

Higher-order rewrite systems and their confluence

Richard Mayr, Tobias Nipkow*

Institut für Informatik, Technische Universität München, D-80290 München, Germany

Abstract

We study higher-order rewrite systems (HRSs) which extend term rewriting to λ -terms. HRSs can describe computations over terms with bound variables. We show that rewriting with HRSs is closely related to undirected equational reasoning. We define pattern rewrite systems (PRSs) as a special case of HRSs and extend three confluence results from term rewriting to PRSs: the critical pair lemma by Knuth and Bendix, confluence of rewriting modulo equations à la Huet, and confluence of orthogonal PRSs.

1. Introduction

Much effort has gone into the study of first-order rewrite systems and as a result there is a large body of knowledge about their properties. In 1972, Knuth and Bendix published their seminal paper [19] which shows that confluence of terminating term-rewriting systems is decidable: a simple test of confluence for the finite set of so-called critical pairs suffices. Later Huet [11] gave the definitive formulation of this result and extended it in several directions, including confluence for term-rewriting modulo certain equational theories.

The objective of this paper is to generalize some of these results from first-order rewrite systems (usually referred to as term-rewriting systems), where all functions are first-order, to rewrite systems over simply typed λ -terms. The aim of this generalization is to lift the rich theory developed around first-order rewrite systems and apply it to systems manipulating higher-order terms, such as program transformers, theorem provers and the like. In particular, this paper can be seen as an investigation of (a fragment of) the meta-theory of theorem provers like HOL [9] and Isabelle [33]: both systems are based on the simply typed λ -terms and their logic contains equality. On a more practical level, the study of higher-order rewriting has lead to the development of a new and improved unification algorithm for a certain subclass of λ -terms [26], which is now part of Isabelle. Isabelle's rewrite engine is based completely on a particular

* Corresponding author. E-mail: nipkow@informatik.tu-muenchen.de. Research supported by ESPRIT BRA 6453, Types, and WG 6028, CCL.

form of higher-order rewriting, namely the pattern rewrite systems introduced below, which is an extremely useful tool for the manipulation of terms with bound variables.

We study two kinds of rewrite systems: *pattern rewrite systems* (PRSs) and the more general *higher-order rewrite systems* (HRSs). PRSs are similar to Klop's *combinatory reduction systems* (CRSs) [16, 18]. Both are generalizations of *term-rewriting systems* (TRSs) [5] to terms with higher-order functions and bound variables. The main difference is that PRSs use the typed λ -calculus as a meta-language, whereas CRSs come with their own untyped abstraction and substitution mechanism. The precise relationship between the two formalisms is explored elsewhere [30]: it is shown that a CRS can be simulated directly by a PRS, whereas the reverse simulation is quite involved. As a consequence, results like confluence carry over very easily from PRSs to CRSs, but not so easily in the other direction. Although there are these technical differences, the abstraction mechanism in both PRSs and CRSs is general enough to represent quantification in formulae, abstraction in functional programs, and many other variable-binding constructs. Using this representation, many operations on formulae and programs can be expressed naturally as higher-order rewrite systems.

In Section 2 we review the terminology and notation of the typed λ -calculus which is used to define object-level rewrite systems, and define some basic properties. In Section 3 we define higher-order rewrite systems (HRSs), pattern rewrite systems (PRSs), the reduction relation they induce on terms and show how they interact with substitutions. Then we show how HRSs induce an equality on terms and how it relates to reduction. In Section 4 the Critical Pair Lemma from Knuth and Bendix is generalized to PRSs. In Section 5 a theorem due to Huet about confluence modulo equality is generalized to PRSs. In Section 6 we deal with orthogonal pattern rewrite systems (OPRSs). These are a special kind of PRSs that have no critical pairs and whose rewrite rules are all left linear. In this section we give two different proofs that OPRSs are confluent. The paper closes with a discussion of related work.

2. Preliminaries

What follows is a description of the meta-language of simply typed λ -calculus which is used to define object-level rewrite systems. The notation is roughly consistent with the standard literature [4, 10].

Starting with some fixed set of *base types* \mathcal{B} the set of all *types* \mathcal{T} is the closure of \mathcal{B} under the function space constructor \rightarrow . The letter τ is used to denote types. Function types associate to the right: $\tau_1 \rightarrow \tau_2 \rightarrow \tau_3$ means $\tau_1 \rightarrow (\tau_2 \rightarrow \tau_3)$. Instead of $\tau_1 \rightarrow \dots \rightarrow \tau_n \rightarrow \tau$ we also write $\overline{\tau_n} \rightarrow \tau$, if τ is a base type.

Terms are generated from a set of typed *variables* $V = \bigcup_{\tau \in \mathcal{T}} V_\tau$ and a set of typed *constants* $C = \bigcup_{\tau \in \mathcal{T}} C_\tau$, where $V_\tau \cap V_{\tau'} = C_\tau \cap C_{\tau'} = \{\}$ if $\tau \neq \tau'$, by λ -abstraction and application. Arbitrary variables are denoted by x, y and z , free variables by upper case letters F, G and H , and constants by c, d, f and g . *Atoms* are constants or variables and are denoted by a and b . Terms are denoted by l, r, s, t and u . We write $t : \tau$ to

indicate that a term t is of type τ . The inductive definition of *simply typed* λ -terms is as follows:

$$\frac{x \in V_\tau}{x : \tau} \quad \frac{c \in C_\tau}{c : \tau} \quad \frac{s : \tau \rightarrow \tau' \quad t : \tau}{(s \ t) : \tau'} \quad \frac{x : \tau \quad s : \tau'}{(\lambda x. s) : \tau \rightarrow \tau'}$$

In the sequel all our λ -terms are assumed to be simply typed.

Instead of $\lambda x_1 \dots \lambda x_n. s$ we also write $\lambda x_1, \dots, x_n. s$ or just $\lambda \overline{x_n}. s$, where the x_i are assumed to be distinct. Similarly instead of $(\dots (t \ u_1) \dots) u_n$ we write $t(u_1, \dots, u_n)$ or just $t(\overline{u_n})$. The notation $t(\overline{u_n})$ includes the possibility $n=0$ if t is of base type. The *free* and *bound* variables occurring in a term s are denoted by $fv(s)$ and $bv(s)$, respectively. A term is called *linear* iff no free variable occurs in it more than once.

We assume the usual definition of α, β and η conversion between λ -terms. We write $s =_\gamma t$, where $\gamma \in \{\alpha, \beta, \eta\}$ if s and t are equivalent modulo γ -conversion. In the sequel α -equivalent terms are identified.

As the simply typed λ -calculus is confluent and terminating w.r.t. β -reduction (η -reduction) every term t has a β -normal form (η -normal form) which is denoted $t \downarrow_\beta$ ($t \downarrow_\eta$). Let t be in β -normal form. Then t is of the form $\lambda \overline{x_k}. a(\overline{u_m})$, where a is called the *head* of t . The η -expanded form of t is defined by

$$t \uparrow^\eta = \lambda \overline{x_{n+k}}. a(\overline{u_m} \uparrow^\eta, x_{n+1} \uparrow^\eta, \dots, x_{n+k} \uparrow^\eta)$$

where $t : \overline{x_{n+k}} \rightarrow \tau$ and $x_{n+1}, \dots, x_{n+k} \notin fv(\overline{u_m})$. Instead of $t \downarrow_\beta \uparrow^\eta$ we write $t \uparrow_\beta^\eta$. A λ -term t is in *long $\beta\eta$ -normal form* iff $t = t \uparrow_\beta^\eta$.

Convention: Unless stated otherwise, the variables r, s, t , etc., range over λ -terms in long $\beta\eta$ -normal form.

Terms can also be viewed as trees. Subterms can be numbered by so-called *positions* which are the paths from the root to the subterm in Dewey decimal notation. Details can be found in [11, 5]. We just briefly review the notation. The *positions* in a term t are denoted by $\mathcal{Pos}(t) \subseteq \mathbb{N}^*$. The letters p and q stand for positions. The root position is ε , the empty sequence. Two positions p and q are appended by juxtaposing them: pq . The letter i is reserved for natural numbers and $i.p$ is the position obtained by appending i to the front of p . Given $p \in \mathcal{Pos}(t)$, t/p is the subterm of t at position p ; $t[u]_p$ is t with t/p replaced by u .

Abstractions and applications yield the following trees:

$$\begin{array}{c} \lambda x \\ | \quad \backslash \\ t \quad t_1 \quad t_2 \end{array}$$

Formally, this is defined as follows:

$$t/\varepsilon = t$$

$$(t_1 \ t_2)/(i.p) = t_i/p$$

$$(\lambda x. t)/(1.p) = t/p$$

Hence, positions in λ -terms are sequences over $\{1, 2\}$. Note that the bound variable in an abstraction is not a separate subterm and can therefore not be accessed by the s/p notation. For $p_1, p_2 \in \text{Pos}(t)$ we define the partial ordering $p_1 \leq p_2$ iff p_1 is a prefix of p_2 . We write $p_1 \parallel p_2$ iff neither $p_1 \leq p_2$ nor $p_2 \leq p_1$. If $p_1 \leq p_2$, the position $p_2/p_1 \in \text{Pos}(t/p_1)$ is defined as p_2 without the prefix p_1 .

Substitutions are finite mappings from variables to terms of the same type. Substitutions are denoted by θ, σ and δ . For $\theta = \{x_1 \mapsto t_1, \dots, x_n \mapsto t_n\}$ we define $\text{Dom}(\theta) = \{x_1, \dots, x_n\}$ and $\text{Cod}(\theta) = \{t_1, \dots, t_n\}$. The application of a substitution to a term is defined by $\theta(t) := (\lambda \bar{x}_k. t)(\bar{t}_n) \uparrow_\beta^\eta$. We often drop the parentheses and simply write θt .

Let θ_1 and θ_2 be substitutions. Then $\theta_1 + \theta_2$ is a substitution with

$$\text{Dom}(\theta_1 + \theta_2) := \text{Dom}(\theta_1) \cup \text{Dom}(\theta_2)$$

defined by

$$(\theta_1 + \theta_2)(F) := \begin{cases} \theta_2(F) & \text{if } F \in \text{Dom}(\theta_2), \\ \theta_1(F) & \text{otherwise.} \end{cases}$$

A *renaming* ρ is an injective substitution with $\text{Cod}(\rho) \subset V$ and $\text{Dom}(\rho) \cap \text{Cod}(\rho) = \{\}$. Renamings are always denoted by ρ .

Note that we will always assume that the domain of a substitution does not contain any variable bound in a term the substitution is applied to. If necessary, the bound variables are renamed automatically.

Two terms s and t are called *unifiable* iff there is a substitution θ , such that $\theta(s) = \theta(t)$. The term s *matches* the term t iff there is a substitution θ , such that $\theta(s) = t$. The problem to decide if a term s matches a term t and to compute the substitution θ is called the *matching problem*, which is very important for rewriting (see the remarks after Definition 3.3).

Given $p \in \mathcal{P}\text{os}(t)$, $bv(t, p)$ is the list of all λ -abstracted variables on the path from the root of t to p :

$$bv(t, \epsilon) = []$$

$$bv((t_1 \ t_2), i, p) = bv(t_i, p)$$

$$bv(\lambda x. t, 1, p) = x . bv(t, p)$$

It is frequently necessary to “lift” a term into a context of certain bound variables. An \bar{x}_k -*lifter* of a term t away from W is a substitution $\sigma = \{F \mapsto (\rho F)(\bar{x}_k) \mid F \in fv(t)\}$ where ρ is a renaming such that $\text{Dom}(\rho) = fv(t)$, $\text{Cod}(\rho) \cap W = \{\}$ and $\rho F : \tau_1 \rightarrow \dots \rightarrow \tau_k \rightarrow \tau$ if $x_1 : \tau_1, \dots, x_k : \tau_k$ and $F : \tau$.

For example, $\sigma = \{F \mapsto G(x), S \mapsto T(x)\}$ is an x -lifter of $f(\lambda y. g(F(y)), S)$ away from any W not containing G or T ; the corresponding renaming is $\rho = \{F \mapsto G, S \mapsto T\}$.

Definition 2.1. We define the *order* of a type in the traditional way:

$$\text{ord}(\tau) = 1 \quad \text{if } \tau \text{ is a base type,}$$

$$\text{ord}(\tau_0 \rightarrow \tau_1) = \max\{\text{ord}(\tau_0) + 1, \text{ord}(\tau_1)\}.$$

The *order* of a substitution is the maximal order of the types of the variables in its domain:

$$\text{ord}(\theta) = \max\{\text{ord}(\tau) \mid V_\tau \cap \text{Dom}(\theta) \neq \{\}\}$$

Note that $\text{ord}(\{\})$, i.e. $\max\{\}$ is defined to be 0.

Given a relation \rightarrow , $\stackrel{=}{\rightarrow}$ denotes the reflexive closure, \leftrightarrow the symmetric closure and $\stackrel{*}{\rightarrow}$ the transitive and reflexive closure of \rightarrow . We write $s \downarrow t$ iff there is a u , such that $s \stackrel{*}{\rightarrow} u$ and $t \stackrel{*}{\rightarrow} u$. The relation \rightarrow is (*locally*) *confluent* if $r \stackrel{*}{\rightarrow} s$ ($r \rightarrow s$) and $r \stackrel{*}{\rightarrow} t$ ($r \rightarrow t$) imply $s \downarrow t$. The relation \rightarrow is *terminating* if there is no infinite sequence $s_i \rightarrow s_{i+1}$ for all $i \in \mathbb{N}$. It is well known that terminating relations are confluent iff they are locally confluent [11].

3. Higher-order rewrite systems

Higher-order rewrite systems are generalizations of first-order rewrite systems [5] to terms with higher-order functions and bound variables. Since unifiability of λ -terms is undecidable in general [8], we often restrict to a certain subclass of λ -terms which behave very much like first-order terms w.r.t. unification.

Definition 3.1. A λ -term t in β -normal form is called a (higher-order) *pattern* if every free occurrence of a variable F is in a subterm $F(\bar{u}_n)$ of t , such that \bar{u}_n is η -equivalent to a list of *distinct* bound variables.

Examples of higher-order patterns are $\lambda x. c(x)$, F , $\lambda x. F(\lambda z. x(z))$ and $\lambda x, y. F(y, x)$. Examples of non-patterns are $F(c)$, $\lambda x. F(x, x)$, $\lambda x, y. F(y, c)$ and $\lambda x. G(H(x))$.

The following crucial result about unification of patterns is due to Dale Miller [23]:

Theorem 3.2. *It is decidable whether two patterns are unifiable; if they are unifiable, a most general unifier can be computed.*

Nipkow [26] presents a simplified form of Miller's unification algorithm and develops it towards a practical implementation. Qian [36] shows that patterns can be unified in linear time.

Definition 3.3. A *rewrite rule* is a pair $l \rightarrow r$ such that l is not a free variable, l and r are of the same *base type*, and $\text{fv}(l) \supseteq \text{fv}(r)$. A *pattern rewrite rule* is a rewrite rule whose left-hand side is a pattern. A *higher-order rewrite system* (HRS) is a set of rewrite rules. A *pattern rewrite system* (PRS) is a set of pattern rewrite rules. The letter R always denotes an HRS (which will often be a PRS). A HRS R induces a relation \rightarrow_R on terms:

$$s \rightarrow_R t \Leftrightarrow \exists (l \rightarrow r) \in R, \quad p \in \mathcal{Pos}(s), \quad \theta. s/p = \theta l \wedge t = s[\theta r]_p.$$

A rewrite rule is called *left-linear* iff its left-hand side is linear. A HRS is called *left-linear* iff all its rewrite rules are left-linear.

Our HRSs are exactly Wolfram’s “higher-order term rewriting systems” [41].

A few remarks are in order:

- Recall that by convention l , r , s and t are in long $\beta\eta$ -normal form.
- Due to the restrictions placed on left-hand sides, they must always be of the form $c(\overline{s_n})$.
- For PRSs, the restriction $fv(l) \supseteq fv(r)$ is preserved under substitution. Thus it has the same effect as for TRSs: rewriting cannot introduce new variables. This fails for general HRSs: although $f(F(X)) \rightarrow g(X)$ meets the restriction, replacing F by $\lambda x. Y$ yields $f(Y) \rightarrow g(X)$ which violates the restriction. This problem is investigated in more detail by Kahrs [15].

Since we derive only very basic results about HRSs, we do not need to impose anything like $fv(l) \supseteq fv(r)$, whereas the latter is essential for the confluence results for PRSs.

- The restriction to rules of base type is necessary because of the simple matching procedure inherent in the definition of \rightarrow_R . Otherwise $\overset{*}{\leftrightarrow}_R$ and $=_R$ do not coincide (see Section 3.1). Pulling rules down to base type by applying them to new variables is fine for HRSs but may fail for PRSs: $\lambda x. c(F(x))$ is a pattern but ceases to be one when pulled down to base type: $c(F(X))$.
- \rightarrow_R is defined only between terms in long $\beta\eta$ -normal form. This simplifies technicalities. Alternatively, we could work with a relation \rightsquigarrow_R defined by $s \rightsquigarrow_R t \Leftrightarrow s \downarrow_\beta^\eta \rightarrow_R t \uparrow_\beta^\eta$.
- The relation \rightarrow_R is decidable if the matching problem is decidable for the left-hand sides of the rules in R . As by Theorem 3.2 even unifiability is decidable for patterns, the relation \rightarrow_R is decidable for any PRS R ; full unification will come in handy in connection with critical pairs.

For general HRSs the situation is different. Although unifiability is known to be undecidable even for second order λ -terms [8], it is not known whether higher-order matching is decidable. Padovani [32] proved that fourth-order matching is decidable, but the general case is still open.

Example 3.4. The standard example of a PRS is pure lambda-calculus itself. The syntax involves just the type *term* of terms and two constants for abstraction and application:

abs: $(term \rightarrow term) \rightarrow term$

app: $term \rightarrow term \rightarrow term$

The rewrite rules are

beta: $app(abs(\lambda x. F(x)), S) \rightarrow F(S)$

eta: $abs(\lambda x. app(S, x)) \rightarrow S$

Note how the use of meta-level application and abstraction removes the need for a substitution operator (in the *beta*-rule) and side conditions (in the *eta*-rule).

The following lemma is a simple consequence of the fact that all rewrite rules must be of base type:

Lemma 3.5. *If R is an HRS and $\lambda x.s \rightarrow_R t$ then $t = \lambda x.u$ and $s \rightarrow_R u$ for some u .*

In the sequel it will be convenient to have an inference-rule based formulation of rewriting.

Definition 3.6. Given an HRS R , let \Rightarrow_R be the least relation on terms in long $\beta\eta$ -normal form which is closed under the following rules:

$$\frac{(l \rightarrow r) \in R}{\theta l \Rightarrow_R \theta r} \quad \frac{s \Rightarrow_R t}{a(\overline{s_m}, s, \overline{u_n}) \Rightarrow_R a(\overline{s_m}, t, \overline{u_n})} \quad \frac{s \Rightarrow_R t}{\lambda x.s \Rightarrow_R \lambda x.t}$$

where a is an atom of type $\overline{\tau_{m+1+n}} \rightarrow \tau$.

Of course, the two definitions of rewriting are equivalent.

Lemma 3.7. *If R is an HRS then \rightarrow_R and \Rightarrow_R coincide.*

Proof. The containment $\Rightarrow_R \subseteq \rightarrow_R$ is shown by induction on the structure of \Rightarrow_R , the reverse containment by induction on the length of p in the definition of \rightarrow_R . \square

In the sequel, we will not distinguish \rightarrow_R and \Rightarrow_R and use whichever definition is most appropriate. In addition, we usually drop the subscript R and simply write \rightarrow .

We will now prove an important theorem about $\xrightarrow{*}$, namely its stability under substitution. For TRSs this is simple and one obtains the stronger result that $s \rightarrow t$ implies $\theta s \rightarrow \theta t$. For HRSs we have to replace \rightarrow by $\xrightarrow{*}$ because β -reductions during the application of a substitution can copy redexes. These copies need to be reduced sequentially; hence the use of $\xrightarrow{*}$ instead of \rightarrow . Later on we will see that a suitable notion of parallel reduction leads to a much nicer stability result (Lemma 6.4).

Definition 3.8. Let \rightarrow be an arbitrary relation on terms. We define $\theta \rightarrow \theta'$ to mean that $\theta(F) \rightarrow \theta'(F)$ holds for all $F \in \text{Dom}(\theta)$.

Theorem 3.9. *Let R be an HRS. If $s \xrightarrow{*} s'$ and $\theta \xrightarrow{*} \theta'$ then $\theta s \xrightarrow{*} \theta' s'$.*

Proof. By induction on the order of θ (see Definition 2.1) with a nested induction on the length of the derivation $s \xrightarrow{*} s'$.

1. If $s = s'$, we prove $\theta s \xrightarrow{*} \theta' s$ by induction on the structure of s . The structure of normal forms dictates that $s = \lambda \overline{x_m}. a(\overline{s_n})$. The innermost induction hypothesis implies $\theta s_i \xrightarrow{*} \theta' s_i$. Now we distinguish two cases.

- (a) If $a \notin \mathcal{D}om(\theta)$ (and hence $a \notin \mathcal{D}om(\theta')$ because the left-hand side of a rewrite rule cannot match a variable) then $\theta s = \lambda \bar{x}_m. a(\bar{\theta s}_n) \xrightarrow{*} \lambda \bar{x}_m. a(\overline{\theta' s}_n) = \theta' s$ follows easily.
- (b) If $a \in \mathcal{D}om(\theta)$ then $\theta a = \lambda \bar{y}_n. t$. By Lemma 3.5 it follows that $\theta' a = \lambda \bar{y}_n. t'$ where $t \xrightarrow{*} t'$. Define the substitutions $\delta = \{y_n \mapsto \bar{\theta s}_n\}$ and $\delta' = \{y_n \mapsto \overline{\theta' s}_n\}$ and notice that we have $\delta \xrightarrow{*} \delta'$. If a is of type $\bar{\tau}_n \rightarrow \tau$, the definition of *ord* implies $ord(\tau_i) < ord(\bar{\tau}_n \rightarrow \tau)$ and hence $ord(\delta) < ord(\theta)$ (note that this also holds in case $n = 0$). Thus the outermost induction hypothesis applies: $\delta t \xrightarrow{*} \delta' t'$ because $t \xrightarrow{*} t'$. Thus we obtain: $\theta s = \lambda \bar{x}_m. (\theta a)(\bar{\theta s}_n) \downarrow_\beta = \lambda \bar{x}_m. (\delta t) \xrightarrow{*} \lambda \bar{x}_m. (\delta' t') = \lambda \bar{x}_m. (\theta' a)(\overline{\theta' s}_n) \downarrow_\beta = \theta' s$.
2. If $s \xrightarrow{*} s' \rightarrow s''$, the inner induction hypothesis yields $\theta s \xrightarrow{*} \theta' s'$. We show $\theta' s' \xrightarrow{*} \theta' s''$ by induction on the structure of the derivation $s' \rightarrow s''$ as in Definition 3.6.
- (a) If $s' = \delta l$ and $s'' = \delta r$ for some $(l \rightarrow r) \in R$, $\theta' s' = \theta' \delta l = (\theta' \delta) l \rightarrow (\theta' \delta) r = \theta' \delta r = \theta' s''$.
- (b) If $s' = \lambda x. t'$ and $s'' = \lambda x. t''$ such that $t' \rightarrow t''$, the innermost induction hypothesis yields $\theta' t' \rightarrow \theta' t''$ and hence $\theta' s' = \lambda x. \theta' t' \xrightarrow{*} \lambda x. \theta' t'' = \theta' s''$.
- (c) If $s' = a(\bar{s}'_n)$, $s'' = a(\bar{s}''_n)$ such that $s'_k \rightarrow s''_k$ for some k and $s'_i = s''_i$ for all $i \neq k$, then the innermost induction hypothesis yields $\theta' s'_k \xrightarrow{*} \theta' s''_k$. Because $\theta' s'_i \xrightarrow{*} \theta' s''_i$ holds trivially for all $i \neq k$, we obtain $\theta' s'_i \xrightarrow{*} \theta' s''_i$ for all i . Now we distinguish two cases.
- (i) If $a \notin \mathcal{D}om(\theta')$ then $\theta' s' = a(\overline{\theta' s}'_n) \xrightarrow{*} a(\overline{\theta' s}''_n) = \theta' s''$ follows easily.
- (ii) If $a \in \mathcal{D}om(\theta')$ then $\theta' a = \lambda \bar{y}_n. t$. Define the new substitutions $\delta' = \{y_n \mapsto \overline{\theta' s}'_n\}$ and $\delta'' = \{y_n \mapsto \overline{\theta' s}''_n\}$ and notice that we have $\delta' \xrightarrow{*} \delta''$. As above, we can show that $ord(\delta') < ord(\theta')$; because \rightarrow preserves types, we also have $ord(\theta') = ord(\theta)$. Thus the outermost induction hypothesis applies: $\delta' t \xrightarrow{*} \delta'' t$ because $t \xrightarrow{*} t$. Thus we obtain $\theta' s' = (\theta' a)(\overline{\theta' s}'_n) \downarrow_\beta = \delta' t \xrightarrow{*} \delta'' t = (\theta' a)(\overline{\theta' s}''_n) \downarrow_\beta = \theta' s''$. \square

This theorem has two obvious corollaries:

- $s \xrightarrow{*} s'$ implies $\theta s \xrightarrow{*} \theta s'$ and
- $\theta \xrightarrow{*} \theta'$ implies $\theta s \xrightarrow{*} \theta' s$.

However, the above proof fails if one reduces the statement of the theorem to one of the corollaries.

A slightly different version of Theorem 3.9 is also shown by Loría [20]: he uses conditional rewrite rules whose left-hand sides are patterns; his proof relies more on considerations about term positions.

Finally we lift Theorem 3.9 from $\xrightarrow{*}$ to \leftrightarrow^* , at least for a special case:

Corollary 3.10. *Let R be an HRS. If $s \leftrightarrow^* s'$ and $t \leftrightarrow^* t'$ then $\{x \mapsto t\} s \leftrightarrow^* \{x \mapsto t'\} s'$.*

Proof. If $s_1 \leftrightarrow s_2 \cdots \leftrightarrow s_m$ and $t_1 \leftrightarrow t_2 \cdots \leftrightarrow t_n$, let $\theta_i = \{x \mapsto t_i\}$. Now Theorem 3.9 implies $\theta_1 s_1 \xrightarrow{*} \theta_2 s_1 \xrightarrow{*} \cdots \xrightarrow{*} \theta_n s_1 \xrightarrow{*} \theta_n s_2 \xrightarrow{*} \cdots \xrightarrow{*} \theta_n s_m$. \square

The general case, i.e. if $\theta \xrightarrow{*} \theta'$ and $s \xrightarrow{*} s'$ then $\theta s \xrightarrow{*} \theta' s'$, is rather more tedious to derive and is left as an exercise.

3.1. Rewriting versus equality

Originally, term rewriting was a means of analyzing equational theories, but it has long since taken on a life of its own. Returning to those roots we need to relate our notion of rewriting to a more “logical” notion of equality. In the sequel E will always denote a set of *equations*, i.e. a set of pairs $s = t$, where s and t are terms of the same type. In particular any rewrite rule $l \rightarrow r$ can also be viewed as an equation $l = r$.

Throughout this subsection we do not assume that terms or substitutions are in any normal form and θt denotes the non-normalizing application of θ to t .

Definition 3.11. A set of equations E induces a relation $=_E$ defined by the following inference rules, which come in three groups:

Basic conversion rules:

$$\frac{(s = t) \in E}{\theta s =_E \theta t} (E) \quad \frac{s =_E t}{s =_E t} (\beta) \quad \frac{s =_\eta t}{s =_E t} (\eta)$$

Equality rules:

$$\frac{}{s =_E s} (refl) \quad \frac{s =_E t}{t =_E s} (sym) \quad \frac{r =_E s \quad s =_E t}{r =_E t} (trans)$$

Congruence rules:

$$\frac{s_1 =_E t_1 \quad s_2 =_E t_2}{(s_1 \ s_2) =_E (t_1 \ t_2)} (app) \quad \frac{s =_E t}{\lambda x. s =_E \lambda x. t} (abs)$$

We call $=_E$ the *equational theory* generated by E .

Note that θ in (E) is present only for convenience: substitution can be simulated by (abs), (app) and (β).

Modulo \downarrow_β^η we have the same relationship between \rightarrow_R and $=_R$ as in the first order case. This can be viewed as a justification of our definition of \rightarrow_R .

Theorem 3.12. If R is an HRS then $s =_R t \Leftrightarrow s \downarrow_\beta^\eta \xrightarrow{*} t \downarrow_\beta^\eta$.

Proof. The \Leftarrow -direction is easy since $=_R$ can mimic $\xrightarrow{*}$ directly. If $s \downarrow_\beta^\eta \xrightarrow{*} t \downarrow_\beta^\eta$ then $s \xrightarrow{*}_\beta s \downarrow_\beta \xrightarrow{*}_\eta s \downarrow_\beta^\eta \xrightarrow{*} t \downarrow_\beta^\eta \xrightarrow{*}_\eta t \downarrow_\beta \xrightarrow{*}_\beta t$. The reductions \rightarrow_β and \rightarrow_η are subsumed by (β) and (η); every \leftrightarrow is replaced by a single (E), possibly combined with (sym), embedded in a tree of congruence and reflexivity rules. Everything is held together with a finite amount of (trans).

For the \Rightarrow -direction assume $s =_R t$. By induction on the structure of the derivation of $s =_R t$, considering each rule in turn, we prove $s \downarrow_\beta^\eta \xleftrightarrow{*} t \downarrow_\beta^\eta$.

(E): $s = \theta l$ and $t = \theta r$ for some $(l \rightarrow r) \in R$. Thus $l \rightarrow r$ and hence $s \downarrow_\beta^\eta = (\theta l) \downarrow_\beta^\eta \xrightarrow{*} (\theta r) \downarrow_\beta^\eta = t \downarrow_\beta^\eta$ by Theorem 3.9.

(β), (η), ($refl$): trivial because $s \downarrow_\beta^\eta = t \downarrow_\beta^\eta$.

($trans$), (sym): by induction hypothesis because $\xleftrightarrow{*}$ is transitive and symmetric.

(abs): by induction hypothesis because $s' \downarrow_\beta^\eta \xleftrightarrow{*} t' \downarrow_\beta^\eta$ implies $(\lambda x. s') \downarrow_\beta^\eta \xleftrightarrow{*} (\lambda x. t') \downarrow_\beta^\eta$.

(app): $s = (s_1 s_2)$ and $t = (t_1 t_2)$. By induction hypothesis we have $s_i \downarrow_\beta^\eta \xleftrightarrow{*} t_i \downarrow_\beta^\eta$. Since s_1 and t_1 are of functional type, $s_1 \downarrow_\beta^\eta$ must be of the form $\lambda x. s'_1$ and $t_1 \downarrow_\beta^\eta$ of the form $\lambda x. t'_1$. Thus Lemma 3.5 implies $s'_1 \xleftrightarrow{*} t'_1$. Let $\theta = \{x \mapsto s_2 \downarrow_\beta^\eta\}$ and $\theta' = \{x \mapsto t_2 \downarrow_\beta^\eta\}$. Corollary 3.10 yields $s \downarrow_\beta^\eta = ((\lambda x. s'_1) s_2) \downarrow_\beta^\eta = (\theta s'_1) \downarrow_\beta^\eta \xleftrightarrow{*} (\theta' t'_1) \downarrow_\beta^\eta = ((\lambda x. t'_1) t_2) \downarrow_\beta^\eta = t \downarrow_\beta^\eta$. \square

Thus, we know that undirected rewriting and equational logic coincide. Note that the proof sketch in [25] is considerably more involved because Theorem 3.9 is not available.

Now we can use the fact that for confluent reductions convertibility and existence of a common reduct coincide:

Corollary 3.13. *If R is a confluent HRS then $s =_R t \Leftrightarrow s \downarrow_\beta^\eta \downarrow_R t \downarrow_\beta^\eta$.*

Loría [20, Theorem 5.1.2] follows [25] to prove the same result for conditional PRSs. Wolfram [41, Theorem 4.11] appears to prove the same result, but all of his equations are by definition in long $\beta\eta$ -normal form and a proof of the (app)-case of his theorem would require something like Theorem 3.9 above or Theorem 3.11 of [25].

It should be pointed out that Theorem 3.12 fails for rules of function type. The one-rule system

$$R = \{\lambda x. c(x, F(x)) \rightarrow \lambda x. d(F(x), x)\}$$

induces a relation $\xleftrightarrow{*}$ which is strictly weaker than $=_R$: $c(a, f(a)) =_R d(f(a), a)$ holds but $c(a, f(a)) \not\xleftrightarrow{*} d(f(a), a)$ does not hold because the definition of \rightarrow insists on rewriting β -normal forms only. Otherwise one could β -expand $c(f(a), a)$ to $(\lambda x. c(x, f(x)))a$ before rewriting it to $(\lambda x. d(f(x), x))a$.

4. The Critical Pair Lemma

In 1972, Knuth and Bendix [19] showed that confluence of terminating rewrite systems is decidable: a simple test of confluence for the finite set of so called critical pairs suffices. Later this result was generalized to PRSs by Nipkow [25], although no proof was given at the time. The purpose of this section is to supply the missing proof and at the same time prepare the ground for the related issue of confluence modulo equality which is treated in the following section.

Definition 4.1. Let there be two rewrite rules $l_i \rightarrow r_i$, $i = 1, 2$, in a PRS and a position $p \in \text{Pos}(l_1)$ such that

- $fv(l_1) \cap bv(l_1) = \{\}$,
- the head of l_1/p is not a free variable in l_1 , and
- the two patterns $\lambda \bar{x}_k.(l_1/p)$ and $\lambda \bar{x}_k.(\sigma l_2)$, where $\{\bar{x}_k\} = bv(l_1, p)$ and σ is an \bar{x}_k -lifter of l_2 away from $fv(l_1)$, have a most general unifier θ .

Then the pattern l_1 *overlaps* the pattern l_2 at position p . The rewrite rules determine the *critical pair* $\langle \theta r_1, \theta(l_1[\sigma r_2]_p) \rangle$. Note that because $\lambda \bar{x}_k.(l_1/p)$ and $\lambda \bar{x}_k.(\sigma l_2)$ unify, l_1/p must be of the form $f(\dots)$.

Two redexes t/p_1 and t/p_2 in a term t are *overlapping* if there are rewrite rules $l_i \rightarrow r_i$, $i = 1, 2$, such that $p_1 \leq p_2$, $t/p_1 = \theta_1 l_1$, $t/p_2 = \theta_2 l_2$ and l_1 overlaps l_2 at position p_2/p_1 .

The *critical pairs* of a PRS R are all the critical pairs arising from overlapping two left-hand sides of rules in R , except for a left-hand side of a rule overlapping itself at position ε . Note that it is possible that a left-hand side l of a rule $l \rightarrow r$ overlaps itself at positions $p \neq \varepsilon$ thus giving rise to a critical pair.

As this definition is difficult to handle, the following lemmas will be useful for dealing with critical pairs. Let us first show that critical pairs represent rewrite peaks:

Lemma 4.2. Let $\langle u_1, u_2 \rangle$ be a critical pair. Then there exists a term s such that $u_1 \leftarrow s \rightarrow u_2$.

Proof. Let $u_1 = \theta r_1$ and $u_2 = \theta(l_1[\sigma r_2]_p)$ as in Definition 4.1 and define $s := \theta l_1$. Thus $s \rightarrow \theta r_1 = u_1$ is trivial. We also have $\lambda \bar{x}_k. \theta(l_1/p) = \theta(\lambda \bar{x}_k.(l_1/p)) = \theta(\lambda \bar{x}_k.(\sigma l_2)) = \lambda \bar{x}_k.(\theta \sigma l_2)$, because $fv(l_1) \cap bv(l_1) = \{\}$. Therefore $\theta(l_1/p) = \theta \sigma l_2$. As l_1 is a pattern and l_1/p is of the form $f(\dots)$, we also have $p \in \text{Pos}(\theta l_1)$ and $(\theta l_1)/p = \theta \sigma l_2$. So we get $s = (\theta l_1)[\theta \sigma l_2]_p \rightarrow (\theta l_1)[\theta \sigma r_2]_p = \theta(l_1[\sigma r_2]_p) = u_2$. \square

Lemma 4.3. Let there be two patterns l_1, l_2 , a position $p \in \text{Pos}(l_1)$ where l_1/p is not of the form $F(\dots)$, $\{\bar{x}_k\} = bv(l_1, p)$ and an \bar{x}_k -lifter σ of l_2 away from $fv(l_1)$. Then the two patterns $\lambda \bar{x}_k.(l_1/p)$ and $\lambda \bar{x}_k.\sigma(l_2)$ are unifiable iff there exist substitutions θ_1 and θ_2 , such that $\theta_1(l_1/p) = \theta_2 l_2$ and $bv(l_1, p) \cap \text{Cod}(\theta_1) = \{\}$.

Proof. (1) For the (\Leftarrow)-direction assume w.l.o.g. $\text{Dom}(\theta_1) \subseteq fv(l_1)$. Let $\theta'_2 := \{\rho(F) \mapsto \lambda \bar{x}_k.\theta_2(F) \mid F \in fv(l_2)\}$ and ρ be the renaming corresponding to σ . We now show that $\theta_0 := \theta_1 \cup \theta'_2$ is a unifier of $\lambda \bar{x}_k.(l_1/p)$ and $\lambda \bar{x}_k.\sigma(l_2)$. From $\theta'_2 \sigma(F) = \theta'_2(\rho(F))(\bar{x}_k) = \theta_2(F)$ for all $F \in fv(l_2)$ it follows that $\theta'_2(\sigma l_2) = \theta_2 l_2$. Therefore,

$$\begin{aligned}
 \theta_0(\lambda \bar{x}_k.(l_1/p)) &= \theta_1(\lambda \bar{x}_k.(l_1/p)) & \text{Dom}(\theta'_2) \cap fv(l_1) &= \{\} \\
 &= \lambda \bar{x}_k.\theta_1(l_1/p) & bv(l_1, p) \cap \text{Cod}(\theta_1) &= \{\} \\
 &= \lambda \bar{x}_k.\theta_2(l_2) \\
 &= \lambda \bar{x}_k.\theta'_2(\sigma(l_2)) = \theta'_2(\lambda \bar{x}_k.\sigma(l_2)) & \text{Cod}(\theta'_2) \cap \{\bar{x}_k\} &= \{\} \\
 &= \theta_0(\lambda \bar{x}_k.\sigma(l_2)) & \sigma \text{ away from } fv(l_1) &\supseteq \text{Dom}(\theta_1).
 \end{aligned}$$

(2) For the (\Rightarrow)-direction assume that θ is a unifier of $\lambda\bar{x}_k.(l_1/p)$ and $\lambda\bar{x}_k.(\sigma l_2)$. Define $\theta_1 := \theta|_{fv(l_1)}$ and $\theta_2 := \theta\sigma$. Then we have $\lambda\bar{x}_k.\theta_1(l_1/p) = \theta(\lambda\bar{x}_k.(l_1/p)) = \theta(\lambda\bar{x}_k.(\sigma l_2)) = \lambda\bar{x}_k.\theta_2(l_2)$, because $fv(l_1) \cap bv(l_1) = \{\}$. Therefore $\theta_1(l_1/p) = \theta_2(l_2)$. As θ is a unifier of $\lambda\bar{x}_k.(l_1/p)$ and $\lambda\bar{x}_k.(\sigma l_2)$ and $\{\bar{x}_k\} = bv(l_1, p) \subseteq bv(\lambda\bar{x}_k.(l_1/p))$ it follows that $bv(l_1, p) \cap Cod(\theta_1) = \{\}$. \square

Lemma 4.4. *Let there be two rules $l_i \rightarrow r_i$, $i = 1, 2$, and a position $p \in Pos(l_1)$ where l_1/p is not of the form $F(\dots)$, and substitutions θ_1 and θ_2 such that $\theta_1(l_1/p) = \theta_2(l_2)$ and $bv(l_1, p) \cap Cod(\theta_1) = \{\}$. Then l_1 overlaps l_2 and there exist a critical pair $\langle s, t \rangle$ and a substitution δ with $\delta s = \theta_1 r_1$ and $\delta t = (\theta_1 l_1)[\theta_2 r_2]_p$.*

Proof. Because $fv(l_i) \supseteq fv(r_i)$ we can assume $Dom(\theta_1) \subseteq fv(l_1)$. Let $\{\bar{x}_k\} = bv(l_1, p)$ and σ an \bar{x}_k -lifter of l_2 away from $fv(l_1)$. It follows from Lemma 4.3 that the two patterns $\lambda\bar{x}_k.(l_1/p)$ and $\lambda\bar{x}_k.(\sigma l_2)$ are unifiable. Let θ_0 be the unifier as defined in Lemma 4.3. By Lemma 3.2 there is a most general unifier θ of $\lambda\bar{x}_k.(l_1/p)$ and $\lambda\bar{x}_k.(\sigma l_2)$. Hence, l_1 overlaps l_2 at position p and there is a critical pair $\langle s, t \rangle$, such that $s = \theta r_1$ and $t = \theta(l_1[\sigma r_2]_p)$. As θ is a most general unifier there exists a substitution δ with $\theta_0 = \delta\theta$. Hence, $\delta s = \theta_0 r_1 = \theta_1 r_1$ and $\delta t = \theta_0(l_1[\sigma r_2]_p) = (\theta_0 l_1)[\theta_0(\sigma r_2)]_p = (\theta_1 l_1)[\theta'_2(\sigma r_2)]_p = (\theta_1 l_1)[\theta_2 r_2]_p$. \square

Corollary 4.5. *l_1 overlaps l_2 iff there exist $p \in Pos(l_1)$, θ_1 and θ_2 , such that $\theta_1(l_1/p) = \theta_2 l_2$ and $bv(l_1, p) \cap Cod(\theta_1) = \{\}$.*

Proof. The (\Leftarrow)-direction follows directly from Lemma 4.4. For the (\Rightarrow)-direction assume that l_1 overlaps l_2 . By Definition 4.1 there is a most general unifier of $\lambda\bar{x}_k.(l_1/p)$ and $\lambda\bar{x}_k.(\sigma l_2)$, where $\{\bar{x}_k\} = bv(l_1, p)$ and σ is an \bar{x}_k -lifter of l_2 away from $fv(l_1)$. Therefore, the result follows from Lemma 4.3. \square

Lemma 4.6 (Critical pair lemma). *Let R be a PRS and \rightarrow the corresponding reduction relation on terms. If $s \rightarrow s_i$, $i = 1, 2$, then either $s_1 \downarrow s_2$, or there are a critical pair $\langle u_1, u_2 \rangle$, a substitution δ and a position $p \in Pos(s)$, such that $s_i = s[\delta u_i]_p$, $i = 1, 2$.*

Proof. By definition of $s \rightarrow s_i$ there are rules $(l_i \rightarrow r_i) \in R$, positions $p_i \in Pos(s)$ and substitutions θ_i , such that $s/p_i = \theta_i l_i$ and $s_i = s[\theta_i r_i]_{p_i}$. Depending on the relative positions of the redexes, there are two cases.

1. $p_1 \parallel p_2$. It follows directly that $s_1 = s[\theta_1 r_1]_{p_1} \rightarrow s[\theta_1 r_1]_{p_1}[\theta_2 r_2]_{p_2} \leftarrow s[\theta_2 r_2]_{p_2} = s_2$.
2. W.l.o.g. $p_1 \leq p_2$. Let $q := p_2/p_1$. It follows that

$$s_2/p_1 = (s[\theta_2 r_2]_{p_2})/p_1 = (s/p_1)[\theta_2 r_2]_q = (\theta_1 l_1)[\theta_2 r_2]_q$$

and

$$(\theta_1 l_1)/q = (s/p_1)/q = s/p_2 = \theta_2 l_2$$

There are two cases:

- (a) The two redexes do not overlap, i.e. $q = q_1 q_2$, such that $q_1 \in \text{Pos}(l_1)$ and $l_1/q_1 = F(\bar{x}_k)$, with $F \in \text{fv}(l_1)$ (remember that l_1 is a pattern). Hence, $\theta_1(F)$ is of the form $\lambda \bar{x}_k. t$ and we define the substitution

$$\theta'_1 := \theta_1 + \{F \mapsto \lambda \bar{x}_k. (t[\theta_2 r_2]_{q_2})\}.$$

It follows that $t/q_2 = (\theta_1(F(\bar{x}_k)))/q_2 = (\theta_1(l_1/q_1))/q_2 = ((\theta_1 l_1)/q_1)/q_2 = (\theta_1 l_1)/q = \theta_2 l_2$ and hence $\theta_1 \xrightarrow{=} \theta'_1$ (Definition 3.8). By stability (Theorem 3.9) it follows that $\theta_1 r_1 \xrightarrow{*} \theta'_1 r_1$.

Let H be a new variable (i.e. unused so far in this context) and

$$\theta_0 := \theta_1 \cup \{H \mapsto \theta'_1(F)\}$$

$$\theta'_0 := \theta_0 + \{F \mapsto \theta'_1(F)\}$$

and $l_0 := l_1[H(\bar{x}_k)]_{q_1}$. From $\theta_1 \xrightarrow{=} \theta'_1$ it follows that $\theta_0 \xrightarrow{=} \theta'_0$. Theorem 3.9 yields $(\theta_1 l_1)[\theta_2 r_2]_q = (\theta_1 l_1)[\theta'_1(F(\bar{x}_k))]_{q_1} = \theta_0 l_0 \xrightarrow{*} \theta'_0 l_0 = \theta'_0 l_1 = \theta'_1 l_1 \rightarrow \theta'_1 r_1$. The step $\theta_0 l_0 \xrightarrow{*} \theta'_0 l_0$ is necessary because F may occur more than once in l_1 .

Combining all this we obtain $s_1/p_1 = \theta_1 r_1 \xrightarrow{*} \theta'_1 r_1 \xleftarrow{*} (\theta_1 l_1)[\theta_2 r_2]_q = s_2/p_1$. Placing it in the context it follows that

$$s_1 = s[\theta_1 r_1]_{p_1} \xrightarrow{*} s[\theta'_1 r_1]_{p_1} \xleftarrow{*} s[(\theta_1 l_1)[\theta_2 r_2]_q]_{p_1} = s[\theta_2 r_2]_{p_2} = s_2$$

and hence $s_1 \downarrow s_2$.

- (b) The two redexes overlap, i.e. $q \in \text{Pos}(l_1)$ and l_1/q is not of the form $F(\dots)$. W.l.o.g. let $\text{bv}(l_1) \cap \text{Cod}(\theta_1) = \{\}$. Thus it follows from Lemma 4.4 that there exist a critical pair $\langle u_1, u_2 \rangle$ and a substitution δ with $\delta u_1 = \theta_1 r_1$ and $\delta u_2 = (\theta_1 l_1)[\theta_2 r_2]_q$. This implies $s_1 = s[\theta_1 r_1]_{p_1} = s[\delta u_1]_{p_1}$ and $s_2 = s[\theta_2 r_2]_{p_2} = s[\theta_1(l_1)[\theta_2 r_2]_q]_{p_1} = s[\delta u_2]_{p_1}$. \square

Theorem 4.7. *The relation \rightarrow is locally confluent iff $u_1 \downarrow u_2$ for all critical pairs $\langle u_1, u_2 \rangle$ of the PRS R .*

Proof. The (\Rightarrow) -direction is a trivial consequence of Lemma 4.2. For the (\Leftarrow) -direction assume $u_1 \downarrow u_2$ for all critical pairs $\langle u_1, u_2 \rangle$ of R . Let $s \rightarrow s_i$, $i = 1, 2$. Then the Critical Pair Lemma 4.6 can be applied. In the first of its two cases the result follows immediately. In the second case there is a critical pair $\langle u_1, u_2 \rangle$, a substitution δ and a position $p \in \text{Pos}(s)$, such that $s_i = s[\delta u_i]_p$, $i = 1, 2$. By assumption there is a term w , such that $u_1 \xrightarrow{*} w \xleftarrow{*} u_2$. Theorem 3.9 yields $\delta u_1 \xrightarrow{*} \delta w \xleftarrow{*} \delta u_2$ and hence $s_1 = s[\delta u_1]_p \xrightarrow{*} s[\delta w]_p \xleftarrow{*} s[\delta u_2]_p = s_2$. Thus $s_1 \downarrow s_2$ and \rightarrow is locally confluent. \square

As for terminating relations confluence and local confluence are equivalent, this yields a decision procedure for the confluence of a terminating PRS.

Corollary 4.8. *Confluence of terminating PRSs is decidable.*

Example 4.9. The syntax of classical predicate logic can be described by the two types *term* and *form* of terms and formulae and by the following constants:

$$\begin{aligned} \neg &: \text{form} \rightarrow \text{form}, \\ \wedge, \vee &: \text{form} \rightarrow \text{form} \rightarrow \text{form}, \\ \forall, \exists &: (\text{term} \rightarrow \text{form}) \rightarrow \text{form}. \end{aligned}$$

For readability we write $\forall x.P(x)$ instead of $\forall(\lambda x.P(x))$ and use \wedge as an infix.

The *negation normal form* [7], where \neg is only applied to atomic formulae, can be defined via the following terminating¹ rewrite system:

$$\begin{aligned} \neg\neg &: \neg\neg P \rightarrow P, \\ \neg\wedge &: \neg(P \wedge Q) \rightarrow (\neg P) \vee (\neg Q), \\ \neg\vee &: \neg(P \vee Q) \rightarrow (\neg P) \wedge (\neg Q), \\ \neg\forall &: \neg\forall x.P'(x) \rightarrow \exists x.\neg P'(x), \\ \neg\exists &: \neg\exists x.P'(x) \rightarrow \forall x.\neg P'(x). \end{aligned}$$

There are 5 critical pairs, all of which arise by unifying the left-hand side of some rule with the subterm $\neg P$ of $\neg\neg P$, and all of which are joinable. For example,

$$\begin{array}{c} \exists x.P'(x) \\ \nearrow \neg\neg \\ \neg\neg\exists x.P'(x) \\ \searrow \neg\exists \\ \neg\forall x.\neg P'(x) \end{array}$$

is joinable because $\neg\forall x.\neg P'(x) \rightarrow \exists x.\neg\neg P'(x) \rightarrow \exists x.P'(x)$. Hence, the system is confluent, i.e. the negation normal form is uniquely determined.

Example 4.10. Going back to Example 3.4, the pure lambda-calculus, we find that both *beta* and *eta* have no critical pairs with themselves, and hence that *beta* and *eta* on their own are locally confluent. Since *eta* also terminates, this implies confluence. For the non-terminating *beta* we have to wait for the notion of orthogonality to deduce confluence. Combining *beta* and *eta* yields two critical pairs:

$$\begin{array}{c} \text{app}(S, T) \\ \nearrow \beta \\ \text{app}(\text{abs}(\lambda x.\text{app}(S, x)), T) \\ \searrow \eta \\ \text{app}(S, T) \end{array}$$

¹ Termination of this and of the other terminating systems in this paper can be proved with the techniques by van de Pol [34].

$$\begin{array}{ccc}
 & & \text{abs}(\lambda x. F(x)) \\
 & \nearrow_{\beta} & \\
 \text{abs}(\lambda x. \text{app}(\text{abs}(\lambda y. F(y)), x)) & & \\
 & \searrow_{\eta} & \\
 & & \text{abs}(\lambda y. F(y)).
 \end{array}$$

Both critical pairs are trivially joinable. Hence *beta* + *eta* is locally confluent. In Section 7 we quote a result which even permits to deduce confluence of *beta* + *eta*.

An interesting extension of this system can be obtained by adding a constant $\perp : \text{term}$ representing the completely undefined term, and the two rules

$$\begin{aligned}
 \perp \text{app} &: \text{app}(\perp, S) \rightarrow \perp \\
 \perp \text{abs} &: \text{abs}(\lambda x. \perp) \rightarrow \perp.
 \end{aligned}$$

This system has two critical pairs: *eta* overlaps $\perp \text{app}$ yielding $\langle \perp, \text{abs}(\lambda x. \perp) \rangle$, and *beta* overlaps $\perp \text{abs}$ yielding $\langle \perp, \text{app}(\perp, S) \rangle$. This is a nice example of completion: if either of the two rules had been omitted, it would have followed from the other one as a critical pair. Hence the system is locally confluent. Unfortunately, we have no theorem which implies confluence.

One of the main selling points of critical pairs has been the fact that they come with a so-called “completion algorithm”: a non-confluent rewrite system can be transformed into an equivalent (w.r.t. the equational theory) but confluent system by adding critical pairs as new reduction rules. As the last example indicates, this is also possible in our higher-order situation. However, higher-order critical pairs may no longer be pattern rewrite rules in case neither of the two components is a pattern. It is easy to see that this unfortunate state cannot arise if the original PRS we start with contains only rewrite rules where both the left and the right-hand sides are patterns, a rare situation in practice.

5. Confluence modulo equality

The most important consequence of confluence is the uniqueness of normal forms. However, there are cases of non-confluent systems where the normal forms of any term are not unique, but somehow similar to each other.

Example 5.1. In Example 4.9 we showed how to express predicate logic formulae as λ -terms. The *prenex normal form* can be described by a rewrite system R consisting of the rules

$$\begin{aligned}
 \mathbf{Q} * &: (\mathbf{Q}x. P'(x)) * Q \rightarrow \mathbf{Q}x. (P'(x) * Q) \\
 * \mathbf{Q} &: P * (\mathbf{Q}x. Q'(x)) \rightarrow \mathbf{Q}x. (P * Q'(x))
 \end{aligned}$$

for all $\mathbf{Q} \in \{\forall, \exists\}$ and $*$ $\in \{\wedge, \vee\}$, together with the rules $\neg\forall$ and $\neg\exists$ from Example 4.9.

This system terminates but is not confluent. This is because $(Qx.P'(x)) * (Qy.Q'(y))$ gives rise to the critical pair $\langle r, s \rangle := \langle Qx.(P'(x) * (Qy.Q'(y))), Qy.((Qx.P'(x)) * Q'(y)) \rangle$ which has two distinct normal forms $Qx.Qy.(P'(x) * Q'(y))$ and $Qy.Qx.(P'(x) * Q'(y))$.

This example shows that commutativity of quantifiers needs to be taken into account as well. Huet [11] introduced the notion of confluence *modulo equality* of first-order term rewriting systems. In this section some of his results are lifted to PRSs.

Let us first look at abstract relations \rightarrow and \sim .

Definition 5.2. A relation \rightarrow is *confluent modulo an equivalence relation* \sim iff

$$x \sim y \wedge x \xrightarrow{*} x' \wedge y \xrightarrow{*} y' \Rightarrow \exists x'', y''. x' \xrightarrow{*} x'' \wedge y' \xrightarrow{*} y'' \wedge x'' \sim y''.$$

A relation \rightarrow is called *normalizing* if for every x there is a y in \rightarrow -normal form such that $x \xrightarrow{*} y$.

Lemma 5.3 (Huet [11]). *Let \rightarrow be normalizing and let $x \downarrow$ denote an arbitrary normal form of x . Then \rightarrow is confluent modulo an equivalence relation \sim iff*

$$x \equiv y \Rightarrow x \downarrow \sim y \downarrow$$

where \equiv is $(\leftrightarrow \cup \sim)^*$.

Thus, we can replace the test for equivalence modulo $\rightarrow \cup \sim$ by a test of \sim -equivalence of \rightarrow -normal forms. If \rightarrow -normal forms are computable and \sim is decidable, so is \equiv .

In the sequel \cdot denotes composition of relations:

$$\rightarrow \cdot \sim = \{(x, z) \mid \exists y. x \rightarrow y \wedge y \sim z\}.$$

Huet also proves a sufficient abstract condition for confluence modulo:

Lemma 5.4 (Huet [11]). *Let \vdash be a symmetric relation, $\sim = \vdash^*$ and \rightarrow a relation, such that $\rightarrow \cdot \sim$ is terminating. Then \rightarrow is confluent modulo \sim iff the conditions \mathcal{A} and \mathcal{B} are satisfied:*

$$\mathcal{A}: \forall x, y, z. x \rightarrow y \wedge x \rightarrow z \Rightarrow y \tilde{\downarrow} z$$

$$\mathcal{B}: \forall x, y, z. x \vdash y \wedge x \rightarrow z \Rightarrow y \tilde{\downarrow} z$$

where $y \tilde{\downarrow} z :\Leftrightarrow \exists u, v. y \xrightarrow{*} u \wedge z \xrightarrow{*} v \wedge u \sim v$.

We will now concentrate on the application of confluence modulo to PRSs.

Definition 5.5. Let R be a PRS and \mathcal{E} a symmetric PRS. Then $\langle R, \mathcal{E} \rangle$ is called an *equational PRS*.

The PRS R defines a relation \rightarrow on terms as usual. Because \mathcal{E} is a symmetric PRS, for every rule $(l \rightarrow r) \in \mathcal{E}$ the reverse rule $(r \rightarrow l)$ is in \mathcal{E} as well. As \mathcal{E} is a PRS it follows that both l and r are patterns, neither l nor r are free variables and $fv(l) = fv(r)$. The symmetric reduction relation on terms defined by \mathcal{E} is denoted by $\vdash_{\mathcal{E}}$ or simply \vdash . The transitive-reflexive closure $\sim := \vdash^*$ is an equivalence relation. It is called the equivalence relation defined by \mathcal{E} .

Definition 5.6. An equational PRS $\langle R, \mathcal{E} \rangle$ is called *confluent* iff the relation \rightarrow defined by R is confluent modulo the equivalence relation \sim defined by \mathcal{E} .

Lemma 5.3 together with Theorem 3.12 implies that $s =_{R \cup \mathcal{E}} t \Leftrightarrow s \downarrow_{\beta}^{\eta} \downarrow_R =_{\mathcal{E}} t \downarrow_{\beta}^{\eta} \downarrow_R$ if R is terminating and $\langle R, \mathcal{E} \rangle$ is confluent. Confluence of equational PRSs can be proved by also considering critical pairs between R and \mathcal{E} .

Definition 5.7. Let $\langle R, \mathcal{E} \rangle$ be an equational PRS. The *critical pairs of \mathcal{E}/R* are defined as all critical pairs arising from overlapping (analogous to Definition 4.1) the left-hand sides of rules $(l_1 \rightarrow r_1)$ and $(l_2 \rightarrow r_2)$, where either $(l_1 \rightarrow r_1) \in \mathcal{E}$ and $(l_2 \rightarrow r_2) \in R$ or $(l_1 \rightarrow r_1) \in R$ and $(l_2 \rightarrow r_2) \in \mathcal{E}$.

Definition 5.8. The *critical pairs of $\langle R, \mathcal{E} \rangle$* are the union of the critical pairs of R and the critical pairs of \mathcal{E}/R .

Finally, we can generalize Huet's characterizations of conditions \mathcal{A} and \mathcal{B} to equational PRSs:

Lemma 5.9. $\langle R, \mathcal{E} \rangle$ satisfies condition \mathcal{A} iff $u_1 \tilde{\downarrow} u_2$ for every critical pair $\langle u_1, u_2 \rangle$ of R .

Proof. (1) Assume condition \mathcal{A} and let $\langle u_1, u_2 \rangle$ be a critical pair of R . By Lemma 4.2 there exists a term s such that $u_1 \leftarrow s \rightarrow u_2$. Hence $u_1 \tilde{\downarrow} u_2$ by condition \mathcal{A} .

(2) Assume $u_1 \tilde{\downarrow} u_2$ for every critical pair $\langle u_1, u_2 \rangle$ of R and $s \rightarrow s_i$, $i = 1, 2$. By the Critical Pair Lemma 4.6 there are two cases:

- (a) $s_1 \downarrow s_2$. By the reflexivity of \sim it follows that $s_1 \tilde{\downarrow} s_2$.
- (b) There is a critical pair $\langle u_1, u_2 \rangle$, a substitution δ and $p \in \text{Pos}(s)$, such that $s_i = s[\delta u_i]_p$, $i = 1, 2$. By assumption there are terms u'_1, u'_2 , such that $u_1 \xrightarrow{*} u'_1 \sim u'_2 \xleftarrow{*} u_2$. Theorem 3.9 yields $\delta u_i \xrightarrow{*} \delta u'_i$, $i = 1, 2$. As $\sim = \vdash^*$ and \vdash is defined by a symmetric PRS, Theorem 3.9 also yields $\delta(u'_1) \sim \delta(u'_2)$. So we get

$$s_1 = s[\delta u_1]_p \xrightarrow{*} s[\delta u'_1]_p \sim s[\delta u'_2]_p \xleftarrow{*} s[\delta u_2]_p = s_2$$

and hence $s_1 \tilde{\downarrow} s_2$. \square

Lemma 5.10. *Let $\langle R, \mathcal{E} \rangle$ be an equational PRS and R left-linear. Condition \mathcal{B} is satisfied iff for every critical pair $\langle u_1, u_2 \rangle$ of \mathcal{E}/R we have $u_1 \downarrow u_2$.*

Proof. (1) Assume condition \mathcal{B} . Let $\langle u_1, u_2 \rangle$ be a critical pair of \mathcal{E}/R . There is a rule $(l_1 \rightarrow r_1) \in R$ and a rule $(l_2 \rightarrow r_2) \in \mathcal{E}$, such that w.l.o.g. l_1 overlaps l_2 . By Lemma 4.2 there is a term s , such that $u_1 \leftarrow s \vdash u_2$. Hence condition \mathcal{B} yields $u_1 \downarrow u_2$. The other case where l_2 overlaps l_1 is symmetric.

(2) Assume $u_1 \downarrow u_2$ for every critical pair $\langle u_1, u_2 \rangle$ of \mathcal{E}/R . Let $s \rightarrow s_1$ and $s \vdash s_2$. By definition of \rightarrow and \vdash there are rules $(l_1 \rightarrow r_1) \in R$, $(l_2 \rightarrow r_2) \in \mathcal{E}$, positions $p_1, p_2 \in \text{Pos}(s)$ and substitutions θ_1, θ_2 , such that $s/p_1 = \theta_1 l_1$, $s_1 = s[\theta_1 r_1]_{p_1}$, $s/p_2 = \theta_2 l_2$, $s_2 = s[\theta_2 r_2]_{p_2}$. There are three cases:

- (a) $p_1 \parallel p_2$. It is trivial that $s_1 = s[\theta_1 r_1]_{p_1} \vdash s[\theta_1 r_1]_{p_1} [\theta_2 r_2]_{p_2} \leftarrow s[\theta_2 r_2]_{p_2} = s_2$. Hence, $s_1 \downarrow s_2$ and condition \mathcal{B} is satisfied.
- (b) $p_2 \leq p_1$ and $q := p_1/p_2$. It follows that $s_1/p_2 = (s[\theta_1 r_1]_{p_1})/p_2 = (s/p_2)[\theta_1 r_1]_q = (\theta_2 l_2)[\theta_1 r_1]_q$ and $(\theta_2 l_2)/q = (s/p_2)/q = s/p_1 = \theta_1 l_1$. There are two cases:
 - (i) The two redexes do not overlap, i.e. $q = q_1 q_2$, such that $q_1 \in \text{Pos}(l_2)$ and $l_2/q_1 = F(\bar{x}_k)$, where $F \in \text{fv}(l_2)$ (remember: l_2 is a pattern). Hence, $\theta_2(F)$ is of the form $\lambda \bar{x}_k. t$ and we define the substitution

$$\theta'_2 := \theta_2 + \{F \mapsto \lambda \bar{x}_k. (t[\theta_1 r_1]_{q_2})\}.$$

Thus, we have

$$t/q_2 = (\theta_2(F(\bar{x}_k)))/q_2 = (\theta_2(l_2/q_1))/q_2 = ((\theta_2 l_2)/q_1)/q_2 = (\theta_2 l_2)/q = \theta_1 l_1$$

and hence $\theta_2 \xrightarrow{=} \theta'_2$. Theorem 3.9 yields $\theta_2 r_2 \rightarrow^* \theta'_2 r_2$.

Let H be a new variable (unused so far in this context) and

$$\theta_0 := \theta_2 \cup \{H \mapsto \theta'_2(F)\}$$

$$\theta'_0 := \theta_0 + \{F \mapsto \theta'_2(F)\}$$

and $l_0 := l_2[H(\bar{x}_k)]_{q_1}$. From $\theta_2 \xrightarrow{=} \theta'_2$ it follows that $\theta_0 \xrightarrow{=} \theta'_0$. Applying Theorem 3.9 we get $(\theta_2 l_2)[\theta_1 r_1]_q = (\theta_2 l_2)[\theta'_2(F(\bar{x}_k))]_{q_1} = \theta_0 l_0 \xrightarrow{*} \theta'_0 l_0 = \theta'_0 l_2 = \theta'_2 l_2 = \theta'_2 l_2 \vdash \theta'_2 r_2$ and therefore

$$s_2/p_2 = \theta_2 r_2 \xrightarrow{*} \theta'_2 r_2 \vdash \theta'_2 l_2 \xleftarrow{*} (\theta_2 l_2)[\theta_1 r_1]_q = s_1/p_2$$

Placing it in the context it follows that

$$\begin{array}{ccc} s_2 = s[\theta_2 r_2]_{p_2} & & s[(\theta_2 l_2)[\theta_1 r_1]_q]_{p_2} = s[\theta_1 r_1]_{p_1} = s_1 \\ \downarrow * & & \downarrow * \\ s[\theta'_2 r_2]_{p_2} & \vdash & s[\theta'_2 l_2]_{p_2} \end{array}$$

and hence $s_1 \downarrow s_2$.

- (ii) The two redexes overlap, i.e. $q \in \text{Pos}(l_2)$ and l_2/q is not of the form $F(\dots)$. W.l.o.g. assume that $\text{bv}(l_2) \cap \text{Cod}(\theta_2) = \{\}$. Thus it follows from Lemma 4.4 that there exist a critical pair $\langle u_1, u_2 \rangle$ and a substitution δ , such that $\delta u_1 = \theta_2 r_2$ and $\delta u_2 = \theta_2(l_2)[\theta_1 r_1]_q$. By Definition 5.7 $\langle u_1, u_2 \rangle$ is a critical pair of \mathcal{E}/R . By assumption there exist terms u'_1, u'_2 , such that $u_1 \xrightarrow{*} u'_1 \sim u'_2 \xleftarrow{*} u_2$. From Theorem 3.9 it follows that $\delta u_i \rightarrow^* \delta u'_i$, $i = 1, 2$. As \sim is defined by a symmetric PRS, Theorem 3.9 also yields $\delta u'_1 \sim \delta u'_2$. So we have $s_2 = s[\delta u_1]_{p_2} \xrightarrow{*} s[\delta u'_1]_{p_2} \sim s[\delta u'_2]_{p_2} \xleftarrow{*} s[\delta u_2]_{p_2} = s_1$ and hence $s_1 \downarrow s_2$.
- (c) The last case is $p_1 \leq p_2$ with $q := p_2/p_1$. This case is similar to case 2b, except that we have to make use of the condition that R is left linear. There are two cases:
- (i) If the two redexes do not overlap, we again have $\theta_1(F) = \lambda \bar{x}_k. t$ and define

$$\theta'_1 := \theta_1 + \{F \mapsto \lambda \bar{x}_k. (t[\theta_2 r_2]_q)\}.$$

So we get $\theta_1 \vdash \theta'_1$ and Theorem 3.9 yields $\theta_1 r_1 \xrightarrow{*} \theta'_1 r_1$. As the PRS R is assumed to be left-linear, the free variable F occurs only once in l_1 . Therefore

$$(\theta_1 l_1)[\theta_2 r_2]_q = (\theta_1 l_1)[\theta'_1(F(\bar{x}_k))]_{q_1} = \theta'_1 l_1 \rightarrow \theta'_1 r_1$$

It follows that

$$\begin{array}{ccc} s_1 = s[\theta_1 r_1]_{p_1} & & s[(\theta_1 l_1)[\theta_2 r_2]_q]_{p_1} = s[\theta_2 r_2]_{p_2} = s_2 \\ & \downarrow * & \downarrow \\ & s[\theta_1 r_1]_{p_1} \sim & s[\theta'_1 r_1]_{p_1} \end{array}$$

and hence $s_1 \downarrow s_2$.

- (ii) The case where the two redexes overlap, i.e. $q \in \text{Pos}(l_1)$ and l_1/q is not of the form $F(\dots)$ is completely analogous to case 2(b)(ii). \square

Huet gives a simple example which shows that left-linearity of R is essential.

With the help of these lemmas it is now possible to formulate a sufficient criterion for the confluence of an equational PRS.

Theorem 5.11. *Let $\langle R, \mathcal{E} \rangle$ be an equational PRS, such that R is left-linear and $(\rightarrow \cdot \sim)$ terminates. Then the equational PRS $\langle R, \mathcal{E} \rangle$ is confluent iff $u_1 \downarrow \sim u_2 \downarrow$ for all critical pairs $\langle u_1, u_2 \rangle$ of $\langle R, \mathcal{E} \rangle$, where $u_i \downarrow$ is an arbitrary \rightarrow -normal form of u_i .*

Proof. First note that since $\rightarrow \cdot \sim$ terminates, so does \rightarrow because \sim is reflexive. Hence the notation $t \downarrow$ is always defined. The theorem follows almost directly from Lemmas 5.4, 5.9 and 5.10 because $u_1 \downarrow \sim u_2 \downarrow \Leftrightarrow u_1 \downarrow \sim u_2 \downarrow$: the (\Leftarrow) -direction is trivial; for the other direction note that if $u_1 \xrightarrow{*} u'_1 \sim u'_2 \xleftarrow{*} u_2$ then confluence implies $u_1 \downarrow \sim u_2 \downarrow$ using Lemma 5.3. \square

Example 5.12. Now we can prove that the PRS R from Example 5.1 is confluent modulo the symmetric PRS

$$\mathcal{E} := \{Qx.Qy.H(x, y) \leftrightarrow Qy.Qx.H(x, y) \mid Q \in \{\forall, \exists\}\}.$$

As usual, \sim denotes the equivalence relation defined by \mathcal{E} . It is easily checked that all critical pairs of R arise by overlapping $Q*$ with $*Q$, and that their two normal forms are $\langle r\downarrow, s\downarrow \rangle = \langle Qx.Qy.(P'(x) * Q'(y)), Qy.Qx.(P'(x) * Q'(y)) \rangle$. Thus we have $r\downarrow \sim s\downarrow$.

In addition, we can overlap the rules $Q*$ and $*Q$ with the rules in \mathcal{E} which gives rise to the critical pairs of \mathcal{E}/R , whose normal forms are $\langle Qx.Qy.(H(x, y) * Q), Qy.Qx.(H(x, y) * Q) \rangle$ and $\langle Qx.Qy.(Q * H(x, y)), Qy.Qx.(Q * H(x, y)) \rangle$, both of which are contained in \sim .

As R is left-linear and $(\rightarrow \cdot \sim)$ terminates, it follows from Theorem 5.11 that $\langle R, \mathcal{E} \rangle$ is a confluent equational PRS: modulo quantifier-commutativity, R computes a unique prenex normal form. As $=_{\mathcal{E}}$ is decidable the relation $=_{R \cup \mathcal{E}}$ is decidable by Lemma 5.3.

It is interesting to note that confluence is destroyed by a frequently employed optimization in computing prenex forms:

$$\begin{aligned} (\forall x.P'(x)) \wedge (\forall y.Q'(y)) &\rightarrow \forall x.(P'(x) \wedge Q'(x)), \\ (\exists x.P'(x)) \vee (\exists y.Q'(y)) &\rightarrow \exists x.(P'(x) \vee Q'(x)). \end{aligned}$$

These new rules give rise to non-trivial critical pairs with R , requiring, for example, the further rule $\forall x.\forall y.(P'(x) \wedge Q'(y)) \rightarrow \forall x.(P'(x) \wedge Q'(x))$. It seems unlikely that confluence can be regained by some form of completion.

6. Orthogonal pattern rewrite systems

We now turn our attention to a subclass of PRSs, the so called orthogonal ones. An *Orthogonal Pattern Rewrite System (OPRS)* is a PRS that is left linear and has no critical pairs. This means that there are no rules whose left-hand sides overlap (see Definition 4.1).

Orthogonal term-rewriting systems have a long history [28, 12, 17]. They have been studied very closely because of their similarity to functional programs with pattern matching. The key property of orthogonal systems is their confluence, regardless of whether they terminate or not. We show that this holds for OPRSs as well. The main idea is to define a relation \geq on terms such that $\rightarrow \subseteq \geq \subseteq \overset{*}{\rightarrow}$, which implies $\geq^* = \overset{*}{\rightarrow}$. It is well known that in this case \rightarrow is confluent if \geq has the *diamond property*: $r \geq s \wedge r \geq t \Rightarrow \exists u. s \geq u \wedge t \geq u$.

6.1. The classical proof

In this section we generalize Aczel's [1] confluence proof from his "consistent sets of contraction schemes" to arbitrary OPRSs. Note that the former are a proper subset

of the latter. The proof proceeds roughly like the one for the untyped λ -calculus due to Tait and Martin-Löf [3]. Although we want to prove the confluence of OPRSs, the first steps towards the standard proof work just as well for HRSs.

Given a fixed HRS R , *parallel reduction* w.r.t. R is the smallest relation \geq on terms which is closed under the following inference rules:

$$\frac{s_i \geq t_i \quad (i=1, \dots, n)}{a(\overline{s_n}) \geq a(\overline{t_n})} \text{ (A)} \quad \frac{s \geq t}{\lambda x.s \geq \lambda x.t} \text{ (L)}$$

$$\frac{s_i \geq t_i \quad (i=1, \dots, n) \quad c(\overline{t_n}) = \theta l \quad (l \rightarrow r) \in R}{c(\overline{s_n}) \geq \theta r} \text{ (R)}$$

where a is an atom of type $\overline{\tau_n} \rightarrow \tau$ and $s_i : \tau_i$. Our relation \geq is essentially Aczel's $>$. It is instructive to look at a few variations on this theme:

- For orthogonal TRSs, (L) is dropped, and (R) becomes $\theta l \geq \theta r$.
- For pure λ -calculus, (A) becomes $s_0 \geq t_0 \wedge s_1 \geq t_1 \Rightarrow (s_0 s_1) \geq (t_0 t_1)$, (R) becomes $s \geq s' \wedge t \geq t' \Rightarrow (\lambda x.s)t \geq \{\lambda x.t'\}(s')$, and we need to add $a \geq a$. The latter is a special case of the above formulation of (A).
- For OPRSs, Nipkow [27] uses an apparently weaker version of (R) which does not allow overlapping reductions: $(\forall F \in fv(l). \theta(F) \geq \theta'(F)) \Rightarrow \theta l \geq \theta' r$. Note that for OPRSs the two versions of (R) coincide because left-hand sides do not overlap. Nevertheless it seems that by using the stronger form above, the confluence proof is simplified.

Further variations of this technique appear in the literature [39, 6].

Parallel reduction has a number of interesting properties.

Lemma 6.1. $s \geq s$ holds for all terms s in long $\beta\eta$ -normal form.

Proof. By induction on the structure of s . \square

Because all rules are of base type we again have

Lemma 6.2. If $\lambda x.s \geq t'$ then $t' = \lambda x.t$ and $s \geq t$ for some t .

More importantly we find that \rightarrow and \geq are strongly related:

Lemma 6.3. $\rightarrow \subseteq \geq \subseteq \overset{*}{\rightarrow}$.

Proof. Using the inductive definition of \rightarrow (Definition 3.6).

- $s \rightarrow t \Rightarrow s \geq t$ is proved by induction on the structure of $s \rightarrow t$. If $\theta l \rightarrow \theta r$ then $l = c(\overline{l_n})$ and Lemma 6.1 implies $\theta l_i \geq \theta l_i$ and thus (R) yields $\theta l \geq \theta r$. If $a(\overline{s_n}) \rightarrow a(\overline{t_n})$, where $s_k \rightarrow t_k$ and $s_i = t_i$ for all $i \neq k$, then $s_i \geq t_i$ for all i by induction hypothesis together with Lemma 6.1. Hence $a(\overline{s_n}) \geq a(\overline{t_n})$ follows by rule (A). The remaining case is trivial.
- $s \geq t \Rightarrow s \overset{*}{\rightarrow} t$ is proved by induction on the structure of $s \geq t$. If $c(\overline{s_n}) \geq \theta r$ by rule (R), then $s_i \overset{*}{\rightarrow} t_i$ follows by induction hypothesis and hence $c(\overline{s_n}) \overset{*}{\rightarrow} c(\overline{t_n}) = \theta l \rightarrow \theta r$.

If $a(\overline{s_n}) \geq a(\overline{t_n})$ by rule (A), $s_i \xrightarrow{*} t_i$ follows by induction hypothesis and hence $a(\overline{s_n}) \xrightarrow{*} a(\overline{t_n})$. The remaining case is trivial. \square

Next we show that \geq interacts very nicely with substitutions. The proof is similar to the one of Theorem 3.9 but considerably simpler.

Lemma 6.4. *If $s \geq s'$ and $\theta \geq \theta'$ then $\theta s \geq \theta' s'$.*

Proof. By induction on the order of θ (see Definition 2.1) with a nested induction on the structure of $s \geq s'$.

- (A) If $s = a(\overline{s_n}) \geq a(\overline{s'_n}) = s'$, then by the inner induction hypothesis $\theta s_i \geq \theta' s'_i$. We distinguish two cases.
1. If $a \notin \mathcal{D}om(\theta)$ (and hence $a \notin \mathcal{D}om(\theta')$) then $\theta s = a(\overline{\theta s_n}) \geq a(\overline{\theta' s'_n}) = \theta' s'$ follows by rule (A).
 2. If $a \in \mathcal{D}om(\theta)$ then $\theta a = \lambda \overline{y_n}. t$. By Lemma 6.2 it follows that $\theta' a = \lambda \overline{y_n}. t'$ where $t \geq t'$. Define the substitutions $\delta = \{y_n \mapsto \theta s_n\}$ and $\delta' = \{y_n \mapsto \theta' s'_n\}$ and notice that we have $\delta \geq \delta'$. If a is of type $\overline{\tau_n} \rightarrow \tau$, the definition of *ord* implies $ord(\tau_i) < ord(\overline{\tau_n} \rightarrow \tau)$ and hence $ord(\delta) < ord(\delta')$ (note that this also holds in case $n=0$). Thus the outer induction hypothesis applies: $\delta t \geq \delta' t'$ because $t \geq t'$. Thus we obtain: $\theta s = (\theta a)(\overline{\theta s_n}) = \delta t \geq \delta' t' = (\theta' a)(\overline{\theta' s'_n}) = \theta' s'$.
- (L) If $s = \lambda x. t$, $s' = \lambda x. t'$ and $t \geq t'$ then the inner induction hypothesis yields $\theta s = \lambda x. (\theta t) \geq \lambda x. (\theta' t') = \theta' s'$ using rule (L).
- (R) If $s = c(\overline{s_n})$, $s_i \geq s'_i$, $c(\overline{s'_n}) = \delta l$ and $s' = \delta r$ then the inner induction hypothesis implies $\theta s_i \geq \theta' s'_i$. Since $c(\overline{\theta' s'_n}) = \theta' \delta l$ rule (R) directly yields $\theta s = c(\overline{\theta s_n}) \geq \theta' \delta r = \theta' s'$. \square

The following lemma expresses a simple idea: in a reduction step $\theta l \geq s$, where l is a linear pattern and does not overlap with any rule, the l -part cannot change, i.e. all reductions must take place inside the terms introduced via θ .

Lemma 6.5. *Let R be an OPRS and let $l_0 = \lambda \overline{x_n}. l$ be a linear pattern that does not overlap any left-hand side of R . If $\theta l \geq s$ and $\mathcal{C}od(\theta) \cap \{\overline{x_n}\} = \{\}$ ($\mathcal{D}om(\theta) \cap \{\overline{x_n}\} = \{\}$ by convention!) then there exists a substitution θ' such that $\theta' l = s$, $\theta|_{fv(l_0)} \geq \theta'$ and $\mathcal{D}om(\theta') = fv(l_0)$.*

Proof. By induction on the structure of l .

1. If $l = c(\overline{l_m})$, then the last inference in $\theta l \geq s$ must be (A) or (R). In either case there are t_i such that $\theta l_i \geq t_i$. Each $\lambda \overline{x_n}. l_i$ is again a linear pattern. Since l_0 does not overlap any left-hand side of R , it follows easily from Corollary 4.5 that neither do the $\lambda \overline{x_n}. l_i$. Now the induction hypothesis yields substitutions θ_i such that $\theta_i l_i = t_i$, $\theta|_{fv(\lambda \overline{x_n}. l_i)} \geq \theta_i$ and $\mathcal{D}om(\theta_i) = fv(\lambda \overline{x_n}. l_i)$. Now let $\theta' := \bigcup_{i=1}^m \theta_i$, which is a well-defined substitution because the $\mathcal{D}om(\theta_i)$ are disjoint, thanks to linearity. In particular, we have $\theta' l = c(\overline{l_m})$, $\theta|_{fv(l_0)} \geq \theta'$ and $\mathcal{D}om(\theta') = fv(l_0)$. If rule (A) was used, $\theta' l = s$ as required. Rule (R) cannot have been used because it would

mean that $c(\overline{t_m}) = \delta l'$, where l' is the left-hand side of a rule in R . Because $\theta' l = \delta l'$, Corollary 4.5 implies that l_0 overlaps l' , a contradiction.

2. If $l = F(\overline{s_m})$ then $l \downarrow_{\eta} = F(\overline{y_m})$ because l is a pattern. Wlog $\theta F = \lambda \overline{y_m}.r$. Hence $\theta l = (\lambda \overline{y_m}.r)(\overline{y_m}) \uparrow_{\beta}^{\eta} = r$ and thus $r \geq s$. Let $\theta' = \{F \mapsto \lambda \overline{y_m}.s\}$. Obviously, θ' has the desired properties.
3. If $l = \lambda x.l'$ then the claim follows easily from the induction hypothesis. (At this point it becomes important not simply to drop the λx because the remaining l' might no longer be a pattern.) \square

Lemma 6.6. *Let R be an OPRS and let l be the left-hand side of a rule in R . If $\theta l = a(\overline{s_n})$ and $s_i \geq t_i$ then there exists a θ' such that $\theta' l = a(\overline{t_n})$ and $\theta \geq \theta'$.*

Proof. l must be of the form $a(\overline{t_n})$ and a must be a constant. Since R is orthogonal, none of the l_i , all of which are patterns, can overlap with any left-hand side of R . By Lemma 6.5 we obtain substitutions θ_i such that $\theta_i l_i = t_i$, $\text{Dom}(\theta_i) = \text{fv}(l_i)$ and $\theta|_{\text{fv}(l_i)} \geq \theta_i$. Because l is linear, the θ_i have disjoint domains and $\theta'' := \bigcup_{i=1}^n \theta_i$ is well-defined, $\theta'' l = a(\overline{t_n})$ and $\theta|_{\text{fv}(l)} \geq \theta''$. Using Lemma 6.1 it is trivial to extend θ'' to θ' with the desired properties. \square

As a corollary we easily obtain

Corollary 6.7 (Coherence). *Let R be an OPRS. If $(l \rightarrow r) \in R$, $\theta l = a(\overline{s_n})$ and $s_i \geq t_i$, $i = 1, \dots, n$, then there exists a θ' such that $\theta' l = a(\overline{t_n})$ and $\theta r \geq \theta' r$.*

We can finally show that \geq has the diamond property, i.e. if $s \geq t_i$, $i = 0, 1$, then there exists a u such that $t_i \geq u$, $i = 0, 1$.

Theorem 6.8. *If R is an OPRS then \geq has the diamond property.*

Proof. We assume that $s \geq s_i$, $i = 0, 1$, and show by induction on the structure of $s \geq s_0$ that there is a t such that $s_i \geq t$, $i = 0, 1$.

(L) If $s = \lambda x.u \geq \lambda x.u_0 = s_0$ and $u \geq u_0$, then Lemma 6.2 implies $s_1 = \lambda x.u_1$ and $u \geq u_1$. By induction hypothesis there exists a u such that $u_i \geq u$ and hence $s_i \geq \lambda x.u =: t$ by (L).

(A) If $s = a(\overline{u_n})$, $u_j \geq u_{0j}$, and $a(\overline{u_{0n}}) = s_0$, then $s \geq s_1$ can only be a consequence of (A) or (R). In either case $u_j \geq u_{1j}$ and hence by induction hypothesis there are t_j such that $u_{ij} \geq t_j$. If $s \geq s_1$ by (A) then $s_1 = a(\overline{u_{1n}})$ and thus $t := a(\overline{t_n})$ closes the diamond.

If $s \geq s_1$ by (R) then $a(\overline{u_{1n}}) = \theta l \rightarrow \theta r = s_1$ for suitable θ and $(l \rightarrow r) \in R$. Coherence yields θ' such that $a(\overline{t_n}) = \theta' l$ and $s_1 = \theta r \geq \theta' r =: t$. By rule (R) we also have $s_0 = a(\overline{u_{1n}}) \geq t$, thus closing the diamond.

(R) If $s = a(\overline{u_n})$, $u_j \geq u_{0j}$ and $a(\overline{u_{0n}}) = \theta l \rightarrow \theta r = s_0$ then $s \geq s_1$ can only be a consequence of (A) or (R). In either case $u_j \geq u_{1j}$ and hence by induction hypothesis there are t_j such that $u_{ij} \geq t_j$. Coherence yields θ' such that $a(\overline{t_n}) = \theta' l$ and

$s_0 = \theta r \geq \theta' r =: t$. By rule (R) we also have $a(\overline{u_{1n}}) \geq t$. If $s \geq s_1$ by (A) then $s_1 = a(\overline{u_{1n}})$, thus closing the diamond.

If $s \geq s_1$ by (R) then $a(\overline{u_{1n}}) = \theta_1 l_1 \rightarrow \theta_1 r_1 = s_1$ for suitable θ_1 and $(l_1 \rightarrow r_1) \in R$. Coherence yields θ'_1 such that $a(\overline{t_n}) = \theta'_1 l_1$ and $\theta_1 r_1 \geq \theta'_1 r_1$. Therefore $\theta' l = \theta'_1 l_1$, i.e. l and l_1 overlap at ε . Because R is orthogonal this means $(l \rightarrow r) = (l_1 \rightarrow r_1)$, $\theta' = \theta'_1$ and hence $s_1 \geq \theta'_1 r_1 = \theta' r = t$. \square

6.2. Confluence by complete superdevelopments

This proof is inspired by Takahashi's proof of the confluence of semi-orthogonal CLC (conditional lambda calculus) [40] (see also Section 7). Takahashi uses *developments*, i.e. chains of reductions where no newly created redexes are contracted, to transform terms t into “normal forms” t^* where all redexes in t have been contracted. Her key idea is to show that if $t \geq t'$ then $t' \geq t^*$, where \geq is the standard notion of parallel reduction instead of Aczel's \geq .

In the sequel we recast her work in the context of OPRSs using \geq . Thus the transformation from t to t^* is not a development, but is more like a superdevelopment defined by van Raamsdonk [37] for λ -calculus. We call the transformation $t \geq t^*$ a *complete superdevelopment*. Although the classical proof and Takahashi's version share the basic lemmas, we find that her auxiliary notion t^* leads to a shorter and more appealing proof of her main lemma (Lemma 6.10) compared to the direct proof of Theorem 6.8. Both proofs are shown to allow the reader a direct comparison.

Definition 6.9. Let R be an OPRS. For every λ -term t in long $\beta\eta$ -normal form the term t^* is defined recursively as follows.

1. $(\lambda x. t)^* = \lambda x. (t^*)$,
2. $a(\overline{t_n})^* = a(\overline{t_n^*})$, if $a(\overline{t_n^*})$ is not a redex,
3. $a(\overline{t_n})^* = \theta(r)$, if there is a rule $(l \rightarrow r) \in R$ and a substitution θ , such that $\theta(l) = a(\overline{t_n^*})$, where a is an atom of type $\overline{\tau_n} \rightarrow \tau$ and $t_i : \tau_i$. The term t^* is well defined, because for OPRSs the rule $(l \rightarrow r)$ and the substitution θ are uniquely determined by the redex $a(\overline{t_n^*})$.

Lemma 6.10. Let R be an OPRS, \geq the corresponding parallel reduction and t a λ -term in long $\beta\eta$ -normal form. Then we have $t \geq t' \Rightarrow t' \geq t^*$

Proof. By induction on the size of t . There are three cases:

- (L) $t = \lambda x. s \geq \lambda x. s' = t'$ with $s \geq s'$. By induction hypothesis $s' \geq s^*$ and hence $t' = \lambda x. s' \geq \lambda x. s^* = t^*$.
- (A) $t = a(\overline{t_n}) \geq a(\overline{t'_n}) = t'$ with $t_i \geq t'_i$ ($i = 1, \dots, n$). By induction hypothesis we have $t'_i \geq t_i^*$ ($i = 1, \dots, n$). Now there are two cases:
 1. If $a(\overline{t_n^*})$ is no redex, i.e. there are no $(l \rightarrow r) \in R$ and θ such that $\theta l = a(\overline{t_n^*})$, then $t' = a(\overline{t'_n}) \geq a(\overline{t_n^*}) = t^*$.
 2. If $a(\overline{t_n^*})$ is a redex, i.e. there are $(l \rightarrow r) \in R$ and θ such that $\theta(l) = a(\overline{t_n^*})$, then $t' = a(\overline{t'_n}) \geq \theta(r) = t^*$.

(R) $t = a(\overline{t_n}) \geq \theta(r)$ for some substitution θ and $(l \rightarrow r) \in R$ such that $t_i \geq t'_i$ ($i = 1, \dots, n$) and $\theta(l) = a(\overline{t_n})$. By induction hypothesis we have $t'_i \geq t_i^*$. It follows from Lemma 6.6 that there is a substitution θ' , such that $\theta'(l) = a(\overline{t_n^*})$ and $\theta \geq \theta'$. Lemma 6.4 yields $t' = \theta(r) \geq \theta'(r) = t^*$. \square

Theorem 6.11. *OPRSs are confluent.*

Proof. It follows from Lemma 6.10 that for any OPRS R the relation \geq has the diamond property. Hence \rightarrow is confluent, because of Lemma 6.3. \square

7. Related work

As already indicated in the introduction, PRSs are closely related to CRSs [16, 18]. Confluence of CRSs has been investigated for orthogonal systems. HRSs are the same as Wolfram's higher-order term rewriting systems. This abundance of slightly different frameworks has lead to a notion of higher-order rewriting system (HORS) which is parameterized by a "substitution calculus" [31, 29] and generalizes all of the aforementioned frameworks. In the case of HRSs/PRSs the substitution calculus is the simply-typed λ -calculus. It has been shown that all *weakly orthogonal* HORSs are confluent [31, 29]. Although the notion of weak orthogonality for HORSs is defined directly rather than in terms of critical pairs, it can be translated as follows:

Definition 7.1. A PRS R is called *weakly orthogonal* iff it is left-linear and all of its critical pairs are of the form $\langle u, u \rangle$.

A direct proof of confluence for all weakly orthogonal PRSs is given by van Raamsdonk [38]. This generalizes one of the results obtained in our paper, but at a considerable increase in complexity. Hence, we believe that the simplified proofs of confluence for OPRSs which we provide have their own merit. On the other hand, van Oostrom's techniques yield further dividends, for example that the weakly-orthogonal combination of left-linear and confluent PRSs is again confluent [29, Theorem 3.5.13], generalizing a theorem by Nipkow [27, Theorem 6.1].

This result about weakly orthogonal systems has some interesting consequences. For example, it implies that lambda-calculus with both *beta* and *eta* (Example 4.10) is confluent: both rules are left-linear and all critical pairs are of the form $\langle u, u \rangle$.

Takahashi [40] has investigated a condition called *semi-orthogonality* which lies in between orthogonality and weak orthogonality: it is strictly weaker than orthogonality but might coincide with weak orthogonality. Takahashi proves confluence of semi-orthogonal "conditional lambda-calculi" (CLC), her own brand of higher-order rewrite systems. As CLC are very close to PRSs, semi-orthogonality can be defined for PRSs in the same way, and her confluence result carries over [22]. It turns out that one of her requirements, namely that the left-hand sides of two different rules do not overlap at position ε , is superfluous.

Finally, there is a large body of research that is concerned with the combination of λ -calculi and rewrite systems (see, for example, [2]). Although superficially similar to our approach, it is in fact quite different: whereas we consider rewriting modulo the conversions of the λ -calculus, i.e. λ -calculus is a meta-language for describing rewrite systems, they combine β -reduction with other restricted forms of rewrite rules on the same level. This yields relatively strong modularity results for special combinations, e.g. adding β -reduction to a left-linear confluent TRS preserves confluence [24], whereas HRSs aim for a general theory of arbitrary higher-order rules.

Finally, it should be mentioned that methods for proving termination of HRSs/PRSs are only just emerging [34, 35, 21, 14, 13]. More work is needed in this area.

Acknowledgement

This paper has benefitted enormously from the amazing scrutiny of Ralph Matthes who was volunteered to present very preliminary fragments of it in a seminar.

References

- [1] P. Aczel, A general Church–Rosser theorem, Tech. Report, Univ. of Manchester, 1978.
- [2] F. Barbanera, M. Fernández, H. Geuvers, Modularity of strong normalization and confluence in the algebraic λ -cube, in: 9th IEEE Symp. Logic in Computer Science, IEEE Computer Society Press, Silver Spring, MD, 1994, pp. 406–415.
- [3] H.P. Barendregt, *The Lambda Calculus, its Syntax and Semantics*, 2nd ed., North-Holland, Amsterdam, 1984.
- [4] H.P. Barendregt, Lambda calculi with types, in: S. Abramsky, D.M. Gabbay, T.S.E. Maibaum (Eds.), *Handbook of Logic in Computer Science*, vol. 2, Oxford Univ. Press, Oxford, 1992, pp. 118–309.
- [5] N. Dershowitz, J.-P. Jouannaud, Rewrite systems, in: J. van Leeuwen (Ed.), *Formal Models and Semantics*, *Handbook of Theoretical Computer Science*, vol. B, Elsevier, Amsterdam, 1990, pp. 243–320.
- [6] D.J. Dougherty, Adding algebraic rewriting to the untyped lambda-calculus, in: R.V. Book (Ed.), *Proc. 4th Int. Conf. Rewriting Techniques and Applications*, *Lecture Notes in Computer Science*, vol. 488, Springer, Berlin, 1991, pp. 34–48.
- [7] J.H. Gallier, *Logic for Computer Science*. Harper & Row, New York, 1986.
- [8] W.D. Goldfarb, The undecidability of the second-order unification problem, *Theoret. Comput. Sci.* 13 (1981) 225–230.
- [9] M.J.C. Gordon, T.F. Melham, *Introduction to HOL: A Theorem-Proving Environment for Higher Order Logic*, Cambridge Univ. Press, Cambridge, 1993.
- [10] J.R. Hindley, J.P. Seldin, *Introduction to Combinators and λ -Calculus*, Cambridge Univ. Press, Cambridge, 1986.
- [11] G. Huet, Confluent reductions: abstract properties and applications to term rewriting systems, *J. ACM*, 27 (1980) 797–821.
- [12] G. Huet, J.-J. Lévy, Computations in orthogonal rewriting systems, in: J.-L. Lassez, G. Plotkin (Eds.), *Computational Logic: Essays in Honor of Alan Robinson*, MIT Press, Cambridge, MA, 1991, pp. 395–443.
- [13] J.-P. Jouannaud, A. Rubio, Higher-order simplification orderings, in: H. Comon (Ed.), *Rewriting Techniques and Applications*, *Lecture Notes in Computer Science*, Springer, Berlin, 1997.
- [14] S. Kahrs, Towards a domain theory for termination proofs, in: J. Hsiang (Ed.), *Rewriting Techniques and Applications*, *Lecture Notes in Computer Science*, vol. 914, Springer, Berlin, 1995, pp. 241–255.
- [15] S. Kahrs, The variable containment problem, in: G. Dowek, J. Heering, K. Meinke, B. Möller (Eds.), *Higher-Order Algebra, Logic, and Term Rewriting*, *Lecture Notes in Computer Science*, vol. 1074, Springer, Berlin, 1996, pp. 109–123.
- [16] J.W. Klop, *Combinatory Reduction Systems*, *Mathematical Centre Tracts*, vol. 127, Mathematisch Centrum, Amsterdam, 1980.

- [17] J.W. Klop, Term rewriting systems, in: S. Abramsky, D.M. Gabbay, T.S.E. Maibaum (Eds.), *Handbook of Logic in Computer Science*, vol. 2, Oxford Univ. Press, Oxford, 1992, pp. 2–116.
- [18] J.W. Klop, V. van Oostrom, F. van Raamsdonk, Combinatory reduction systems: introduction and survey, *Theoret. Comput. Sci.* 121 (1993) 279–308.
- [19] D.E. Knuth, P.B. Bendix, Simple word problems in universal algebra, in: J. Leech (Ed.), *Computational Problems in Abstract Algebra*, Pergamon Press, Oxford, 1970, pp. 263–297.
- [20] C.A. Loria-Sáenz, A theoretical framework for reasoning about program construction based on extensions of rewrite systems, Ph.D. Thesis, Universität Kaiserslautern, 1993.
- [21] O. Lysne, J. Piris, A termination ordering for higher-order rewrite systems, in: J. Hsiang (Ed.), *Rewriting Techniques and Applications*, Lecture Notes in Computer Science, Springer, Berlin, 1995, pp. 26–40.
- [22] R. Mayr, Konfluenz bei Termersetzungssystemen auf λ -Termen, Master's Thesis, Institut für Informatik, Technische Universität München, 1994.
- [23] D. Müller, A logic programming language with lambda-abstraction, function variables, and simple unification, *J. Logic Comput.* 1(4) (1991) 497–536.
- [24] F. Müller, Confluence of the lambda-calculus with left-linear algebraic rewriting, *Inform. Process. Lett.* 41 (1992) 293–299.
- [25] T. Nipkow, Higher-order critical pairs, in: 6th IEEE Symp. Logic in Computer Sci., IEEE Computer Soc. Press, Silver Spring, MD, 1991, pp. 342–349.
- [26] T. Nipkow, Functional unification of higher-order patterns, in: 8th IEEE Symp. Logic in Computer Science, IEEE Computer Soc. Press, Silver Spring, MD, 1993, pp. 64–74.
- [27] T. Nipkow, Orthogonal higher-order rewrite systems are confluent, in: M. Bezem, J.F. Groote (Eds.), *Proc. Int. Conf. Typed Lambda Calculi and Applications*, Lecture Notes in Computer Science, vol. 664, Springer, Berlin, 1993, pp. 306–317.
- [28] M.J. O'Donnell, Computing in Systems Described by Equations, Lecture Notes in Computer Science, vol. 58, Springer, Berlin, 1977.
- [29] V. van Oostrom, Confluence for abstract and higher-order rewriting, Ph.D. Thesis, Vrije Universiteit, Amsterdam, 1994.
- [30] V. van Oostrom, F. van Raamsdonk, Comparing combinatory reduction systems and higher-order rewrite systems, in: J. Heering, K. Meinke, B. Möller, T. Nipkow (Eds.), *Higher-Order Algebra, Logic and Term Rewriting*, Lecture Notes in Computer Science, vol. 816, Springer, Berlin, 1994, pp. 276–304.
- [31] V. van Oostrom, F. van Raamsdonk, Weak orthogonality implies confluence: the higher-order case, in: A. Nerode (Ed.), *Logical Foundations of Computer Science*, Lecture Notes in Computer Science, vol. 813, Springer, Berlin, 1994, pp. 379–392.
- [32] V. Padovani, On equivalence classes of interpolation equations, in: M. Dezani-Ciancaglini, G. Plotkin (Eds.), *Typed Lambda Calculi and Applications*, Lecture Notes in Computer Science, vol. 902, Springer, Berlin, 1995, pp. 335–349.
- [33] L.C. Paulson, Isabelle: A Generic Theorem Prover, Lecture Notes in Computer Science, vol. 828, Springer, Berlin, 1994.
- [34] J. van de Pol, Termination proofs for higher-order rewrite systems, in: J. Heering, K. Meinke, B. Möller, T. Nipkow (Eds.), *Higher-Order Algebra, Logic and Term Rewriting*, Lecture Notes in Computer Science, vol. 816, Springer, Berlin, 1994, pp. 305–325.
- [35] J. van de Pol, H. Schwichtenberg, Strict functionals for termination proofs, in: M. Dezani-Ciancaglini, G. Plotkin (Eds.), *Typed Lambda Calculi and Applications*, Lecture Notes in Computer Science, vol. 902, Springer, Berlin, 1995, pp. 350–364.
- [36] Z. Qian, Linear unification of higher-order patterns, in: M.-C. Gaudel, J.-P. Jouannaud (Eds.), *Proc. Coll. Trees in Algebra and Programming*, Lecture Notes in Computer Science, vol. 668, Springer, Berlin, 1993, pp. 391–405.
- [37] F. van Raamsdonk, Confluence and superdevelopments, in: C. Kirchner (Ed.), *Rewriting Techniques and Applications*, Lecture Notes in Computer Science, vol. 690, Springer, Berlin, 1993, pp. 168–182.
- [38] F. van Raamsdonk, Confluence and normalization for higher-order rewriting, Ph.D. Thesis, Vrije Universiteit, Amsterdam, 1996.
- [39] S. Stenlund, Combinators, λ -Terms, and Proof Theory, D. Reidel, Dordrecht, 1972.
- [40] M. Takahashi, λ -calculi with conditional rules, in: M. Bezem, J.F. Groote (Eds.), *Typed Lambda Calculi and Applications*, Lecture Notes in Computer Science, vol. 664, Springer, Berlin, 1993, pp. 406–417.
- [41] D.A. Wolfram, *The Clausal Theory of Types*, Cambridge Tracts in Theoretical Computer Science, vol. 21, Cambridge Univ. Press, Cambridge, 1993.