & \\
N = D_0'[d]\downarrow_{SC}N_1 \arrow[rr, Rightarrow] & & E_0[d]\downarrow_{SC}Q_1
\end{tikzcd}$$

If in $C[l]\downarrow_{SC} \Rightarrow C'[l]\downarrow_{SC}$ a redex critical for $g \to d$ is contracted, then we consider two possibilities. If there are two disjoint redexes in s that are critical for $g \to d$, then we have $D_0[g]\downarrow_{SC} = D_0[d]\downarrow_{SC}$. Then

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

Suppose next that all redexes in $C[l]\downarrow_{SC}$ that are critical for $g \to d$ are nested and suppose at least one of them is contracted in $C[l]\downarrow_{SC} \Rightarrow C'[l]\downarrow_{SC}$. Let the largest one of them be an instance of $g' \to d'$. So $M_0 = E_0[g']\downarrow_{SC}$. 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 $C^*[l]\downarrow_{SC} = E_0[d']\downarrow_{SC}M_1$. By weak orthogonality, $E_0[d']\downarrow_{SC} = D_0[d]\downarrow_{SC}$. We have

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

2 Suppose $M \Rightarrow N$ is due to the last clause of the definition of $\Rightarrow$. Then $M = C[l]\downarrow_{SC}$ and $N = C'[r]$ 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 similar to the previous case. So suppose $M \Rightarrow P$ is also due to the last clause of the definition on $\Rightarrow$. Then $M = D[g]\downarrow_{SC}$ and $P = D'[d]\downarrow_{SC}$ for a rewrite rule $g \to d$ and an elco $D[\ ]$ for g. If in $C[l]\downarrow_{SC} \Rightarrow C'[l]\downarrow_{SC}$ no redex critical for $g \to d$ is contracted, then $C'[l]\downarrow_{SC} = D''[g]\downarrow_{SC}$. By weak orthogonality, we have $C'[r]\downarrow_{SC} = D''[d]\downarrow_{SC}$. Then we have

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

If in $C[l]\downarrow_{SC} \Rightarrow C'[l]\downarrow_{SC}$ a redex critical for $g \to d$ is contracted and in $D[g]\downarrow_{SC} \Rightarrow D'[g]\downarrow_{SC}$ a redex critical for $l \to r$ is contracted, then by weak orthogonality $C[r]\downarrow_{SC} = C[l]\downarrow_{SC} = D[g]\downarrow_{SC} = D[d]\downarrow_{SC}$. Then we have

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

□

3.2 A 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

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

in which in $N_i \twoheadrightarrow N_{i+1} \leftarrow M_{i+1}$ 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 $\mathcal{U}_i$.

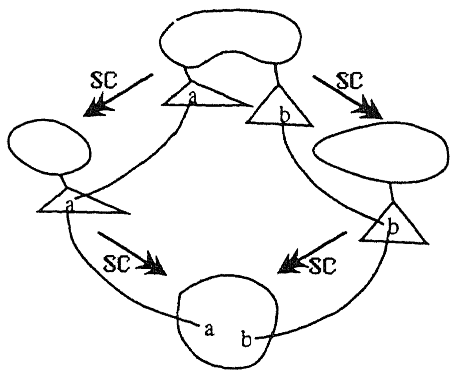
What problems do arise, when orthogonality is relaxed to weak orthogonality? The only problem is that the redex u_{i+1} might overlap with some redexes in the set $V := \{v | \exists u \in \mathcal{U}_i . u \lfloor u_i \rfloor v\}$ of residuals of $\mathcal{U}_i$ in M_{i+1}. But then we know by weak 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 3.6 Let $u : M \rightarrow N$ be a rewrite step, consisting of the expansion $e : M \;{}^{!}\!\twoheadleftarrow C[l]$, the replacement step $C[l \rightarrow r] : C[l] \rightarrow C[r]$ and the reduction $d : C[r] \twoheadrightarrow^{!} N$. The descendant relation induced by u is defined by $\lfloor u \rfloor := \lfloor e \rfloor; \lfloor C[l \rightarrow r] \rfloor; \lfloor d \rfloor$, where ; denotes relation composition. Descendants of redexes are defined via the descendant of their head-symbol.

Definition 3.7 Let $\mathcal{U} = \{u_1, \ldots, u_n\}$ be a set of redexes in a term M, where $u_i = (\phi_i, l_i \rightarrow r_i)$ is a redex at position ϕ_i in M with respect to rule $l_i \rightarrow r_i$.

1 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 $\mathcal{U}$.
2 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.



simultaneous extraction of two redexes

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 3.8 (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 $u : M \rightarrow M'$ and L and R the sets of left- and right-hand sides of $\mathcal{V}$.

1 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 $h[l] : C[l] \twoheadleftarrow_{SC} D[L,l]$, and the linear reduction $h[r] : D[L,r] \twoheadrightarrow_{SC} C[r]$ (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, $\lfloor g; D[L,l\to r]; f\rfloor = \lfloor e; C[l\to r]; d\rfloor$. This follows by some easy calculations.

2 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 $g' : M' \twoheadleftarrow_{SC} D'[L']$ of $\mathcal{U}'$ from M', which is smaller than g, in the following sense: $(D[R,r],n) \to^{+}_{SC} \times_{lex} > (D'[R'],n')$, 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 $h'[R] : D[R,r] \twoheadrightarrow_{SC} E[R] \equiv D'[R']$. 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 $D'[\ ,\ ]$.

3 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:

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

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 3.9 *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 3.10 *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.

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

2 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 propery called *cubicity.*

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

LEMMA 3.11 *Every Higher-Order Rewriting System is cubic.*

The next theorem states that every weakly orthogonal Higher-Order Rewriting System is confluent, solving a problem which was raised in [DJK93, Problem 61].

THEOREM 3.12 *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 Systemss) are confluent, thereby solving a problem which was raised by the first author in [DJK93, Problem 62].

THEOREM 3.13 *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 $\mathcal{S}$.*

3.3 Acknowledgements

We thank Jan Willem Klop, Fer-Jan de Vries and Zurab Khasidashvili for discussions and for remarks on previous versions of this paper.

References

[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. Pp. 72–81, Proceedings of 7th LICS, 1992.

[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.

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

[Nip91] Tobias Nipkow. Higher-order critical pairs. Pp. 342–349 of Proceedings of 6th LICS, 1991.

[Nip93] Tobias Nipkow. Orthogonal higher-order rewrite systems are confluent. Pp. 306–317 of LNCS 664, Proceedings of TLCA'93, 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. To appear in LNCS, Proceedings of HOA'93.

[Wol93] D.A. Wolfram. *The Clausal Theory of Types*, volume 21 of *Cambridge Tracts in Theoretical Computer Science.* Cambridge University Press, 1993.