Confluence and Superdevelopments

Femke van Raamsdonk*

CWI, P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Abstract. In this paper a short proof is presented for confluence of a quite general class of reduction systems, containing λ-calculus and term rewrite systems: the orthogonal combinatory reduction systems. Combinatory reduction systems (CRSs for short) were introduced by Klop generalizing an idea of Aczel. In CRSs, the usual first-order term rewriting format is extended with binding structures for variables. This permits to express besides first order term rewriting also λ-calculus, extensions of λ-calculus and proof normalizations. Confluence will be proved for orthogonal CRSs, that is, for those CRSs having left-linear rules and no critical pairs. The proof proceeds along the lines of the proof of Tait and Martin-Löf for confluence of λ-calculus, but uses a different notion of 'parallel reduction' as employed by Aczel. It gives rise to an extended notion of development, called 'superdevelopment'. A superdevelopment is a reduction sequence in which besides redexes that descend from the initial term also some redexes that are created during reduction may be contracted. For the case of λ-calculus, all superdevelopments are proved to be finite. A link with the confluence proof is provided by proving that superdevelopments characterize exactly the Aczel's notion of 'parallel reduction' used in order to obtain confluence.

1 Introduction

The study of λ-calculus and of the foundations of functional programming has led to a rich variety of classes of reduction systems, an important one being first-order term rewriting. For a lot of different although related reduction systems, different although related proofs of syntactic properties have been given.

A first attempt to provide a uniform framework for a number of extensions of λ-calculus was given by Hindley's $\lambda(a)$-reductions [3]. A much more extensive format in the form of contraction schemes, was developed by Aczel [1]. However, contraction schemes do not contain first-order term rewriting. Inspired by Aczel, Klop defined the Combinatory Reduction Systems (CRSs) [6]. The set-up of CRSs originates from term rewriting. Term rewriting systems (TRSs for short) handle pattern-matching, but they lack a notion of binding, like for instance in

$$\begin{array}{c} \int_a^b x^2 dx \\ \forall x(P(x) \Rightarrow Q(x)) \\ \lambda x.x \end{array}$$

* supported by NWO/SION project 612-316-606 *Extensions of orthogonal rewrite systems - syntactic properties.*

So the first thing to be done is adding binding structures for variables. If it comes to 'using' binding structures, a notion of substitution is needed, like in

$$1/3\ b^3 - 1/3\ a^3$$
$$P(t) \Rightarrow Q(t)$$
$$x[x := z]$$

This leads to introducing metavariables, that indicate places where an arbitrary term can be plugged in, with the possibility to express substitutions. In the case of CRSs, this is done by assigning to all metavariables a fixed arity. If it is not specified in the example which function, which predicates and which term is meant, the expressions can be written as

$$\int_a^b Z(x)dx$$
$$\forall x(Z(x) \Rightarrow Z'(x))$$
$$\lambda x.Z(x)$$

So CRSs can be viewed as TRSs with binding structures for variables and with metavariables having a fixed arity. Extending term rewriting considerably in this manner yields a general framework in which proofs of syntactic properties can be obtained in a uniform way.

In this paper, the property 'confluence' is a matter of concern. It means that for every two coinitial reduction sequences $s \twoheadrightarrow t$ and $s \twoheadrightarrow u$ a term v can be found such that $t \twoheadrightarrow v$ and $u \twoheadrightarrow v$. The term v is called a common reduct of t and u. An equivalent notion is the Church-Rosser property, stating that every pair of convertible terms has a common reduct.

In this paper confluence is proved for all orthogonal CRSs. Confluence for orthogonal CRSs has been proved by Klop [6], by proving first that all developments are finite. The strategy of the proof presented in this paper is very similar to the way confluence for λ-calculus has been proved by Tait and Martin-Löf. In a similar way, Aczel proves confluence for his contraction schemes.

In the proof a relation $\geq$ on terms is used which reflexive-transitive closure equals reduction. When characterizing this relation in terms of reduction, it turns out that the reduction sequences exactly corresponding to this relation $\geq$ form a generalization of developments; they are therefore called *superdevelopments*. We prove that superdevelopments in λ-calculus are always finite.

Another new proof of confluence of a large class of reduction systems with bound variables is given by Khasidashvili [5]. This proof proceeds along the lines of the one by Klop, but a slightly stronger version of the Church-Rosser property is proved. Takahashi [11] proves confluence of λ-calculi with conditional rules in a way similar to the method of Tait and Martin-Löf. Nipkow [8] proves confluence for orthogonal higher-order rewrite systems (HRSs) in the same manner. HRSs are close to CRSs but the starting point is λ-calculus rather than term rewriting. Kahrs investigates an extension of λ-calculus with first-order rewriting, and proves several syntactic properties [4]. Confluence results for another class of general reduction systems, the so-called subtree replacement systems, are studied by Rosen and O'Donnell in respectively [10] and [9].

2 Combinatory Reduction Systems

Metavariables and variables are distinguished in the following way:

- Metavariables indicate places in rules where an arbitrary term can be plugged in, like x in the TRS rule $F(x) \to G(x)$ and like M and N in the β-reduction rule for λ-calculus, $(\lambda x.M)N \to M[x := N]$.
- Variables are on the one hand used to build up terms, like in the term $F(x)$ in some TRS and in xz in λ-calculus, and on the other hand they are used to serve as placeholders indicating a place where a substitution can be carried out, like in the term $\lambda x.x$ in λ-calculus.

Metavariables occur only in left- and right-hand side of rules. A nullary metavariable Z occurring in the left-hand side of some rule, can be instantiated by an arbitrary term. A unary metavariable occurring in the form $Z(x)$ in the left-hand side of a rule can be instantiated by a term t possibly but not necessarily containing free occurrences of the variable x. If this metavariable occurs in the right-hand side of the rule in the form $Z(s)$ then it is instantiated by the term t in which all free occurrences of x are replaced by s. For example, the β-reduction rule of λ-calculus, with metavariables M and N is usually written as $(\lambda x.M)N \to M[x := N]$ but is given as well as $(\lambda x.M(x))N \to M(N)$. In the latter formulation, the metavariables M and N have arity 1 and 0 respectively.

The alphabet of a CRS consists of

- variables, written as $x\ y\ z\ \ldots$,
- metavariables with a fixed arity, written as $Z\ Z_0\ Z_1\ \ldots$,
- function symbols with a fixed arity, written as $F\ G\ H\ \ldots$,
- an abstraction operator, written as $[\_]\_$,
- parentheses and commas.

A term t from which some variable x has been abstracted is written as $[x]t$. Function symbols of arity 0 are called *constant symbols*. Metaterms and terms are built up from the alphabet given above.

Definition 1. The set MTerms of metaterms is the smallest set satisfying
(1) $x \in$ MTerms for every variable x,
(2) if $t \in$ MTerms then $[x]t \in$ MTerms for every variable x,
(3) if F is a function symbol of arity n and $t_1, \ldots, t_n \in$ MTerms, then $F(t_1, \ldots, t_n) \in$ MTerms,
(4) if Z is a metavariable of arity n and $t_1, \ldots, t_n \in$ MTerms, then $Z(t_1, \ldots, t_n) \in$ MTerms.

The set Terms of terms consists of all metaterms without any metavariable.

If C is a constant symbol then we write C instead of $C()$. Identity on MTerms and on Terms is denoted by $=$.

In a metaterm or term of the form $[x]t$, we call t the *scope* of the abstraction $[x]$. A variable x occurs *free* in a metaterm or term if it is not in the scope of a $[x]$. It occurs *bound* otherwise. A metaterm or term is called *closed* if all variables occur bound. Like in λ-calculus, bound variables can be renamed. (Meta)terms that are identical up to a renaming of bound variables are considered equal. This

permits to adopt the convention that in a term no variable is abstracted over twice or more. Instead of $[x_1]\ldots[x_n]t$ we write $[x_1\ldots x_n]t$.

Note that the abstraction in a (meta)term of the form $[x]t$, is purely syntactic. The actual (operational) meaning of this abstraction has to be expressed by a function symbol and its reduction rules.

Let $\Box$ be a fresh symbol. A *context* is a (meta)term with one or more occurrences of $\Box$. A context with exactly one occurrence of $\Box$ is written as $C[\,]$, and one with n occurrences of $\Box$ as $C[,\ldots,]$. If $C[,\ldots,]$ is a context with n occurrences of $\Box$ and $t_1,\ldots,t_n$ are (meta)terms, then $C[t_1,\ldots,t_n]$ denotes the result of replacing from left to right the occurrences of $\Box$ by $t_1,\ldots,t_n$. A (meta)term s is said to be a *sub(meta)term* of of a (meta)term t if a context $C[\,]$ exist such that $t = C[s]$.

Example 1. The alphabet of the untyped λ-calculus consists of
- a unary function symbol λ for λ-abstraction
- a binary function symbol Ap for application

Then we write for instance $\lambda([x]x)$ for $\lambda x.x$, $\lambda([x]\mathsf{Ap}(x,y))$ for $\lambda x.xy$, and $\mathsf{Ap}(\lambda([x]y),z)$ for $(\lambda x.y)z$.

Note that according to the definition of Terms from this alphabet a lot of terms can be built that don't correspond to λ-terms. One reason is that it cannot be specified that in order to form a λ-term, the argument of the symbol λ must be an abstraction term. This can be done by extending the notion of arity. The presence of 'junk' terms doesn't yield a problem when applying for instance the confluence result, since a term corresponding to a λ-term reduces always to terms corresponding to λ-terms.

A reduction relation on the terms of a CRS is generated by a set of reduction rules.

Definition 2. A *reduction rule* is a pair of metaterms (α,β) written as $\alpha \to \beta$, satisfying the following constraints:
- α and β are closed,
- α is of the form $F(\alpha_1,\ldots,\alpha_n)$,
- metavariables occurring in β occur in α as well,
- metavariables in α occur only in the form $Z(x_1,\ldots,x_n)$ with $x_1,\ldots,x_n$ distinct variables.

A reduction rule acts as a scheme from which actual reduction steps can be obtained. Metavariables in the left-hand side of a rule indicate places where an arbitrary term can be substituted. Variables occur only bound and serve to indicate places where substitutions are carried out. A left-hand side of a rule is not allowed to be a metavariable nor an abstraction term; the former because this would permit to rewrite an arbitrary term and the latter because the abstraction is considered to be purely syntactic without any operational meaning. Allowing to have metavariables in β that do not occur in α would permit to introduce terms out of the blue by reducing. Substitution mechanisms are expressed by the reduction rules as follows: $Z(x_1,\ldots,x_n)$ in the left-hand side of some rule

is instantiated by a term s possibly but not necessarily containing the variables $x_1, \ldots, x_n$. An occurrence of Z in the right-hand side in the form $Z(t_1, \ldots, t_n)$ is then instantiated by s in which the free occurrences of $x_1, \ldots, x_n$ are replaced by $t_1, \ldots, t_n$ respectively.

Example 2. The β-reduction rule of λ-calculus is in CRS format written as $\mathsf{Ap}(\lambda([x]Z(x)), Z') \to Z(Z')$.

Now it will be described how the reduction rules generate an actual reduction relation on Terms. Therefore the concept of 'valuation' is introduced. Valuations express how metavariables are instantiated by terms. Before defining valuations we introduce as a notational device the n-ary *substitute* (a name due to Kahrs [4]).

Definition 3. Let t be a term is some CRS R.
(1) For an n-tuple of distinct variables $x_1, \ldots, x_n$, the expression $\underline{\lambda}(x_1, \ldots . x_n).t$ is an n-ary substitute.
(2) An n-ary substitute $\underline{\lambda}(x_1, \ldots . x_n).t$ can be applied to an n-tuple of terms $(s_1, \ldots, s_n)$, yielding as result the term t with $s_1, \ldots, s_n$ simultaneously substituted for $x_1, \ldots, x_n$ respectively:

$$(\underline{\lambda}(x_1, \ldots . x_n).t)(s_1, \ldots, s_n) = t[x_1 := s_1, \ldots, x_n := s_n]$$

So an n-ary substitute can be considered as a function $\mathsf{Terms}^n \to \mathsf{Terms}$. For an n-ary substitute $\underline{\lambda}(x_1, \ldots . x_n).t$ the variables $x_1, \ldots, x_n$ are considered to be bound in t and may be renamed so that no name clashes occur. The variables in t that don't occur in $(x_1, \ldots, x_n)$ and that are not bound in t are called the free variables of the substitute $\underline{\lambda}(x_1, \ldots . x_n).t$.

Now the definition of a valuation can be given.

Definition 4. A valuation is a map σ that assigns to an n-ary metavariable Z an n-ary substitute:

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

The map σ is extended to a homomorphism on metaterms (denoted as σ as well) in the following way:
(1) $\sigma(x) = x$,
(2) $\sigma([x]t) = [x]\sigma(t)$ for a variable x and a metaterm t,
(3) $\sigma(F(t_1, \ldots, t_n)) = F(\sigma(t_1), \ldots, \sigma(t_n))$ for a function symbol F of arity n and metaterms $t_1, \ldots, t_n$,
(4) $\sigma(Z(t_1, \ldots, t_n)) = \sigma(Z)(\sigma(t_1), \ldots, \sigma(t_n))$ for an metavariable Z of arity n and metaterms $t_1, \ldots, t_n$.

Without any conditions on the valuations, in instantiated reduction rules variables can be bound unintendedly. Two kinds of problems can occur.

First, variables can be captured by abstractors by plugging in terms for metavariables. This is for instance the case if $F([x]Z)$ is instantiated by a valuation that assigns x to Z. So we have to require that bound variables in rules are renamed such that they differ from free variables in substitutes.

Second, in the instance of a right-hand side, substitution can yield unintended bindings. This is for example the case if $Z(Z')$ is instantiated by the valuation σ with $\sigma(Z(u)) = \underline{\lambda}(u).F([y]u)$ and $\sigma(Z') = y$. Then $\sigma(Z(Z')) = F([y]y)$. Situations like this can be avoided by requiring for every two different metavariables $Z_1(x_1, \ldots, x_{k_1})$ and $Z_2(x_1, \ldots, x_{k_2})$ occurring in the same reduction rule, the free variables in $\sigma(Z_1(x_1, \ldots, x_{k_1}))$ to be different from the bound variables in $\sigma(Z_2(x_1, \ldots, x_{k_2}))$. In the following we will suppose that these requirements are met. Informally, they can be thought of as 'rename bound variables as much as possible, in order to avoid free occurrences of x to be captured unintentionally by abstractors $[x]$.'

Finally the actual reduction relation $\rightarrow$ on Terms can be defined.

Definition 5. Let $\alpha \rightarrow \beta$ be a reduction rule and σ a valuation. An instance $\sigma(\alpha)$ of the left-hand side of a reduction rule is called a *redex*. The associated right-hand side $\sigma(\beta)$ is called its *contractum*. Replacing a redex by its contractum in a context is called a *reduction step* and is written as $C[\sigma(\alpha)] \rightarrow C[\sigma(\beta)]$. A sequence of zero or more reduction steps is called a *reduction sequence* or *reduction*. If a reduction from s to t exists we write $s \twoheadrightarrow t$ and say that s *reduces* to t, and t is called a *reduct* of s.

Example 3. The reduction step $(\lambda x.x)y \rightarrow y$ in λ-calculus is obtained by instantiating the β-reduction rule $\mathsf{Ap}(\lambda([x]Z(x)), Z') \rightarrow Z(Z')$ in the following way: $\sigma(Z) = \underline{\lambda}(z).z$ and $\sigma(Z') = y$. Then $\sigma(\lambda[x].Z(x)) = \lambda([x]x)$ so the left-hand side of the rule instantiated by σ is $\mathsf{Ap}(\lambda[x]x, y)$. This term reduces to $\sigma(Z(Z')) = (\underline{\lambda}(z).z)(y) = y$.

A CRS is called *left-linear* if in none of its reduction rules the same metavariable occurs twice or more in the left-hand-side. Two reduction rules $\alpha \rightarrow \beta$ and $\alpha' \rightarrow \beta'$ are said to *overlap* if there exist valuations σ and σ' such that $\sigma(\alpha) = \sigma'(\alpha'_1)$ for a submetaterm α'_1 of α' that is not of the form $Z(x_1, \ldots, x_n)$. Then a context $C[\,]$ exists such that $C[\sigma(\alpha)] = \sigma'(\alpha')$. In the case that the reduction rules $\alpha \rightarrow \beta$ and $\alpha' \rightarrow \beta'$ are the same, we require additionally the context $C[\,]$ to be non-trivial. The term $C[\sigma(\alpha)] = \sigma'(\alpha')$ can be reduced in two different ways, yielding as a result $C[\sigma(\beta)]$ or $\sigma'(\beta')$ respectively.

If a CRS doesn't contain overlapping rules then it is called *non-ambiguous*. If for every two overlapping rules $\alpha \rightarrow \beta$ and $\alpha' \rightarrow \beta'$ with $C[\sigma(\alpha)] = \sigma'(\alpha')$ it holds that $C[\sigma(\beta)] = \sigma'(\beta')$, then the CRS is said to be *weakly non-ambiguous*.

A CRS that is left-linear and non-ambiguous is called *orthogonal*. A CRS that is left-linear and weakly non-ambiguous is called *weakly orthogonal*.

Example 4. An orthogonal CRS is λ-calculus with β-reduction; if η-reduction is added it is a weakly orthogonal one. The η-rule is in CRS format written as $\lambda([x]\mathsf{Ap}(Z, x)) \rightarrow Z$. Each time that a term contains a β-and an η-redex such that they share a λ, both possible reduction steps yield the same result. For instance, the term $@(\lambda([x]@(y, x), z))$ is itself a β-redex and contains non-trivially the η-redex $\lambda([x]@(y, x))$. Reducing the β-redex yields the same result as reducing the η-redex, namely $@(y, z)$.

CRSs themselves are untyped systems. Nevertheless various typed systems like simply typed λ-calculus and system F can be written in the CRS framework.

In the original definition of CRSs by Klop, all function symbols are nullary and a distinguished symbol for application is used. Metavariables have, exactly like in this set-up, a fixed arity.

It is easily seen that every applicative system can be written as a functional one by writing all applications explicitly. A functional system can be written in applicative format by turning all function symbols into constants and adding a binary operator for application that is not written explicitly. Then the functional CRS correspond to a sub-CRS of its applicative version. A *sub-CRS* of a CRS R is defined as a CRS obtained from R by restricting the set of terms to a subset that is closed under reduction. So the functional and applicative formats have the same expressive power.

The confluence result, like most syntactic results for CRSs, carries over directly to sub-CRSs. This explains why it is no problem when the CRS representation of a system contains 'junk' terms.

3 Confluence for Orthogonal CRSs

In this section all orthogonal CRSs will be proved to be confluent. The strategy of the proof is essentially the same as the one of the proof of confluence for λ-calculus with β-reduction by Tait and Martin-Löf [2]. This proof method is employed by several others [1], [8], [11], mostly in order to prove confluence.

A relation $\geq$ on Terms is defined such that its reflexive-transitive closure equals reduction. For this relation $\geq$ the diamond property, given in the next definition, will be proved.

Definition 6. A binary relation $\triangleright$ satisfies the *diamond property* if for every a, b and c such that $a \triangleright b$ and $a \triangleright c$ there exists a d such that $b \triangleright d$ and $c \triangleright d$.

Having proved the diamond property for $\geq$, confluence of the reduction relation follows immediately.

Definition 7. The relation $\geq$ on Terms is defined as follows:
(1) $x \geq x$ for every variable x,
(2) if $s \geq t$ then $[x]s \geq [x]t$ for every variable x,
(3) if $s_1 \geq t_1, \ldots, s_n \geq t_n$ then $F(s_1, \ldots, s_n) \geq F(t_1, \ldots, t_n)$ for every n-ary function symbol F,
(4) if $s_1 \geq t_1, \ldots, s_n \geq t_n$ and $F(t_1, \ldots, t_n) = \sigma(\alpha)$ for some reduction rule $\alpha \to \beta$ and valuation σ, then $F(s_1, \ldots, s_n) \geq \sigma(\beta)$.

The fourth clause can be depicted as follows:

$$\begin{array}{ccccc} & & F(s_1, \ldots, s_n) & & \\ & & \| & \| & \| \\ \sigma(\alpha) = & & F(t_1, \ldots, t_n) & \to & \sigma(\beta) \end{array}$$

The first three clauses of the definition state that $\geq$ is a reflexive relation that is closed under term formation. The fourth clause expresses that $s \geq t$ if s reduces to t by a parallel 'inside-out' reduction, in which possibly redexes that are 'created upwards' are contracted. Note that in this clause $F(s_1, \ldots, s_n)$ is not necessarily a redex.

The next proposition states that $\geq$ is indeed a useful relation to prove the diamond property for.

Proposition 8. *The transitive closure of $\geq$ equals reduction.*

The crucial step in proving the diamond property for $\geq$ is proving that $\geq$ satisfies a property named 'coherence'. This notion is originally introduced by Aczel [6].

Definition 9. A binary relation $\rhd$ on Terms is said to be *coherent* with respect to reduction if the following holds: if $F(a_1, \ldots, a_n) = \sigma(\alpha)$ for some reduction rule $\alpha \to \beta$ and valuation σ, and $a_1 \rhd b_1, \ldots, a_n \rhd b_n$, then we have for some valuation τ that $F(b_1, \ldots, b_n) = \tau(\alpha)$ with $\sigma(\beta) \rhd \tau(\beta)$.

Coherence can be depicted as follows:

$$\begin{array}{ccc} F(a_1, \ldots, a_n) & \to & a \\ \bigtriangledown & & \bigtriangledown \\ F(b_1, \ldots, b_n) & \to & b \end{array}$$

Two technical propositions are needed in order to prove coherence of $\geq$ with respect to reduction.

Proposition 10. *If $a \geq b$ and $s_1 \geq t_1, \ldots, s_n \geq t_n$ then*

$$a[x_1 := s_1, \ldots, x_n := s_n] \geq b[x_1 := t_1, \ldots . x_n := t_n]$$

PROOF. Induction on the derivation of $a \geq b$. □

Proposition 11. *Let t be a metaterm containing only the metavariables $Z_1, \ldots, Z_k$. Let σ and τ be valuations. If $\sigma(Z_i(x_1, \ldots, x_{n_i})) \geq \tau(Z_i(x_1, \ldots, x_{n_i}))$ for $i = 1, \ldots, k$, then $\sigma(t) \geq \tau(t)$.*

PROOF. Induction on the structure of t. □

Lemma 12. *The relation $\geq$ is coherent with respect to reduction.*

PROOF. Suppose $F(a_1, \ldots, a_n) = \sigma(\alpha) \to \sigma(\beta)$ and $a_1 \geq b_1, \ldots, a_n \geq b_n$. By Proposition 8 we have $a_1 \twoheadrightarrow b_1, \ldots, a_n \twoheadrightarrow b_n$. By non-ambiguity, we know that all reduction steps take place in instances of metavariables in α. Together with left-linearity this yields that $F(b_1, \ldots, b_n)$ is still an instance of α, say $F(b_1, \ldots, b_n) = \tau(\alpha)$ with contractum $\tau(\beta)$. Now it has to be proved that $\sigma(\beta) \geq \tau(\beta)$. Appropriate first parts of derivations of $a_1 \geq b_1, \ldots, a_n \geq b_n$ form a derivation for $\sigma(Z_i(x_1, \ldots, x_{k_i})) \geq \tau(Z_i(x_1, \ldots, x_{k_i}))$ for every k_i-ary metavariable Z_i occurring in α. Note that metavariables in α only occur in this form. In β occur only metavariables that occur in α as well, so by Proposition 11 we can conclude $\sigma(\beta) \geq \tau(\beta)$. □

Theorem 13. *The relation $\geq$ satisfies the diamond property.*

PROOF. We shall prove that for any a, b and c such that $a \geq b$ and $a \geq c$ there exists a d such that $b \geq d$ and $c \geq d$. The proof proceeds by induction on the derivation of $a \geq b$.

- If $a \geq b$ is $x \geq x$ then necessarily $c = x$. Take $d := x$.
- If $a \geq b$ is $[x]a' \geq [x]b'$ with $a' \geq b'$, then c is necessarily of the form $[x]c'$. By induction hypothesis, a d' exists such that $b' \geq d'$ and $c' \geq d'$. By defining $d := [x]d'$, both $b \geq d$ and $c \geq d$ are satisfied.
- If $a \geq b$ is $F(a_1, \ldots, a_n) \geq F(b_1, \ldots, b_n)$ with $a_1 \geq b_1, \ldots, a_n \geq b_n$, then $a \geq c$ can be due to the third or to the fourth clause of the definition of $\geq$.

  First we consider the case that $a \geq c$ is $F(a_1, \ldots, a_n) \geq F(c_1, \ldots, c_n)$ with $a_1 \geq c_1, \ldots, a_n \geq c_n$. By induction hypothesis $d_1, \ldots, d_n$ exist such that $b_i \geq d_i$ and $c_i \geq d_i$ for $i = 1, \ldots, n$. Define $d := F(d_1, \ldots, d_n)$. Then $b \geq d$ and $c \geq d$ hold.

  Second we consider the case that $a \geq c$ is due to the fourth clause of the definition of $\geq$. In that case $a = F(a_1, \ldots, a_n)$ and there exist $c_1, \ldots, c_n$ such that $a_1 \geq c_1, \ldots, a_n \geq c_n$ and $F(c_1, \ldots, c_n) = \sigma(\alpha) \to \sigma(\beta) = c$ for some reduction rule $\alpha \to \beta$ and valuation σ. By induction hypothesis $d_1, \ldots, d_n$ exist such that $b_i \geq d_i$ and $c_i \geq d_i$ for $i = 1, \ldots, n$. By coherence of $\geq$ we have $F(d_1, \ldots, d_n) = \tau(\alpha)$ with $\sigma(\beta) \geq \tau(\beta)$. Define $d := \tau(\beta)$. Then $b \geq d$ holds by the fourth clause of the definition of $\geq$ and $c \geq d$ holds by coherence of $\geq$.
- The last case to be considered is when $a \geq b$ is due to the fourth clause of the definition of $\geq$. Then $a = F(a_1, \ldots, a_n)$ and there exist $b_1, \ldots, b_n$ such that $a_1 \geq b_1, \ldots, a_n \geq b_n$ and $F(b_1, \ldots, b_n)$ is a redex with contractum b, say $F(b_1, \ldots, b_n) = \sigma(\alpha) \to \sigma(\beta) = b$ for a reduction rule $\alpha \to \beta$. Again, $a \geq c$ can be due to either the third or the fourth clause of the definition of $\geq$.

  The case that $a \geq c$ is due to the third clause of the definition is similar to the second case in the previous step in the proof.

  Second we consider the case that $a \geq c$ is a consequence of the fourth clause of the definition of $\geq$. Then there exist $c_1, \ldots, c_n$ such that $a_1 \geq c_1, \ldots, a_n \geq c_n$ and we have $F(c_1, \ldots, c_n) = \sigma'(\alpha') \to \sigma'(\beta') = c$ for some reduction rule $\alpha' \to \beta'$ and valuation σ'. By induction hypothesis, $d_1, \ldots, d_n$ exist such that $b_1 \geq d_i$ and $c_i \geq d_i$ for $i = 1, \ldots, n$. Coherence of $\geq$ yields that one has $F(d_1, \ldots, d_n) = \tau(\alpha)$ with $\sigma(\beta) = b \geq \tau(\beta)$ and $F(d_1, \ldots, d_n) = \tau'(\alpha')$ with $\sigma'(\beta') = c \geq \tau'(\beta')$. So $\tau(\alpha) = \tau'(\alpha')$ and by orthogonality we have $\tau = \tau'$ and $\alpha = \alpha'$. Define $d := \tau(\beta)$. By coherence we have $b = \sigma(\beta) \geq d$ and $c = \sigma'(\beta') \geq d$.

□

The main result of this section is a direct result of this theorem.

Corollary 14. *All orthogonal CRSs are confluent.*

4 Superdevelopments for λ-calculus

In this section we fix attention to λ-calculus. Confluence for λ-calculus is proved by Tait and Martin-Löf using a relation $\mapsto_1$ whose transitive closure equals reduction. Another proof of confluence can be given by first proving that all developments are finite (see [2]). A development is a reduction sequence in which only descendents of redexes that are present in the initial term may be contracted. It is not allowed to contract redexes that are created along the way. The crucial notions in both proofs are related in the following way: $M \mapsto_1 N$ if and only if a (complete) development $M \twoheadrightarrow N$ exists (see [2]).

The relation between $\geq$ and $\mapsto_1$ is as follows: $M \mapsto_1 N$ implies $M \geq N$ but not necessarily vice versa. Questions arising are: can the reduction sequences corresponding exactly to $\geq$ be characterized, and, if so, are these reduction sequences always finite?

In this section we shall characterize the reduction sequences corresponding exactly to the relation $\geq$ on λ-terms. In order to do so, a set of labelled λ-terms Λ_l and labelled β-reduction $\rightarrow_{\beta_l}$ on them will be defined. Lambda's will be labelled by a label from a countably infinite set of labels I, and application nodes will be labelled by a subset of I. If the labelling of a λ-term M satisfies certain conditions, then its β_l-reduction to normal form is, after having erased all labels, a superdevelopment. All superdevelopments are proved to be finite.

In [7] Lévy analyses the different ways in which β-redexes can be created. The following possibilities are distinguished (written in the usual notation for λ-calculus):

(1) $((\lambda x.\lambda y.M)N)P \rightarrow_\beta (\lambda y.M[x := N])P$
(2) $(\lambda x.x)(\lambda y.M)N \rightarrow_\beta (\lambda y.M)N$
(3) $(\lambda x.C[xM])(\lambda y.N) \rightarrow_\beta C'[(\lambda y.N)M']$ where C' and M' stand for C respectively M in which all free occurrences of x have been replaced by $\lambda y.N$.

The first two created redexes are 'innocent' and may be contracted in a superdevelopment. Note that, if we think of a λ-term as a tree built from application- and λ-nodes, the redexes in the first two cases are 'created upwards'. In the last case, on the other hand, the redex isn't created upwards, and may not be contracted in a superdevelopment. The result that all superdevelopments are finite illustrates that all infinite β-reduction sequences in λ-calculus are due to the third way of redex creation; indeed redex creation e.g. in the reduction sequence of $(\lambda x.xx)(\lambda x.xx)$ happens in this way.

In the following, we shall write the application nodes explicitly, but abstraction terms as usual. Further, the relation $\geq$ when only used on λ-terms can be simplified a bit.

Definition 15. The relation $\geq$ on λ-terms is defined in the following way:

(1) $x \geq x$ for each variable x,
(2) if $M \geq M'$ then $\lambda x.M \geq \lambda x.M'$ for a λ-term M,
(3) if $M \geq M'$ and $N \geq N'$ then $\mathsf{Ap}(M, N) \geq \mathsf{Ap}(M', N')$ for λ-terms M and N,

(4) if $M \geq \lambda x.M'$ and $N \geq N'$, then $\mathsf{Ap}(M,N) \geq M'[x := N']$ for λ-terms M and N.

We proceed by defining the set of labelled λ-terms.

Definition 16. The set Λ_l of labelled λ-terms is defined as the smallest set such that
(1) $x \in \Lambda_l$ for every variable x,
(2) if $M \in \Lambda_l$ and $i \in I$, then $\lambda_i x.M \in \Lambda_l$,
(3) if $M, N \in \Lambda_l$ and $X \subset I$, then $\mathsf{Ap}^X(M,N) \in \Lambda_l$.

Erasing all labels of a term $M \in \Lambda_l$ is done by a function $E : \Lambda_l \to \Lambda$ that is defined inductively as follows:

$$E(x) = x$$
$$E(\lambda_i x.M) = \lambda x.E(M)$$
$$E(\mathsf{Ap}^X(M_1, M_2)) = \mathsf{Ap}(E(M_1), E(M_2))$$

Let $\wp(I)$ denote the powerset of I, i.e. the set of all subsets of I. A *labelling* L for a λ-term M is a partial function from the symbols of M to $I \cup \wp(I)$, that is only defined on symbols Ap and λ, to which a subset of I respectively an element of I is assigned. The result of applying L to M is written as M^L. So $E(M^L) = M$.

The reduction rule β_l on Λ_l is defined as

$$\mathsf{Ap}^X(\lambda_i x.M, N) \to_{\beta_l} M[x := N] \quad \text{if } i \in X$$

where the substitution $[x := N]$ is defined as usual. Like usually in λ-calculus, we adopt the variable convention, i.e. all bound variables in a statement are supposed to be different from the free ones.

Definition 17.
- A term $M \in \Lambda_l$ is called *good* if no label X of an application node contains the index i of a λ occurring outside the scope of this application node.
- A labelling L is said to be *good for* a λ-term M if M^L is a good term.
- A labelling L is an *initial labelling* for a λ-term M if it is good for M and all λ's have a different label.
- Two labelling are said to be *disjoint* if no element of I occurs in both labellings.
- Two terms M and N of Λ_l are said to be *disjointly labelled* if there exist λ-terms M' and N' and disjoint labellings L_1 and L_2 such that $M'^{L_1} = M$ and $N'^{L_2} = N$.

For example, a good term is $\mathsf{Ap}^{\{2\}}(\mathsf{Ap}^{\{1\}}(\lambda_1 x.\lambda_2 y.xy, z), u)$, but, on the other hand, the term $\mathsf{Ap}^{\{1\}}(\lambda_1 x.\mathsf{Ap}^{\{2\}}(x,y), \lambda_2 y.y)$ isn't good. It is clear that all subterms of a good term are good. The property 'good' is preserved under reduction, i.e. β_l-reduction cannot push a λ outside the scope of an application node in which it occurred originally. This is proved in the following proposition, that will be used implicitly.

Proposition 18. *If $M \in \Lambda_I$ is a good term and $M \to_{\beta_I} N$, then N is a good term.*

PROOF. Let $\mathsf{Ap}^X(\lambda_i x.P, Q)$ be a good term with $i \in X$ (so it is a β_I-redex). It is proved by induction on the structure of P that $P[x := Q]$ is a good term. □

Definition 19. A reduction sequence $M \twoheadrightarrow_\beta N$ is a *superdevelopment* if there exists an initial labelling L such that $M^L \twoheadrightarrow_{\beta_I} N^{L'}$ is a β_I-reduction sequence to normal form (with L' some labelling).

The following proposition states that no β_I-redexes are created by substitution. With the 'pattern' of a redex, the application and lambda symbol on top are meant.

Proposition 20. *If $\mathsf{Ap}^X(\lambda_i x.P, Q) \in \Lambda_I$ is a good term and a β_I-redex, then all patterns of β_I-redexes in $P[x := Q]$ descend either totally from P or totally from Q.*

PROOF. Suppose $\mathsf{Ap}^X(\lambda_i x.P, Q)$ is a good term with $i \in X$ and we have in $P[x := Q]$ a subterm of the form $\mathsf{Ap}^Y(\lambda_j y.R, S)$. If the symbol Ap with label Y originates from P and λ_j from Q, then $j \notin Y$, because $\mathsf{Ap}^X(\lambda_i x.P, Q)$ is a good term. So in that case $\mathsf{Ap}^Y(\lambda_j y.R, S)$ is not a β_I-redex. It is impossible to have in $P[x := Q]$ a subterm $\mathsf{Ap}^Y(\lambda_j y.R, S)$ with Ap^Y originating from Q and λ_j from P. So if $\mathsf{Ap}^Y(\lambda_j y.R, S)$ is a β_I-redex in $P[x := Q]$, then Ap^Y and λ_j originate either both from P or both from Q. □

Theorem 21. (FINITE SUPERDEVELOPMENTS) *If a λ-term M is labelled by an initial labelling L then all its β_I-reductions are finite.*

PROOF. Suppose infinite β_I-reduction sequences exist, and let M be a minimal (with respect to the number of symbols) λ-term, labelled by an initial labelling, that admits an infinite β_I-reduction sequence. By minimality M has to be an application, so M is of the form $\mathsf{Ap}^X(M_1, M_2)$. The infinite β_I-reduction sequence starting with M then must be of the form

$$\mathsf{Ap}^X(M_1, M_2) \to_{\beta_I} \mathsf{Ap}^X(\lambda_i x.M_1', M_2') \to_{\beta_I} M_1'[x := M_2'] \to_{\beta_I} \dots$$

In this reduction sequence, we have $M_1 \twoheadrightarrow_{\beta_I} \lambda_i x.M_1'$ and $M_2 \twoheadrightarrow_{\beta_I} M_2'$, and moreover $i \in X$. Turn M_1' into a context $C[,\dots,]$ by replacing all free occurrences of x by $\Box$. So $C[x,\dots,x] = M_1'$. The last step in the reduction sequence above can now be written as $\mathsf{Ap}^X(\lambda_i x.C[x,\dots,x], M_2') \to_{\beta_I} C[M_2',\dots,M_2']$ Now we claim that all reducts of this reduction sequence are of the form $C'[M_{21}'',\dots,M_{2n}'']$ with $C[,\dots,] \twoheadrightarrow_{\beta_I} C'[,\dots,]$ and $M_2' \twoheadrightarrow_{\beta_I} M_{2i}''$ for $i = 1,\dots,n$. So all reductions take place either in descendants of $C[,\dots,]$ or in descendants of M_2'. The claim follows from proposition 20 and the observation that nothing can be substituted into a descendant of M_2'. From the claim it follows immediately that either M_1 or M_2 admits an infinite reduction sequence, contradicting the minimality of M. □

Definition 22. A reduction relation $\rightarrow$ is called *weakly confluent* if for every two coinitial one-step reductions a common reduct can be found. So if $s \rightarrow t$ and $s \rightarrow u$ then a term v exists such that $t \twoheadrightarrow v$ and $u \twoheadrightarrow v$.

In exactly the same way as weak confluence for λ-calculus with β-reduction is obtained one obtains (by checking that the labels match) the same result for β_I-reduction on labelled λ-terms. Together with the property that all β_I-reduction sequences are finite we can conclude by Newman's Lemma that β_I-reduction is confluent. So each term of Λ_I has a unique β_I-normal form. Confluence for Λ_I with β_I-reduction can also be obtained by noting that it is an orthogonal CRS.

Proposition 23. *Let P and Q be good terms of Λ_I that are disjointly labelled. If $P \twoheadrightarrow_{\beta_I} P'$ and $Q \twoheadrightarrow_{\beta_I} Q'$ are β_I-reductions to normal form, then $P[x := Q] \twoheadrightarrow_{\beta_I} P'[x := Q']$ and $P'[x := Q']$ is a β_I-normal form.*

PROOF. The proof proceeds by induction on the structure of P.

- If P is a variable then the statement follows trivially.
- If $P = \lambda_i y.P_1$, then a β_I-reduction sequence of P to its normal form P' is of the form $\lambda_i y.P_1 \twoheadrightarrow_{\beta_I} \lambda_i y.P_1'$ with $P_1 \twoheadrightarrow_{\beta_I} P_1'$ a reduction to normal form. By induction hypothesis, $P_1[x := Q] \twoheadrightarrow_{\beta_I} P_1'[x := Q']$ is a reduction to normal form. Since $(\lambda_i y.P_1)[x := Q] = \lambda_i y.P_1[x := Q]$ it follows that $P[x := Q] \twoheadrightarrow_{\beta_I} P'[x := Q']$ is a reduction sequence to normal form.
- The last case to be considered is when $P = \mathsf{Ap}^X(P_1, P_2)$. Let $P_1 \twoheadrightarrow_{\beta_I} P_1'$ and $P_2 \twoheadrightarrow_{\beta_I} P_2'$ be reduction sequences to normal form. We distinguish two cases. First we consider the case that the reduction of P to normal form is of the form $\mathsf{Ap}^X(P_1, P_2) \twoheadrightarrow_{\beta_I} \mathsf{Ap}^X(P_1', P_2')$. By induction hypothesis we have that $P_1[x := Q] \twoheadrightarrow_{\beta_I} P_1'[x := Q']$ and $P_2[x := Q] \twoheadrightarrow_{\beta_I} P_2'[x := Q']$ are reduction sequences to normal form. We have $\mathsf{Ap}^X(P_1, P_2)[x := Q] \twoheadrightarrow_{\beta_I} \mathsf{Ap}^X(P_1', P_2')[x := Q']$. The result is in β_I-normal form, because even if $P'[x := Q']$ has a λ as head symbol this doesn't yield a β_I-redex. Namely, if this λ originates from P_1', its label is not contained in X (otherwise $\mathsf{Ap}^X(P_1', P_2')$ wouldn't be a β_I-normal form), and if the λ originates from Q' it doesn't yield a β_I-redex either since P and Q are disjointly labelled.
  In the second case $\mathsf{Ap}^X(P_1', P_2') = \mathsf{Ap}^X(\lambda_i y.P_{11}', P_2')$ is a β_I-redex with contractum $P_{11}'[y := P_2']$. Since $\lambda_i y.P_{11}'$ and P_2' are in β_I-normal form, the term $P_{11}'[y := P_2']$ is by proposition 20 in β_I-normal form. By induction hypothesis we have that $P_1[x := Q] \twoheadrightarrow_{\beta_I} \lambda_i y.P_{11}'[x := Q']$ and $P_2[x := Q] \twoheadrightarrow_{\beta_I} P_2'[x := Q']$ are reduction sequences to normal form. We have $\mathsf{Ap}^X(P_1, P_2)[x := Q] \twoheadrightarrow_{\beta_I} \mathsf{Ap}^X(\lambda_i y.P_{11}'[x := Q'], P_2'[x := Q']) \rightarrow_{\beta_I} (P_{11}'[x := Q'])[y := P_2'[x := Q']]$. By the substitution lemma of λ-calculus, which holds as well for the labelled case, this term equals $(P_{11}'[y := P_2'])[x := Q']$ which is by Proposition 20 a β_I-normal form.

□

This proposition yields, together with the property of unique normal forms, that if the term $\mathsf{Ap}^X(\lambda_i x.P, Q)$ is good and a β_I-redex, and its reduct $P[x := Q]$ reduces to a β_I-normal form M, then M is of the form $P'[x := Q']$, with P' and Q' the normal forms of P and Q respectively.

Theorem 24. *If $M \geq M'$, then there exists an initial labelling L such that $M^L \twoheadrightarrow_{\beta_I} M'^{L'}$ for some labelling L' and $M'^{L'}$ is a β_I-normal form.*

PROOF. The proof proceeds by induction on the derivation of $M \geq M'$. The first two easy steps are omitted.

- If $M \geq M'$ is $\mathsf{Ap}^X(M_1, M_2) \geq \mathsf{Ap}^X(M_1', M_2')$ with $M_1 \geq M_1'$ and $M_2 \geq M_2'$, then by induction hypothesis labellings L_1 and L_2 exist such that $M_1^{L_1} \twoheadrightarrow_{\beta_I} M_1'^{L_1'}$ and $M_2^{L_2} \twoheadrightarrow_{\beta_I} M_2'^{L_2'}$ are reductions to β_I-normal form. Without loss of generality we can suppose L_1 and L_2 to be disjoint. Take as labelling L for $\mathsf{Ap}^X(M_1, M_2)$ the union of L_1 and L_2 extended by assigning $\emptyset$ to the head-symbol Ap. Then $\mathsf{Ap}^\emptyset(M_1^{L_1}, M_2^{L_2}) \twoheadrightarrow_{\beta_I} \mathsf{Ap}^\emptyset(M_1'^{L_1'}, M_2'^{L_2'})$ is a reduction sequence to normal form.
- If $M \geq M'$ is due to the fourth clause of the definition of $\geq$, then $M = \mathsf{Ap}(M_1, M_2)$ and $M' = M_1'[x := M_2']$ with $M_1 \geq \lambda x.M_1'$ and $M_2 \geq M_2'$. By induction hypothesis, labelling L_1 and L_2 exist such that $M_1^{L_1} \twoheadrightarrow_{\beta_I} (\lambda x.M_1')^{L_1'}$ and $M_2^{L_2} \twoheadrightarrow_{\beta_I} M_2'^{L_2'}$ are reduction sequences to normal form. Again, L_1 and L_2 are supposed to be disjoint. Define an initial labelling L as the union of L_1 and L_2 extended by assigning $\{i\}$ to the head-symbol Ap if i is the label assigned by L_1' to the head-symbol λ of $\lambda x.M_1'$. Then

  $$M^L = \mathsf{Ap}^{\{i\}}(M_1^{L_1}, M_2^{L_2}) \twoheadrightarrow_{\beta_I} \mathsf{Ap}^{\{i\}}((\lambda x.M_1')^{L_1'}, M_2'^{L_2'}) \twoheadrightarrow_{\beta_I} M_1'^{L_1''}[x := M_2'^{L_2'}]$$

  The result of this reduction sequence is in β_I-normal form by Proposition 20. □

Theorem 25. *If $M \in \Lambda_I$ is a good term and $M \twoheadrightarrow_{\beta_I} M'$ is a β_I-reduction sequence to normal form, then $E(M) \geq E(M')$.*

PROOF. The proof proceeds by induction on the structure of M. We omit the first two steps which are trivial. If $M = \mathsf{Ap}^X(M_1, M_2)$, two possibilities have to be distinguished. First the case is considered that the reduction of M to its normal form M' is of the form $\mathsf{Ap}^X(M_1, M_2) \twoheadrightarrow_{\beta_I} \mathsf{Ap}^X(M_1', M_2')$ with $M_1 \twoheadrightarrow_{\beta_I} M_1'$ and $M_2 \twoheadrightarrow_{\beta_I} M_2'$ β_I-reductions to normal form. By induction hypothesis, we have that $E(M_1) \geq E(M_1')$ and $E(M_2) \geq E(M_2')$. This yields, applying the third clause of the definition of $\geq$, that $\mathsf{Ap}(E(M_1), E(M_2)) \geq \mathsf{Ap}(E(M_1'), E(M_2'))$, or $E(M) \geq E(M')$.

Second we consider the case that the reduction sequence of M is of the form $\mathsf{Ap}^X(M_1, M_2) \twoheadrightarrow_{\beta_I} \mathsf{Ap}^X(\lambda_i x.M_1', M_2') \twoheadrightarrow_{\beta_I} M_1'[x := M_2'] \twoheadrightarrow_{\beta_I} M'$. By Proposition 23, M' is of the form $M_1''[x := M_2'']$ with M_1'' and M_2'' the normal forms of M_1' and M_2' respectively. Then $M_1 \twoheadrightarrow_{\beta_I} \lambda_i x.M_1''$ and $M_2 \twoheadrightarrow M_2''$ are β_I-reduction sequences to normal form. By induction hypothesis, we have $E(M_1) \geq E(\lambda_i x.M_1'')$ and $E(M_2) \geq E(M_2'')$. By the fourth clause of the definition of $\geq$, we have $E(M) \geq E(M_1'')[x := E(M_2'')] = E(M_1''[x := M_2'']) = E(M')$ □

Corollary 26. *$M \geq N$ if and only if there exists a superdevelopment $M \twoheadrightarrow N$.*

Acknowledgements

I am very grateful to my supervisor Jan Willem Klop for introducing me to the subject of higher-order rewriting. Special thanks to Vincent van Oostrom for several examples and counterexamples. I would like to thank Aart Middeldorp, Tobias Nipkow and Fer-Jan de Vries for comments on earlier versions of this paper. The paper benefitted from the remarks of the anonymous referees.

References

1. P. Aczel. A general Church-Rosser theorem. Technical report, University of Manchester, 1978.
2. H.P. Barendregt. *The Lambda Calculus, its Syntax and Semantics.* North Holland, second edition, 1984.
3. R. Hindley. The equivalence of complete reductions. *Transactions of the American Mathematical Society*, 229:227–248, 1977.
4. S. Kahrs. *λ-rewriting.* PhD thesis, Universität Bremen, 1991.
5. Z. Khasidashvili. Church-Rosser theorem in orthogonal combinatory reduction systems. Technical Report 1824, INRIA Rocquencourt, 1992.
6. J.W. Klop. *Combinatory Reduction Systems.* Mathematical Centre Tracts Nr. 127. CWI, Amsterdam, 1980. PhD Thesis.
7. J.-J. Lévy. *Réductions correctes et optimales dans le lambda-calcul.* PhD thesis, Université de Paris VII, 1978.
8. T. Nipkow. Orthogonal higher-order rewrite systems are confluent. In M. Bezem and J.F. Groote, editors, *Proceedings of the International Conference on Typed Lambda Calculi and Applications*, pages 306–317, Utrecht, 1993. Springer LNCS 664.
9. M.J. O'Donnell. *Computing in Systems described by Equations.* Lecture Notes in Computer Science 58. Springer-Verlag, 1977.
10. B.K. Rosen. Tree-manipulating systems and Church-Rosser theorems. *JACM*, 20(1):160–187, 1973.
11. M. Takahashi. λ-calculi with conditional rules. In M. Bezem and J.F. Groote, editors, *Proceedings of the International Conference on Typed Lambda Calculi and Applications*, pages 306–317, Utrecht, 1993. Springer LNCS 664.