



**HAL**  
open science

# Confluence in UnTyped Higher-Order Theories by means of Critical Pairs

Gaspard Férey, Jean-Pierre Jouannaud

► **To cite this version:**

Gaspard Férey, Jean-Pierre Jouannaud. Confluence in UnTyped Higher-Order Theories by means of Critical Pairs. 2021. hal-03126102v1

**HAL Id: hal-03126102**

**<https://inria.hal.science/hal-03126102v1>**

Preprint submitted on 1 Feb 2021 (v1), last revised 14 Sep 2021 (v3)

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Confluence in UnTyped Higher-Order Theories by means of Critical Pairs

GASPARD FÉREY, LSV, ENS de Paris-Saclay, France

JEAN-PIERRE JOUANNAUD, LSV, ENS de Paris-Saclay, France

---

User-defined higher-order rewrite rules are becoming a standard in proof assistants based on intuitionistic type theory. This raises the question of proving that they preserve the properties of beta-reductions for the corresponding type systems. We develop here techniques that reduce confluence proofs to the checking of various forms of critical pairs for higher-order rewrite rules extending beta-reduction on pure lambda-terms. The present paper concentrates on the case where rewrite rules are left-linear and critical pairs can be joined without using beta-rewrite steps. The other two cases will be addressed in forthcoming papers.

Additional Key Words and Phrases: Lambda calculus, Church-Rosser property, Confluence, Decreasing diagrams, Ancestor peaks, Critical peaks

## ACM Reference format:

Gaspard Férey and Jean-Pierre Jouannaud. 2017. Confluence in UnTyped Higher-Order Theories by means of Critical Pairs. *Proc. ACM Program. Lang.* 1, 1, Article 1 (January 2017), 35 pages.  
<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

---

## 1 INTRODUCTION

The two essential properties of a type theory, consistency and decidability of type checking, follow from three simpler ones: type preservation, strong normalization and confluence. In dependent type theories however, confluence is often needed to prove type preservation and strong normalization, making all three properties interdependent if termination is used in the confluence proof. This circularity can be broken in two ways: by proving all properties together within a single induction [10]; or by proving confluence on untyped terms first, and then successively type preservation, confluence on typed terms, and strong normalization. We develop the latter way here, focusing on untyped confluence.

In Coq [3] and Agda [21], rewrite rules introduced by the user originate from the definition of inductive types of some form. They satisfy a precise format which has been well studied, for which confluence is always satisfied. But Agda and Coq developers have recently announced the development of new versions that would allow user-defined rewrite rules [6], and several on-going developments in Agda are already using this facility. User defined rewrite rules are indeed at the heart of DEDUKTI [8], which has been mostly used so far as a logical framework, user-defined rewrite rules originating then from complex higher-order encodings for which inductive types do not provide enough flexibility. Investigating the preservation of confluence by user-defined rewrite rules in  $\lambda$ -calculus appears therefore to be very timely.

Let  $\mathcal{R}$  be the set of user-defined rewrite rules, which come in addition to the  $\beta$ -rule. The rewrite relation underlying the type theory is then generated by both  $\mathcal{R}$  and the  $\beta$ -rule. Studying the meta-theory of such a type theory implies investigating the confluence property of  $\beta \cup \mathcal{R}$ . Since our prime target is DEDUKTI, which is a calculus without extentionality, we do not consider here the  $\eta$  rule in addition to the  $\beta$ -rule.

---

2017. 2475-1421/2017/1-ART1 \$15.00

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

50 There are three main tools for analyzing confluence of a rewrite relation: Newman's Lemma [19],  
51 Hindley-Rosen's Lemma [12], and van Oostrom's Theorem which generalizes both previous  
52 ones [26]. Since beta rewrites are non-terminating in pure lambda calculus, Newman's Lemma  
53 does not apply. And if the rules have non-trivial critical pairs, then Hindley-Rosen's Lemma does  
54 not apply either. Even the subtle use of Hindley-Rosen's Lemma allowing for development-closed  
55 critical pairs [25] is too restrictive for practical usage. The way out is the use of van Oostrom's  
56 decreasing diagrams [24]. The fact that beta reductions do not terminate for pure lambda terms is  
57 no obstacle since we do not rely on termination for showing confluence when using decreasing  
58 diagrams. A further reason for considering pure lambda terms is that it is then easy to deduce  
59 confluence for any type system, including dependent type systems, for which the rules enjoy type  
60 preservation.

61 Van Oostrom's theorem is abstract, its application to term rewriting relations conceals many  
62 difficulties. Further, neither confluence nor termination are preserved by adding a confluent and  
63 terminating set of rewrite rules to a  $\lambda$ -calculus. A counter-example to termination in the simply  
64 typed  $\lambda$ -calculus is given in [22]. Numerous counter-examples to confluence in the pure  $\lambda$ -calculus  
65 are given in [15]. The problem we address is by no means simple.

66 Our untyped  $\lambda$ -calculus is intended to fit with the implementation of DEDUKTI. The format of  
67 rules is classical: left-hand sides must be patterns [17, 18], which are extremely useful for describing  
68 encodings of a type theory in another, a keen application to us. Considering untyped terms requires  
69 some adaptations of the usual higher-order rewriting definitions. In particular, we shall consider that  
70 the meta-variables used in rules have an arity which is not fixed, but bounded. These adaptations  
71 impact unification: we shall precisely analyze unification of linear untyped patterns and show the  
72 existence of most general unifiers computable in linear time.

73 Our main contribution is that a set  $\mathcal{R}$  of rules which is terminating on the set of pure  $\lambda$ -terms and  
74 whose left-hand sides are linear patterns, preserves confluence of the  $\lambda$ -calculus if its critical pairs  
75 are joinable by using rules in  $\mathcal{R} \cup M\eta$ , where  $M\eta$  is the extensionality rule for the meta-variables.

76 This result is then demonstrated with the example of a theory of global states due to Plotkin  
77 and Power [23], whose rules have overlapping linear higher-order patterns as left-hand sides. The  
78 confluence of this example had indeed been shown already in [11]. Hamana shows first that the  
79 (simply) typed rules are terminating, then that the higher-order critical pairs are joinable, using  
80 Newman's Lemma to deduce its confluence. Our method applies independently of the typing system,  
81 hence we can deduce that the same example remains confluent when using a dependently typed  
82 discipline. As we shall see, the critical pairs are not development closed, hence this result does not  
83 follow from existing techniques.

84 The applicability of our result to Church-style dependent type theories in which abstraction  
85 need carrying a type as first argument is not direct, it requires arguing that the confluence of the  
86 theory follows from the confluence of its Curry-style type-free version. We will justify the use of  
87 our techniques in this case.

88 We recall the notion of labeled reduction and decreasing diagram in Section 2, and describe our  
89 higher-order setting in Section 3. Matching and unification of untyped patterns and basic properties  
90 of untyped higher-order rewriting are considered in Section 4. Local rewriting peaks are analyzed  
91 in Section 5. Our confluence result is stated and proved in Section 6. Related work and applicability  
92 of our results is discussed in Section 7. Concluding remarks come last.

## 2 LABELED REDUCTIONS

### 2.1 Reductions

Given a binary relation  $\longrightarrow$  on terms, called *rewriting*, we use:  $\longleftarrow$  for its inverse,  $\Longrightarrow$  for its parallelization, allowing one to rewrite at once several subterms of a given term, when none is a subterm of another, and  $\longleftrightarrow$ ,  $\longrightarrow$ , and  $\longleftrightarrow$ , for its closures by, respectively, symmetry; reflexivity and transitivity (called *derivation*); and reflexivity, symmetry and transitivity (called *convertibility*).

A term  $s$  is *in normal form* if there is no  $t$  such that  $s \longrightarrow t$ . We define a *normal form* for an arbitrary term  $s$  as a term  $t$  in normal form, denoted by  $s \downarrow$ , such that  $s \longrightarrow t$ . *Termination* (or *strong normalization*) is the impossibility of an infinite rewriting sequence  $t_0 \longrightarrow t_1 \longrightarrow \dots \longrightarrow t_n \longrightarrow \dots$  and guarantees the existence of normal forms for every term. A *local peak* is a triple of terms  $(s, u, t)$  such that  $s \longleftarrow u \longrightarrow t$ ;  $u$  is the *source* and  $s, t$  are its *reducts*. Two terms  $s, t$  are *joinable* if  $s \longrightarrow v \longleftarrow t$  for some  $v$ , making the peak  $s \longleftarrow u \longrightarrow t$  *joinable*. The property that every two convertible terms are joinable is called *confluence* (or *Church-Rosser*). Confluence guarantees the unicity of normal forms for every term.

When rewriting terminates, it is well-known that the joinability of all local peaks implies the confluence property, this is the so-called Newman's lemma. When it does not, it is then necessary to strengthen joinability, this is the rôle of *decreasing diagrams*.

### 2.2 Decreasing diagrams

In the following, we consider rewrite relations all of whose elementary steps are equipped with a label belonging to some well-founded set whose strict partial order is denoted by  $\triangleright$ .

*Definition 2.1 (Decreasing diagram [24]).* Given a labeled relation  $\longrightarrow$  on an abstract set, we denote by  $DS(m, n)$  the set of *decreasing rewrite sequences* of the form  $u \xrightarrow{\delta} v$  or  $u \xrightarrow{\gamma} s \xrightarrow{n} t \xrightarrow{\delta} v$  such that the labels in  $\gamma$  and  $\delta$  are strictly smaller than, respectively,  $m$ , and,  $m$  or  $n$ . The steps labeled by  $\gamma, n$  and  $\delta$ , are called the *side steps*, *facing step* and *middle steps* of the decreasing sequence, respectively.

Given a local peak  $v \xleftarrow{m} u \xrightarrow{n} w$ , a *decreasing (rewrite) diagram* is a pair of derivations from  $v$  and  $w$  to some common term  $t$ , belonging to  $DS(m, n)$  and  $DS(n, m)$ , respectively.

Decreasing rewrite diagrams are represented in Figure 2 and abbreviated as DDs. Note that a facing step of a decreasing diagram may be missing, its side steps are then absorbed by the middle ones. A more general notion of decreasing diagram appears in [26], we won't need it here.

**THEOREM 2.2 ([24]).** *A labeled relation is Church-Rosser if all its local peaks have DDs.*

Van Oostrom's theorem generalizes to rewriting modulo an equational theory  $=_E$  in which case  $\triangleright$  must be compatible with the equational theory [13]. This is for example the case when rewriting terms of the  $\lambda$ -calculus for which  $\alpha$  conversion is built-in and must be compatible with the chosen definition of reduction over terms. Equational steps are considered to have a minimal label and often ignored.

## 3 HIGHER-ORDER REWRITING

Given an *untyped* lambda calculus generated by a vocabulary made of three pairwise disjoint sets, a signature  $\mathcal{F}$  of *function symbols*, a set  $\mathcal{X}$  of *variables*, and a set  $\mathcal{Z}$  of *meta-variables*, we are interested in the calculus  $\lambda\mathcal{F}$ , whose reduction relation extends the  $\beta$ -rule of the underlying  $\lambda$ -calculus by a set  $\mathcal{R}$  of user-defined rewrite rules built over that vocabulary. Were we to analyze

the confluence of  $\mathcal{R}$  alone, then, the situation would be similar to the first-order case, at least when left-hand sides of rules are patterns [17]. Unfortunately, confluence of  $\mathcal{R} \cup \beta$  cannot, in general, be deduced from the confluence of its two components.

### 3.1 Terms in $\lambda\mathcal{F}$

$\lambda\mathcal{F}$  is a mix of the pure lambda-calculus and Klop's combinatory reduction systems [15], that fits with DEDUKTI [8]. Terms are those of the pure lambda calculus enriched with  $\mathcal{F}$ -headed terms of the form  $f(\bar{u})$  with  $f \in \mathcal{F}$  and meta-terms of the form  $Z[\bar{v}]$  with  $Z \in \mathcal{Z}$ . Only variables can be abstracted over. Elements of the vocabulary have arities, denoted by vertical bars as in  $|f|$  or  $|Z|$ . Variables have arity zero. The grammar of terms is the following:

$$u, v := x \in \mathcal{X} \mid (u v) \mid \lambda x.u \mid f(\bar{u}) \mid Z[\bar{v}] \quad \text{where } f \in \mathcal{F}, |\bar{u}| = |f|, Z \in \mathcal{Z} \text{ and } |\bar{v}| \leq |Z|$$

Following usage, we don't duplicate parentheses, writing  $f(x y)$  for  $f((x y))$ , and use brackets instead of parentheses for meta-variables. It is sometimes convenient to name the head symbol of the expression  $(s t)$ : we use the symbol @ for that purpose throughout the paper. We use the small letters  $f, g, h, \dots$  for function symbols and  $x, y, z, \dots$  for variables, and reserve capital letters  $X, Y, Z, \dots$  for meta-variables. When convenient, a small letter like  $x$  may denote any variable in  $\mathcal{X} \cup \mathcal{Z}$ . We use the notation  $|\_|$  to denote various quantities besides arities, such as the length of a list, the size of an expression or the cardinality of a set. Given a list  $\bar{u}$ ,  $\bar{u}[m..n]$  denotes the finite sublist  $u_m, \dots, u_n$ . Commas in lists are sometimes omitted.

Unlike function symbols and Klop's meta-variables, meta-variables here have an arity which is not fixed, but bounded, a handy feature used in DEDUKTI which has several objectives. First of all, the number of dependent arrows in a dependent type  $T$  is not fixed, it may depend upon, say, the value of a natural number this type depends upon. However, any occurrence of a meta-variable of type  $T$  used in a rewrite rule must have a finite number of arguments, the maximum of these numbers can then be taken as the arity of that meta-variable. This also allows to have implicit arguments in meta-variables, a feature that eases code development. Another use of this facility in DEDUKTI is to speed up computations by avoiding type-checking terms along rewriting derivations. The pattern matching algorithm, as we shall see in Section 4, requires using the arity of meta-variables instead of their type. Finally, verifying in DEDUKTI that rewrite rules preserve types is based, as expected, on solving type equality constraints, which in turn requires inferring the arities of the meta-variables that occur in those rules.

Arities extend naturally to all terms, writing  $ar(t)$  for the *arity* of an arbitrary term  $t$ , by induction on their structure:  $ar(\lambda x.t) = 1 + ar(t)$ ,  $ar(X[\bar{t}]) = |X| - |\bar{t}|$  and  $ar((u v)) = ar(x) = ar(f(\bar{u})) = 0$ .

Positions in terms are words over the natural numbers (assuming  $|\lambda x.\_| = 1$ ), using  $\cdot$  for concatenation,  $\Lambda$  for the empty word,  $P/p$  for  $\{q : p \cdot q \in P\}$ ,  $\leq_p$  for the prefix order (*above*),  $\geq_p$  for its inverse (*below*),  $<_p$  and  $>_p$  for their strict parts, and  $p\#q$  for  $(p \not\leq_p q \wedge p \not\geq_p q)$  (*parallel*). An order  $>$  on positions is extended to sets of positions as follows:  $P > Q$  iff  $\forall p \in P \exists q \in Q$  such that  $p > q$ .

Given a term  $M$ , we use  $\mathcal{P}os(M)$ ,  $\mathcal{V}\mathcal{P}os(M)$ ,  $\mathcal{M}\mathcal{P}os(M)$  for its sets of: all positions, positions of free variables, positions of meta-variables, respectively; and  $\mathcal{V}ar(M)$ ,  $\mathcal{B}\mathcal{V}ar(M)$  and  $\mathcal{M}\mathcal{V}ar(M)$  for its sets of: free variables, bound variables and meta-variables, respectively. A term  $M$  is *ground* if  $\mathcal{V}ar(M) = \emptyset$ , *closed* if  $\mathcal{M}\mathcal{V}ar(M) = \emptyset$ , And linear if  $|\mathcal{M}\mathcal{P}os(M)| = |\mathcal{M}\mathcal{V}ar(M)|$ .

Given now a term  $M$  and a position  $p \in \mathcal{P}os(M)$ , we use  $M(p)$  for its symbol at position  $p$ ,  $M|_p$  for the subterm of  $M$  at position  $p$ ,  $M[N]_p$  for the term obtained by replacing in  $M$  the subterm  $M|_p$  by the term  $N$ . The latter notation extends to sets  $P$  of parallel positions in  $M[\bar{N}]_P$  or  $M[N]_P$  in case all terms in  $\bar{N}$  are identical to the term  $N$ . This use of brackets in the meta-language of terms is reminiscent of its use in the term language, namely in  $Z[\bar{v}]$ . Both kinds of brackets may occur in

a same expression, as long as the replacement bracket is indexed by a position or set of positions. We sometimes use the notation  $u[v]_p$  to stipulate that the subterm of  $u$  at position  $p$  is the term  $v$ . The context is supposed to help discriminating between these different uses of the bracket notation. We say that a variable  $x \in \mathcal{B}\mathcal{V}ar(M)$  is *bound above*  $p$  in  $M$  if  $M|_q = \lambda x.N$  for some  $q <_{\mathcal{P}} p$ .

### 3.2 Substitutions

A *substitution* is a mapping from a finite set of variables and meta-variables to terms, written as  $\sigma = \{x_1 \mapsto M_1, \dots, x_n \mapsto M_n\}$ , or  $\sigma = \{\bar{x} \mapsto \bar{M}\}$ , where  $ar(M_i) \geq ar(x_i)$ , which extends to an *arity-preserving* and *capture-avoiding* homomorphism on terms as defined next. Let  $Dom(\sigma) = \{x_1, \dots, x_n\} \subseteq X \cup \mathcal{Z}$  be the *domain* of  $\sigma$  while  $Ran(\sigma) = \bigcup_{i=1}^n \mathcal{V}ar(M_i) \cup \mathcal{M}\mathcal{V}ar(M_i)$  is its *image*. By convention, we write  $\sigma(x_i)$  for  $M_i$ . As in  $\lambda$ -calculus, substituting in terms requires renaming bound variables to avoid capturing free ones. Using post-fixed notation, applying a substitution to a term is therefore defined as follows:  $x_i\sigma = M_i$ ;  $y\sigma = y$  if  $y \notin Dom(\sigma)$ ;  $f(\bar{t})\sigma = f(\bar{t}\sigma)$ ;  $(u\ v)\sigma = (u\sigma\ v\sigma)$ ; and  $(\lambda x.u)\sigma = \lambda z.u(\{x \mapsto z\}\sigma)$  where  $z \notin Dom(\sigma) \cup Ran(\sigma)$  ( $x$  needs not be renamed in  $z$  if it already satisfies the condition). Assuming now  $Z \mapsto \lambda \bar{x}.s \in \sigma$ , where  $s$  is not an abstraction, the additional rule for meta-variables, inspired from Klop's definition of substitutions in the case of fixed arities [15], is as follows:

- Case 1:  $|\bar{u}| = m \leq n = |\bar{x}|$ , then  $Z[\bar{u}]\sigma = \lambda \bar{x}[m+1..n].s\{\bar{x}[1..m] \mapsto \bar{u}\sigma\}$ ,

hence delaying the replacement of those arguments of  $Z$  that are missing.

- Case 2:  $m > n$ , then (since  $ar(s) \geq |Z| - |\bar{x}| \geq |\bar{u}| - |\bar{x}| > 0$ )  $s = Y[\bar{t}]$  and  $\bar{u} = \bar{v}\bar{w}$  with  $|\bar{v}| = n$  and  $|Y| - |\bar{t}| \geq |\bar{w}|$ , in which case  $Z[\bar{u}]\sigma = Y[\bar{t}\{\bar{x} \mapsto \bar{v}\sigma\}, \bar{w}\sigma]$ .

The result  $t\sigma$  of substituting the term  $t$  is called an *instance* (of  $t$ ) and the operation itself an *instantiation*. The substitution  $\sigma$  is *ground* (resp., *closed*) when so are all  $M_i$ 's.

A substitution  $\sigma$  can be *restricted to* or *deprived from* (meta-)variables in some set  $V$ , written  $\sigma|_V$  and  $\sigma_{\setminus V}$  respectively.

*Example 3.1.* Let  $s = \lambda x.(X[Y, f(x, Y)]\ g(Y))$  and  $\sigma = \{X \mapsto \lambda yzz'.h(z', z), Y \mapsto h(a, a)\}$  where  $|X| = 3$  and  $|Y| = 0$ . Then,  $s\sigma = \lambda x.(\lambda z'.h(z', f(x, h(a, a)))\ g(h(a, a)))$ . Note how the first two arguments of  $X$  are directly substituted in  $\sigma(X)$  while the missing argument creates a  $\beta$ -redex.

Let now  $\tau = \{X \mapsto \lambda z.Z[z], Y \mapsto a\}$  where  $|Z| = 3$ . Then  $s\tau = \lambda x.(Z[a, f(x, a)]\ g(a))$ .

Our definition of substitution ensures the following important property:

LEMMA 3.2. *Arities are non-decreasing under substitution:*  $\forall M, \sigma\ (ar(M\sigma) \geq ar(M))$ .

PROOF. Easy for term constructors. For meta-variables,  $|Z| \leq n + ar(s)$  by definition.

Case 1 :  $ar(Z[\bar{u}]) = |Z| - m \leq n + ar(s) - m \leq (n - m) + ar(s\{\bar{x}[1..m] \mapsto \bar{u}\sigma\}) \leq ar(Z[\bar{u}]\sigma)$  by induction hypothesis.

Case 2 :  $ar(Z[\bar{u}]) = |Z| - m \leq n - m + ar(s) = -|\bar{w}| + |Y| - |\bar{t}| = ar(Z[\bar{u}]\sigma)$ .  $\square$

Rewriting terms extends to substitutions as expected, while substitutions are extended to sequences of terms and to substitutions in the natural way, if  $X\sigma = \lambda \bar{x}.u$  then  $X(\sigma\tau) = \lambda \bar{x}.u\tau$ , keeping the same post-fixed notation. We write  $u\sigma\tau = (u\sigma)\tau = u(\sigma\tau)$ .

Terms compare in the *subsumption* order,  $t \geq s$  if  $t =_{\alpha} s\gamma$  for some substitution  $\gamma$ , whose equivalence  $\doteq$  defines a bijection between the free variables of  $s, t$ , and strict part  $\triangleright$  is well-founded. This order extends to substitutions in the natural way. Given a set  $\Sigma$  of substitutions, a *most general* substitution is a substitution  $\theta \in \Sigma$ , when it exists, such that any closed substitution  $\sigma \in \Sigma$  is an instance of  $\theta$ , that is  $\sigma = \theta\tau$  for some  $\tau$ . This perhaps unusual definition of a most general substitution which captures closed instances only instead of all instances, will suffice for our needs and simplify some statements and proofs.

Given a term  $u$  and a list  $P = \{p_i\}_{i=1}^{i=n}$  of parallel positions in  $u$ , we define the term obtained by *splitting*  $u$  along  $P$  as  $\underline{u}_P = u[Z_1[\bar{x}_1]]_{p_1} \dots [Z_n[\bar{x}_n]]_{p_n}$  ( $u$  is cut below  $P$ ) and its associated substitution by  $\bar{u}^P = \{Z_i \mapsto \lambda \bar{x}_i. u|_{p_i}\}_{i=1}^{i=n}$  ( $u$  is cut above  $P$ ), where, for all  $i \in [1, n]$ ,  $\bar{x}_i$  is the list of all variables of  $u|_{p_i}$  bound in  $u$  above  $p_i$  and  $Z_i$  is a fresh meta-variable of arity (exactly)  $|\bar{x}_i|$ . The definition of substitution for meta-variables ensures that  $\underline{u}_P \bar{u}^P = u$ , which justifies writing  $u = u[u|_P]$  as a familiar shorthand. Note the two kinds of brackets in  $\underline{u}_P$ .

Our use of splitting in this paper will be systematic unless it alters readability for no good reason. This invention permitted by Klop's notion of meta-variable, is the only technique we know of which allows to manipulate terms with binders safely, in case renaming of variables needs to take place independently in a term and in its context, as will often be the case here. We do not claim for originality here, although we cannot tell who made use of it first.

### 3.3 Functional reductions

Arrow signs used for rewriting will often be decorated, below by a name, and above by a position  $p$  or set of positions  $P$ , as in  $s \xrightarrow[R]{P} t$  or by a property that this position or set of positions satisfies, as in

$u \xrightarrow[R]{\geq p P} v$  and in  $v = u \downarrow_{R}^{\geq p P}$  ( $v$  is obtained from  $u$  by normalizing its subterms  $v|_{p \in P}$  with  $\mathcal{R}$ ).

Two different kinds of reductions coexist in our calculus, functional and higher-order reductions. Both are meant to operate on closed terms. However, rewriting open terms will sometimes be needed, in which case rewriting is intended to rewrite all their closed instances at once.

*Functional reduction* is the relation on terms generated by the rule  $(\lambda x. u \ v) \xrightarrow[\beta_\alpha]{} u\{x \mapsto v\}$ . The usually omitted  $\alpha$ -index stresses that renaming bound variables, called  $\alpha$ -conversion, is built-in.

The particular case for which  $v$  is a variable is denoted by  $\beta^0$ , a notation introduced by Miller. Instantiating a  $\beta^0$ -step may yield a full  $\beta$ -step. For example,  $s = (\lambda x. (\lambda y. g(y) \ x) \ a) \xrightarrow[\beta^0]{1:1} (\lambda x. g(x) \ a) \xrightarrow[\beta]{\Delta} g(a)$

while  $s \xrightarrow[\beta]{\Delta} (\lambda y. g(y) \ a) \xrightarrow[\beta]{\Delta} g(a)$ . This is our main motivation for using Klop's notion of substitution for meta-variables, whose benefits will appear in the next subsection.

We will also use a particular case of extensionality, for meta-variables only:  $\lambda z. X[\bar{u}, z] =_{M\eta} X[\bar{u}]$  if  $|X| > |\bar{u}|$ ,  $z$  fresh. When oriented from left to right,  $M\eta$  is terminating and confluent. It has another important property:  $M\eta$ -steps disappear when taking ground instantiations, a key property for us.

**LEMMA 3.3.** *Let  $\bar{u}$  be a sequence of closed terms,  $z$  a variable such that  $z \notin \text{Var}(\bar{u})$ ,  $X$  a meta-variable such that  $|X| > |\bar{u}|$  and  $\sigma$  a closed substitution. Then  $\lambda z. X[\bar{u}, z]\sigma = X[\bar{u}]\sigma$ .*

**PROOF.** Since  $\sigma$  is closed, and  $|X| \geq |\bar{u}| + 1$ , then  $\{X \mapsto \lambda \bar{x} z. v\} \in \sigma$  with  $|\bar{x}| = |\bar{u}|$ . We then get  $(\lambda z. X[\bar{u}, z])\sigma = \lambda z. (X[\bar{u}, z])\sigma = \lambda z. v\{\bar{x} \mapsto \bar{u}\sigma, z \mapsto z\} = \lambda z. v\{\bar{x} \mapsto \bar{u}\sigma\} = X[\bar{u}]\sigma$ .  $\square$

**COROLLARY 3.4.** *Let  $u, v$  be terms such that  $u =_{M\eta} v$  and  $\sigma$  a closed substitution. Then,  $u\sigma = v\sigma$ .*

### 3.4 Higher-order reductions

*Higher-order reductions* result from rules whose left-hand sides are higher-order patterns in Miller's or Nipkow's sense [17], although they need not be typed:

**Definition 3.5 (Pattern).** A *pre-redex* of arity  $n$  in a term  $L$  is an unapplied meta-term  $Z[\bar{x}]$  whose arguments  $\bar{x}$  are  $n$  pairwise distinct variables. A *pre-pattern* is a  $\beta$ -normal term all of whose meta-variables occur in pre-redexes. A *pattern* is a ground pre-pattern which is neither a pre-redex nor an abstraction, that is, is not headed by a meta-variable or  $\lambda$ .

It is important to assume, as does Nipkow, that patterns are  $\beta$ -normal. Note also that patterns are ground: free variables are not needed, one can use meta-variables of arity zero instead. Pre-patterns play an important rôle in pattern matching and unification, since patterns must be deconstructed.

Erasing types from a Nipkow's pattern yields a pattern in our sense, since his pre-redexes being of base type, they cannot be applied. This restriction is not important until later, when we address the question of matching and unification of patterns.

Observe that pre-redexes in pre-patterns occur at parallel positions, whose set plays a key rôle:

*Definition 3.6 (Fringe).* The fringe  $F_L$  of a pattern  $L$  is the set of parallel positions of its pre-redexes. We denote by  $\mathcal{FPos}(L) = \{p \in \mathcal{Pos}(L) : F_L \not\leq p\}$  the set of *functional positions* of the pre-pattern  $L$ . We also define  $F_\beta = \{1, 2\}$  for convenience. A pattern has a non-empty set of functional positions.

*Example 3.7.* The term  $L = f(\lambda xyz.g(X[x, y, z], X[x, y]))$  is a pattern. Its pre-redexes are the terms  $X[x, y, z]$  and  $X[x, y]$ . Its fringe is the set  $F_L = \{1^5, 1^4 \cdot 2\}$ . The term  $f(\lambda xyz.g(X[x, y, z]) (a X))$  is also a pattern, its fringe is the set  $\{1^6, 2^2\}$ . Terms  $f(\lambda x.X[x, x])$ ,  $f(X[a])$ ,  $f(X[Y])$ , and  $f(X Y)$ , are no patterns.

Note that the set of functional positions coincides with the usual notion for first-order terms. Since patterns are ground terms, for all pre-redexes  $Z[\bar{x}] = L|_p$  at position  $p \in F_L$  in the pattern  $L$ , the variables  $\bar{x}$  are all locally bound above  $p$  in  $L$ .

We can now define higher-order rules and rewriting:

*Definition 3.8 (Rule).* A (higher-order) rule is a triple  $i : L \rightarrow R$ , whose (possibly omitted) index  $i$  is a natural number, left-hand side  $L$  is a pattern, and right-hand side  $R$  is a ground  $\beta$ -normal term such that  $\mathcal{MVar}(R) \subseteq \mathcal{MVar}(L)$ . The rule is *left-linear* if  $L$  is linear and *right-linear* if  $R$  is linear.

So, rules are pairs of (specific) ground terms. They may have meta-variables, but don't admit free variables. This allows to clearly separate the object language (which has no meta-variables), from the meta-language (which has meta-variables). Rules, critical pairs and splittings belong to the meta-language.

*Definition 3.9 (Higher-order untyped rewriting).* Given a term  $u$ , a position  $p \in \mathcal{Pos}(u)$ , and a rule  $i : L \rightarrow R$  then  $u$  rewrites with  $i$  at  $p$ , written  $u \xrightarrow[p]{i} v$ , or  $u \xrightarrow[L \rightarrow R]{p} v$ , iff  $u|_p = L\gamma$  for some substitution  $\gamma$ , and  $v = u[R\gamma]_p$ . We write  $u \xrightarrow[\mathcal{R}]{p} v$  for  $\exists i \in \mathcal{R}, u \xrightarrow[p]{i} v$ .

Let's now make our splitting notations fully explicit. Whenever  $u \xrightarrow[p]{i} v$ , we have by definition:

- $\underline{u}_p = u[X[\bar{x}]]_p$  and  $\bar{u}^p = \{X \mapsto \lambda \bar{x}. u|_p\}$  with  $\bar{x}$  the variables bound above  $p$  in  $u$  and  $X$  a fresh meta-variable of arity  $|\bar{x}|$ .
- $u = \underline{u}_p \bar{u}^p = \underline{u}_p \{X \mapsto \lambda \bar{x}. u|_p\} = \underline{u}_p \{X \mapsto \lambda \bar{x}. L\gamma\}$
- $v = \underline{u}_p \{X \mapsto \lambda \bar{x}. R\gamma\}$ , hence  $\underline{v}_p = \underline{u}_p$ ,  $\bar{v}^p = \{X \mapsto \lambda \bar{x}. R\gamma\}$  and  $v|_p = R\gamma$ .

*Example 3.10.* Let  $L = \text{der}(\lambda x.\text{times}(A, F[x])) \rightarrow \text{times}(A, \text{der}(\lambda y.F[y])) = R$  and  $\sigma$  be the identity substitution  $\{F \mapsto \lambda x.x\}$ . Then,  $L\sigma = \text{der}(\lambda x.\text{times}(2, x))$  and  $R\sigma = \text{times}(2, \text{der}(\lambda y.y))$ , hence  $\text{der}(\lambda x.\text{times}(2, x)) \rightarrow \text{times}(2, \text{der}(\lambda y.y))$ . However,  $\text{der}(\lambda x.\text{times}(x, x))$  is a normal form.

In sharp contrast with Nipkow [17], we observe that we do not need matching modulo  $\beta^0$ , since the corresponding  $\beta^0$ -steps are now hidden in Klop's definition of substitution for meta-variables. We however show in Section 7.1 that our main confluence result applies to Nipkow's definition: the use of Klop's definition of substitution for meta-variables can be seen as a technical choice.



### 3.6 The rewrite theory of global states

Our running example here will be Plotkin's and Power's theory of global states for a single location [23]. It is described by two types,  $\text{Val}$  for values and  $A$  for states, a unary operation  $\text{lk}$  for looking up a state, a binary operation  $\text{ud}$  for updating a state, and five higher-order rules which satisfy our format, the meta-variables having arities (unlike in the original article). First, the signature:

$$\text{lk} : (\text{Val} \rightarrow A) \rightarrow A \quad | \quad \text{ud} : \text{Val}, A \rightarrow A$$

$\text{lk}(\lambda v.t)$  looks up the state, binds its value to  $v$ , and continues with  $t$  while  $\text{ud}(v, t)$  updates the state to  $v$ , and continues with  $t$ . Types are given for a better understanding, they do not play any rôle here. Let us now give the rules, using  $U, V, W$  (resp.  $X$ ) (resp.  $Y$ ) (resp.  $T$ ) for meta variables of arity 0 (resp. 1) (resp. 2) (resp. 3). We shall reserve  $Z$  for fresh meta-variables of an arbitrary arity, when such meta-variables are needed. These meta-variables may appear primed when too many of a given arity are needed, as it will be the case when computing critical pairs. This convention will be followed in all examples, throughout the paper.

$$\begin{array}{lll} (ll) & \text{lk}(\lambda w.\text{lk}(\lambda v.Y[w, v])) & \rightarrow \text{lk}(\lambda v.Y[v, v]) & (ll) \\ (lu) & \text{lk}(\lambda v.\text{ud}(v, X[v])) & \rightarrow \text{lk}(\lambda v.X[v]) & | \quad \text{lk}(\lambda v.U) \rightarrow U & (l) \\ (ul) & \text{ud}(V, \text{lk}(\lambda v.X[v])) & \rightarrow \text{ud}(V, X[V]) & | \quad \text{ud}(U, \text{ud}(V, W)) \rightarrow \text{ud}(V, W) & (uu) \end{array}$$

This typed higher-order theory was studied by Hamana, who was interested in its confluence investigated with his Haskell-based analysis tool SOL [11]. Our presentation is a simplification of Hamana's, in which one rule was actually superfluous. Note that all rules are left-linear.

In this example, all meta-variables take a constant number of arguments, equal to their arity. Using our meta-variables with a bounded arity, we can reformulate this system by eliminating its  $\eta$ -expansions:

$$\begin{array}{lll} (ll) & \text{lk}(\lambda w.\text{lk}(Y[w])) & \rightarrow \text{lk}(\lambda v.Y[v, v]) & (ll) \\ (lu) & \text{lk}(\lambda v.\text{ud}(v, X[v])) & \rightarrow \text{lk}(X) & | \quad \text{lk}(\lambda v.U) \rightarrow U & (l) \\ (ul) & \text{ud}(V, \text{lk}(X)) & \rightarrow \text{ud}(V, X[V]) & | \quad \text{ud}(U, \text{ud}(V, W)) \rightarrow \text{ud}(V, W) & (uu) \end{array}$$

We could of course, eliminate the  $\eta$ -expansions from the left-hand sides, and keep them in the right-hand sides. We will see that the precise formulation of the rules, when there are many possible variations, impacts their confluence properties.

## 4 PATTERN MATCHING AND UNIFICATION OF LINEAR PATTERNS

Computing critical pairs, which play a key rôle in the analysis of overlapping peaks, requires unifying two left-hand sides of rules, that is two patterns, while checking their joinability requires rewriting arbitrary terms, possibly having meta-variables, which requires in turn pattern matching arbitrary terms against left-hand sides of rules, that is patterns. In both cases, the only variables to be instantiated are meta-variables belonging to patterns. For unification, the range of the unifying substitutions cannot contain free variables, since there are none in patterns, while for matching, the range can be arbitrary. Both algorithms, unification and pattern matching are described by rewrite rules operating on equational problems, which are conjunctions of equations between pre-patterns (for unification) or between a pre-pattern and a term (for matching), a format invariant under rule application, as will be shown.

### 4.1 Matching and unification problems

The coming definitions do not assume patterns to be linear, but the unification and matching rewrite rules do. We discuss briefly how to remove this restriction in conclusion.

442 *Definition 4.1.* A (matching or unification) *equational problem* is a conjunction of elementary  
 443 equations. An *elementary equation* is either the constant  $\perp$  or is of the form  $u = v$  in which  $u$  is a  
 444 pre-pattern and  $v$  is either a pre-pattern (unification case), or an arbitrary term (matching case).

445 Given the problem  $P = \bigwedge_i u_i = v_i$ , we denote by  $LMVar(P)$  its set of left meta-variables  
 446  $\bigcup_i MVar(u_i)$ , and by  $rVar(P)$  and  $rMVar(P)$  its respective sets of right variables  $\bigcup_i Var(v_i)$   
 447 and meta-variables  $\bigcup_i MVar(v_i)$ .

448 We now define solutions and unifiers of an equational problem, the unifiers being representations  
 449 of their solutions.  
 450

451 *Definition 4.2 (Solutions and unifiers).* Let  $P$  be the equational problem  $\bigwedge_i u_i = v_i$ .

452 A substitution  $\gamma$  is a *matching solution* of  $P$  if  $Dom(\gamma) = LMVar(P)$  and  $\forall i (u_i \gamma =_\alpha v_i)$ .

453 A ground substitution  $\gamma$  is a *unifier* (or *unifying substitution*) of  $P$  if  $\forall i u_i \gamma =_\alpha v_i \gamma$ .

454 A ground closed unifier of domain  $MVar(P)$  is called a *unifying solution* of  $P$ .

455 The equational problems containing  $\perp$  have no matching solution nor unifier.

456 We use the word solution without qualification when which kind is meant does not matter or is  
 457 made clear by the context.  
 458

459 Unlike the first-order case, unification problems need not have free variables, since we can always  
 460 replace free variables by meta-variables of arity zero. Further, variables that become free in pre-  
 461 patterns by pulling out their binder should obviously not be instantiated. Therefore, assuming that  
 462 unifiers are ground is no restriction. Similarly, the domain of a matching solution of an equational  
 463 problem  $P$  can be restricted without loss of generality to  $LMVar(P)$ . The range of the matching  
 464 solution is accordingly restricted to  $rMVar(P) \cup rVar(P)$ , a property that will be used without  
 465 notice. Unifiers satisfy a similar, but more subtle property.

466 **LEMMA 4.3.** *If a unification problem  $P$  has a unifier  $\gamma$ , then it has a more general unifier  $\sigma$  such  
 467 that  $Dom(\sigma) \subseteq MVar(P)$ .*

468 **PROOF.** *By induction on the number  $n$  of meta-variables in  $Dom(\gamma) \setminus MVar(P)$ .*

- 469 • *Base case:*  $Dom(\gamma) \subseteq MVar(P)$ , then  $\sigma = \gamma$ .
- 470 • *Induction step:* let  $X$  be a meta-variable such that  $X \mapsto s \in \gamma$  and  $X \notin MVar(P)$ . Let  $\tau$  be the  
 471 substitution of domain  $Dom(\gamma) \setminus X$  such that for all  $Y \in Dom(\tau)$ ,  $Y \mapsto \lambda \bar{y}.t\{X \mapsto X'\} \in \tau$   
 472 iff  $Y \mapsto \lambda \bar{y}.t \in \gamma$ . Then,  $\gamma = \tau\{X' \mapsto X, X \mapsto s\}$ , hence  $\tau$  is more general than  $\gamma$ . Now,  
 473  $u\tau = u\gamma\{X \mapsto X'\} = v\gamma\{X \mapsto X'\} = v\tau$ , hence  $\tau$  is a unifier of  $P$ . By induction hypothesis,  
 474 there exists a unifier  $\sigma$  smaller than  $\tau$ , hence smaller than  $\gamma$  by transitivity of subsumption,  
 475 which satisfies our requirements.  
 476

## 477 4.2 Matching and unification rules

478 *Definition 4.4.* A unification problem  $\mathcal{P}$  is *linear* if no meta-variable occurs more than once in  $\mathcal{P}$ .  
 479 A matching problem  $\mathcal{P}$  is *linear* if no meta-variable occurs more than once in the left-hand sides of  
 480 the elementary equations of  $\mathcal{P}$ .  
 481

482 In the sequel, we will usually omit  $=_\alpha$ , and also restrict ourselves to linear equational problems.  
 483 This is enough for our use of unification to solving equations between left-hand sides of rules, since  
 484 we restrict ourselves to left-linear rewrite rules.

485 Describing the matching and unification rules requires the following preliminary definition:

486 *Definition 4.5.* A free variable  $x \in \mathcal{X}$  is *protected* in a pre-pattern  $u$  if all its occurrences in  $u$  belong  
 487 to a pre-redex of  $u$ , that is, take place below  $F_u$ . We denote by  $\mathcal{U}Var(u)$  the set of unprotected  
 488 variables of  $u$ .  
 489

491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539

<i>Dec-Fun</i>	$f(\bar{u}) = f(\bar{v}) \longrightarrow \bigwedge_{i=1}^{i= f } u_i = v_i$	if $f \in \mathcal{F} \cup \mathcal{X} \cup \{\@\}$
<i>Dec-Abs</i>	$\lambda x.u = \lambda y.v \longrightarrow u\{x \mapsto z\} = v\{y \mapsto z\}$	with $z$ fresh
<i>Conflict</i>	$f(\bar{u}) = g(\bar{v}) \longrightarrow \perp$	if $f, g \in \mathcal{F} \cup \mathcal{X} \cup \{\@\, \lambda\}$ and $f \neq g$
<i>Meta-Abs</i>	$X[\bar{x}] = \lambda y.v \longrightarrow X[\bar{x}y] = v$	if $X \in \mathcal{Z}$ and $ X  >  \bar{x} $
<i>Fail-Arity</i>	$X[\bar{x}] = u \longrightarrow \perp$	if $ X  -  \bar{x}  > ar(u)$
<i>Fail-Protect</i>	$X[\bar{x}] = u \longrightarrow \perp$	if $\exists z \in \mathcal{V}ar(u)$ s.t. $z \notin (\bar{x} \cup r\mathcal{V}ar(P_0))$
<i>Fail-Arity</i>	$X[\bar{x}] = f(\bar{u}) \longrightarrow \perp$	if $ X  >  \bar{x}  \wedge f \in \{\@\} \cup \mathcal{F} \cup \bar{x}$
<i>Fail-Protect</i>	$X[\bar{x}] = u \longrightarrow \perp$	if $\exists z \in \mathcal{U}\mathcal{V}ar(u)$ s.t. $z \notin \bar{x}$
<i>Swap</i>	$u = Y[\bar{y}] \longrightarrow Y[\bar{y}] = u$	if $u$ is not a pre-redex
<i>Flip</i>	$X[\bar{x}] = Y[\bar{y}] \longrightarrow Y[\bar{y}] = X[\bar{x}]$	if $ X  -  \bar{x}  >  Y  -  \bar{y} $
<i>Drop</i>	$X[\bar{x}] = u[Y[\bar{y}]]_q \longrightarrow$ $X[\bar{x}] = u[Z[\bar{z}]]_q \wedge Y[\bar{y}] = Z[\bar{z}]$	if $\left\{ \begin{array}{l} \bar{y} \notin \bar{x} \cup \mathcal{B}\mathcal{V}ar(u) \\  X  =  \bar{x}  \text{ if } u(\Lambda) \in \mathcal{F} \cup \{\@\, \lambda\} \\ \mathcal{U}\mathcal{V}ar(u) \subseteq \bar{x} \\  Y  -  \bar{y}  \geq  X  -  \bar{x}  \text{ if } q = \Lambda \end{array} \right.$ with $\left\{ \begin{array}{l} \bar{z} = \bar{y} \cap (\bar{x} \cup \mathcal{B}\mathcal{V}ar(u)) \\ Z \text{ fresh s.t. }  Z  =  Y  -  \bar{y}  +  \bar{z}  \end{array} \right.$

Fig. 1. Matching and unification rules for linear equational problems

For an example,  $x$  is protected in  $f(g(X[x]), X)$ . It is not protected in  $f(g(X[x]), x)$  because of its second occurrence. Protected variables can be eliminated from a term by appropriately instantiating its meta-variables, while unprotected variables cannot be eliminated. An important known observation to be justified later is that elementary unification problems for which a free variable occurs unprotected on one side, and does not occur at all on the other side have no solution.

Pattern matching and unification are described by the rewrite rules given in Figure 1. Rules written in black apply to both matching and unification problems. Rules written in green apply to matching problems only, while rules in blue apply to unification problems only. Note that the constant  $\perp$  is absorbing, a black rule that will remain implicit.

The initial problem to be matched or unified is denoted by  $P_0$ . Rule *Fail-Protect* refers to  $P_0$ .

Apart from *Meta-Abs*, the first subset of four black rules treats equations between expressions which are not pre-redexes. Those equations can be decomposed or fail, depending on the respective root symbol of the left-hand and right-hand sides. These rules are just the same as those for first-order unification, with the exception, of course, of *Dec-Abs*. Note that *Dec-Fun* removes equations of the form  $x = x$ , and that we abuse our notations in *Conflict* by assimilating  $\lambda$  with a symbol. The rôle of *Meta-Abs* is to ensure that the arity condition for meta-variables is met by the substitution that will be obtained if the algorithm succeeds.

The second subset of rules is made of failure rules, two green followed by two blue. Failure may come from an arity violation (first or third rule), or from the existence of an unprotected variable (in the case of matching, a variable occurring free in the right-hand side of an equation is unprotected if it does not occur in the left-hand side or in the initial problem).

There are three remaining blue rules for unification that constitute the third subset. When the right-hand side of an equation is a pre-redex, it is moved to the left by *Swap* if the left-hand side is not a pre-redex, or else by *Flip* if the left-hand side is lacking more (implicit) arguments than the

right-hand side. *Drop* applies to equations with a pre-redex on the left, in case some free variables, occur protected in the right-hand side while they do not occur on the left-hand side and must therefore be eliminated. Note that the first condition of this rule ensures that it is only fired if the set  $\bar{y} \setminus (\bar{x} \cup \mathcal{B}Var(u))$  of such variables is non-empty. The following three conditions, in the order they are listed, postpone the application of *Drop* until *Fail-Arity*, *Fail-Protect* and *Flip*, in this order, can no longer apply.

In the particular case where  $q = \wedge$ ,  $|X| - |\bar{x}| = |Y| - |\bar{y}|$  and  $\bar{x} \subseteq \bar{y}$  then *Drop* applies to  $X[\bar{x}] = Y[\bar{y}]$  and produces  $X[\bar{x}] = Z[\bar{x}] \wedge Y[\bar{y}] = Z[\bar{x}]$  which could be improved in an implementation with a special instance of *Drop* to produce  $Y[\bar{y}] = X[\bar{x}]$  only, as does *Flip*.

Note that the set of rules can be easily transformed into a deterministic algorithm as no two rules apply to the same equation, except (i) two different (blue or green) failure rules, which does not hamper determinism since they deliver the same result,  $\perp$ ; (ii) *Meta-Abs* and each of the failure rules but *Fail-Arity*, in which case the considered failure rule still applies after *Meta-Abs*.

### 4.3 Examples

Before proving properties of these matching and unification rules, we show below examples of reduction sequences using the unification rules that are useful for the reader's understanding. All four examples transform a unification problem into one in normal form.

*Example 4.6.* Let's illustrate some rules, using meta-variables according to our convention.

$$\begin{aligned}
 f(\lambda z.X[z]) &= f(\lambda z.z) \xrightarrow{\text{Dec-Fun}} \lambda z.X[z] = \lambda z.z \xrightarrow{\text{Dec-Abs}} X[z] = z \\
 f(\lambda z.X) &= f(\lambda z.z) \xrightarrow{\text{Dec-Fun}} \lambda z.X = \lambda z.z \xrightarrow{\text{Dec-Abs}} X = z \xrightarrow{\text{Fail-Protect}} \perp \\
 f(\lambda y.f(U)) &= f(X) \xrightarrow{\text{Dec-Fun}} \lambda y.f(U) = X \xrightarrow{\text{Swap}} X = \lambda y.f(U) \xrightarrow{\text{Meta-Abs}} X[y] = f(U) \\
 f(Y) &= f(\lambda y.f(U)) \xrightarrow{\text{Dec-Fun}} Y = \lambda y.f(U) \xrightarrow{\text{Meta-Abs}} Y[y] = f(U) \xrightarrow{\text{Fail-Arity}} \perp \\
 T &= \lambda y.Y[y] \xrightarrow{\text{Meta-Abs}} T[y] = Y[y] \xrightarrow{\text{Flip}} Y[y] = T[y] \\
 Y[z] &= \lambda x.T[y, z] \xrightarrow{\text{Meta-Abs}} Y[z, x] = T[y, z] \xrightarrow{\text{Drop}} Y[z, x] = Z[z] \wedge T[y, z] = Z[z]
 \end{aligned}$$

(where  $Z$  is a fresh variable of arity 1)

*Drop* applies here to an elementary equation in which there are extra variables on both sides, eliminating, perhaps surprisingly, both problems at once: the two generated equations have a pre-redex on the left-hand side which contains all free variables occurring on the other side.

We now show examples of matching and unification problems that will be useful later when computing the critical pairs of the theory of global states. We will not do all computations needed later on, only a few interesting ones originating from the first or second versions of that theory:

*Example 4.7.* We consider here two matching problems based on the second set of rules of the theory of global states. Firstly, matching the term  $\text{lk}(\lambda w.\text{ud}(w, X'[w]))$  against the left-hand side  $\text{lk}(\lambda w.\text{ud}(w, X[w]))$  of rule (lu):

$$\begin{aligned}
 \text{lk}(\lambda w.\text{ud}(w, X[w])) &= \text{lk}(\lambda w.\text{ud}(w, X'[w])) \xrightarrow{\text{Dec-Fun}} \lambda w.\text{ud}(w, X[w]) = \lambda w.\text{ud}(w, X'[w]) \\
 &\xrightarrow{\text{Dec-Abs}} \text{ud}(w, X[w]) = \text{ud}(w, X'[w]) \xrightarrow{\text{Dec-Fun}} w = w \wedge X[w] = X'[w] \xrightarrow{\text{Dec-Fun}} X[w] = X'[w]
 \end{aligned}$$

Secondly, matching the term  $\text{lk}(\lambda v.\text{lk}(Y[v]))$  against the left-hand side of rule (ll):

$$\begin{aligned}
 \text{lk}(\lambda v.\text{lk}(Y[v])) &= \text{lk}(\lambda v.\text{lk}(v)) \xrightarrow{\text{Dec-Fun}} \lambda v.\text{lk}(Y[v]) = \lambda v.\text{lk}(v) \\
 &\xrightarrow{\text{Dec-Abs}} \text{lk}(Y[v]) = \text{lk}(v) \xrightarrow{\text{Dec-Fun}} Y[v] = v
 \end{aligned}$$

This equation is in normal form as the left-hand side is a pre-redex of the same arity, 1, as the right-hand side whose set of free variables is a subset of the pre-redex arguments.

*Example 4.8.* We consider here first, three unification problems using rules of the first set of rules of the theory of global states. The first unifies the left-hand sides of rules (ll) and (l):

$$\begin{aligned} \text{lk}(\lambda w. \text{lk}(\lambda v. Y[w, v])) = \text{lk}(\lambda w. U) &\xrightarrow{\text{Dec-Fun}} \lambda w. \text{lk}(\lambda v. Y[w, v]) = \lambda w. U \xrightarrow{\text{Dec-Abs}} \\ \text{lk}(\lambda v. Y[w, v]) = U &\xrightarrow{\text{Swap}} U = \text{lk}(\lambda v. Y[w, v]) \xrightarrow{\text{Drop}} U = \text{lk}(\lambda v. Z[v]) \wedge Y[w, v] = Z[v] \end{aligned}$$

(where  $Z$  is a fresh variable of arity 1)

Now, we consider the unification of the left-hand sides of rules (lu) and (l):

$$\begin{aligned} \text{lk}(\lambda v. \text{ud}(v, X[v])) = \text{lk}(\lambda v. U) &\xrightarrow{\text{Dec-Fun}} \lambda v. \text{ud}(v, X[v]) = \lambda v. U \xrightarrow{\text{Dec-Abs}} \\ \text{ud}(v, X[v]) = U &\xrightarrow{\text{Swap}} U = \text{ud}(v, X[v]) \xrightarrow{\text{Fail-Protect}} \perp \end{aligned}$$

since  $v$  occurs unprotected as first argument of  $\text{ud}$  in  $\text{ud}(v, X[v])$ , making unification fail, which therefore rules out a potential critical pair.

Finally unification of the left-hand sides of (l) with a subterm of the left-hand side of (ul):

$$\begin{aligned} \text{lk}(\lambda v. U) = \text{lk}(\lambda v. X[v]) &\xrightarrow{\text{Dec-Fun}} \lambda v. U = \lambda v. X[v] \xrightarrow{\text{Dec-Abs}} U = X[v] \xrightarrow{\text{Drop}} U = Z \wedge X[v] = Z \end{aligned}$$

(where  $Z$  is a fresh variable of arity 0)

We now carry out the same three computations using the second set of rules of the theory of global states. For the first:

$$\begin{aligned} \text{lk}(\lambda w. \text{lk}(Y[w])) = \text{lk}(\lambda w. U) &\xrightarrow{\text{Dec-Fun}} \lambda w. \text{lk}(Y[w]) = \lambda w. U \xrightarrow{\text{Dec-Abs}} \text{lk}(Y[w]) = U \xrightarrow{\text{Swap}} U = \text{lk}(Y[w]) \\ \xrightarrow{\text{Drop}} U = \text{lk}(Z) \wedge Y[w] = Z &\quad \text{(where } Z \text{ is a fresh variable of arity 0)} \end{aligned}$$

The second computation is exactly the same. We move to the third:

$$\text{lk}(\lambda v. U) = \text{lk}(X) \xrightarrow{\text{Dec-Fun}} \lambda v. U = X \xrightarrow{\text{Swap}} X = \lambda v. U \xrightarrow{\text{Meta-Abs}} X[v] = U$$

We observe that the computations from the second set of rules are identical in the failure case, but slightly different in the other two. We will see later the impact on the unifying substitutions.

#### 4.4 Soundness

We now go on studying the matching/unification rules. First, we verify that the rules operate on equational problems (linearity will be considered later):

**LEMMA 4.9.** *Assume that an equational problem  $P$  rewrites to  $P'$  by using one of the matching/unification rules. Then,  $P'$  is an equational problem.*

**PROOF.** All rules preserve the property that pre-redexes are never applied. □

A difficulty is that not all rules preserve the set of solutions: consider the matching equation  $\lambda x. X = \lambda x. x$ , in which  $|X| = 0$ , which yields the new equation  $X = x$  by *Dec-Abs*. This equation has for matching solution the substitution  $\gamma = \{X \mapsto x\}$ , which is not a solution of the original equation since  $(\lambda x. X)\gamma = \lambda y. x$  for some fresh  $y$  while  $(\lambda x. x)\gamma = \lambda x. x$ . This leads us to the following notion of soundness:

**Definition 4.10.** Given two problems  $P$  and  $P_0$ , a matching solution  $\gamma$  of  $P$  is *adequate* for  $P_0$  if  $\text{Dom}(\gamma) \subseteq \mathcal{M}\mathcal{V}\text{ar}(P_0)$  and  $\text{Ran}(\gamma) \subseteq r\mathcal{M}\mathcal{V}\text{ar}(P_0) \cup r\mathcal{V}\text{ar}(P_0)$ .

This problem does not pop up with unification because unifiers and unifying solutions are ground, and therefore,  $\gamma = \{X \mapsto x\}$  is not a solution of either  $\lambda x.X = \lambda x.x$  or  $X = x$ .

*Definition 4.11.* We say that a matching rule is *sound*, given a problem  $P_0$ , if it preserves the matching solutions that are adequate for  $P_0$ , and that a unification rule is sound if it preserves all unifying solutions.

Obviously, preservation of all matching solutions implies soundness of a matching rule.

LEMMA 4.12. Dec-Fun and Conflict preserve all solutions.

LEMMA 4.13. Dec-Abs is sound.

PROOF. We carry out the proof for the case of unification.  $\gamma$  is a solution of  $\lambda x.u = \lambda x.v$  iff  $(\lambda x.u)\gamma = (\lambda x.v)\gamma$  iff  $(\lambda z.u\{x \mapsto z\})\gamma = (\lambda z.v\{x \mapsto z\})\gamma$  where  $z$  is a fresh variable, iff  $(u\{x \mapsto z\})\gamma = (v\{x \mapsto z\})\gamma$ , since  $z$  does not need be renamed in order to push the substitution  $\gamma$  inside the abstractions (using the assumption that  $z \notin \mathcal{R}an(\gamma)$ ).  $\square$

We continue with the arity-checking rules, using the property that arities are non-decreasing under substitution:

LEMMA 4.14. Meta-Abs preserves all solutions.

PROOF. Assume that  $|X| > |\bar{x}|$ . Let be the elementary unification problems  $X[\bar{x}] = \lambda y.v$  and  $X[\bar{x}y] = v$  Without loss of generality, we assume that  $y \notin \bar{x}$ . Let  $|X| = n$  and  $|\bar{x}| = m$ , hence  $n > m$ .

Let  $\sigma = \{X \mapsto \lambda \bar{x}[1..p].s\}$  where  $p \leq m$  and  $s$  is not an abstraction. Then,  $s$  must be headed by a meta-variable  $Y$  such that  $|Y| \geq n - p > 0$  to satisfy the arity constraint of a substitution, hence  $\sigma = \{X \mapsto \lambda \bar{x}[1..p].Y[\bar{t}]\}$  for some  $\bar{t}$ . Hence  $X[\bar{x}]\sigma$  is then a term headed by  $Y$ , which cannot be equal modulo renaming to  $\lambda y.v$ . Therefore,  $\sigma$  cannot be a solution of the equation  $X[\bar{x}] = \lambda y.v$ .

A substitution  $\sigma$  solution of that equation must therefore be of the form  $\{X \mapsto \lambda \bar{x}\lambda y.s\}$ , where  $ar(s) \geq 0$ . Then,  $X[\bar{x}]\sigma = s\sigma$ , hence  $\sigma$  is a solution of the equation  $X[\bar{x}] = \lambda y.v$  iff  $\sigma$  is a solution of the equation  $X[\bar{x}y] = v$ , which ends the proof.  $\square$

LEMMA 4.15. The elementary matching problem  $X[\bar{x}] = u$  has no solution if  $ar(X[\bar{x}]) > ar(u)$ .

PROOF. By non-decreasingness Lemma 3.2,  $ar(X[\bar{x}]\sigma) \geq ar(X[\bar{x}]) > ar(u)$ .  $\square$

LEMMA 4.16. Assume that  $|X| > |\bar{x}|$  and  $f \in \mathcal{F} \cup \{\@\} \cup \mathcal{X}$ . Then, the elementary unification problem  $X[\bar{x}] = f(\bar{u})$  has no solution.

PROOF. Again,  $ar(X[\bar{x}]\sigma) \geq ar(X[\bar{x}]) > 0 = ar(f(\bar{u}\sigma)) = ar(f(\bar{u})\sigma)$ , since  $x\sigma = x$  when  $f \in \mathcal{X}$ .  $\square$

We now move to the case where extra variables occur in right-hand sides, whether protected or not, starting with the two failure rules.

LEMMA 4.17. Fail-Protect is sound.

PROOF. Let  $u$  be a term containing a variable  $z \notin \bar{x} \cup \mathcal{V}ar(P_0)$ . A solution  $\gamma$  for the matching problem  $X[\bar{x}] = u$  must contain  $X \mapsto \lambda \bar{x}.u$ , which is not adequate for  $P_0$ .  $\square$

LEMMA 4.18. The elementary unification problem  $X[\bar{x}] = u$  has no solution if  $z \in \mathcal{U}\mathcal{V}ar(u) \setminus \bar{x}$ .

PROOF. Let  $X[\bar{x}] = u$  be an elementary unification problem such that  $z \in \mathcal{U}\mathcal{V}ar(u) \setminus \bar{x}$ , and let us assume it has a solution  $\gamma$  such that  $X \mapsto \lambda \bar{y}.v \in \gamma$  with  $ar(v) \geq |X| - |\bar{y}|$ . By definition of a solution,  $z \notin \mathcal{V}ar(X[\bar{x}]\gamma)$  on the one hand, and  $z \notin \mathcal{D}om(\gamma)$ , hence  $z\gamma = z$ , thus  $z \in \mathcal{V}ar(u\gamma)$  on the other hand. Since  $X[\bar{x}]\gamma$  and  $u\gamma$  have different sets of free variables, no  $\alpha$ -renaming can make them equal, contradicting our assumption that  $\gamma$  is a solution.  $\square$

LEMMA 4.19. Let  $E$  be the elementary unification problem  $X[\bar{x}] = u[Y[\bar{y}]]_q$  and  $P$  the unification problem  $X[\bar{x}] = u[Z[\bar{z}]]_q \wedge Y[\bar{y}] = Z[\bar{z}]$ , where  $\bar{z} = \bar{y} \cap (\bar{x} \cup \mathcal{B}\mathcal{V}ar(u))$  and  $Z$  is a fresh variable of arity  $|Y| - |\bar{y}| + |\bar{z}|$ . Then  $\gamma$  is a solution of  $P$  iff  $\gamma \upharpoonright_Z$  is a solution of  $E$ .

PROOF. Assume first that  $\gamma$  is a unifier of  $P$ . Then,  $X[\bar{x}]\gamma = u[Z[\bar{z}]]_q\gamma = u\gamma[Z[\bar{z}]\gamma]_q$  by definition of a substitution and the fact that  $u[Z[\bar{z}]]_q$  is a pre-pattern. Since  $Z[\bar{z}]\gamma = Y[\bar{y}]\gamma$ , it follows that  $X[\bar{x}]\gamma = u[Z[\bar{z}]]_q\gamma = u[Y[\bar{y}]]_q\gamma = u[Y[\bar{y}]]_q\gamma$ , hence  $\gamma \upharpoonright_Z$  unifies  $E$ .

Conversely, let  $\gamma = \{X \mapsto \lambda\bar{x}'.w, Y \mapsto \lambda\bar{y}'.w'\}$  be a unifier of  $E$  such that  $Z \notin \text{Dom}(\gamma)$ . We can extend  $\gamma$  as  $\gamma'$  by defining  $\gamma' = \gamma \cup \{Z \mapsto \lambda\bar{z}.Y[\bar{y}]\gamma\}$ . Then,  $Z[\bar{z}]\gamma' = Y[\bar{y}]\gamma = Y[\bar{y}]\gamma'$ , hence  $\gamma'$  unifies  $Y[\bar{y}] = Z[\bar{z}]$ . Further,  $\gamma'$  unifies the equation  $X[\bar{x}] = u[Z[\bar{z}]]_q$  by the same token as before, hence  $\gamma'$  unifies  $P$ .  $\square$

We can now conclude:

LEMMA 4.20. The matching/unification rules are terminating and sound.

PROOF. Termination of the matching rules is clear. For unification, we interpret an equational problem by the multiset of interpretations of its elementary equations, so that it is enough to show that the interpretation of each elementary equation generated by a unification rule is strictly less than the interpretation of its left-hand side. An elementary equation  $u = v$  is interpreted by the quadruple  $\langle m, n, p, q \rangle$ , where  $m$  is the size of the equation from which all pre-redexes have been removed,  $n = 1$  if  $u$  is not a pre-redex otherwise 0, and  $p = 1$  if  $ar(u) > ar(v)$  otherwise 0, and  $q$  is the number of variables occurrences in  $v$ . It is easy to see that all rules but the last three generate elementary equations whose interpretation has a strictly smaller first component. Now, *Swap*, *Flip*, *Drop* decrease respectively their second, third, fourth component without changing their previous ones. In the case of *Drop*, this follows from the easy property that  $\bar{z} \subseteq \bar{y}$ .

Soundness follows from: from Lemma 4.12 for *Dec-Fun*, Lemma 4.13 for *Dec-Abs*, Lemma 4.14 for *Meta-Abs*; Lemma 4.15 for *Fail-Arity*; Lemma 4.17 for *Fail-Protect*; Lemma 4.16 for *Fail-Arity*; Lemma 4.18 for *Fail-Protect*; commutativity of equality for *Swap* and *Flip*; and Lemmas 4.19 and 4.3 for *Drop*.  $\square$

## 4.5 Solved forms

We now give the characterization of equational problems in normal form for those rules:

*Definition 4.21 (Solved forms).*

- (1) An equation  $u = v$  is in *arity-solved form* if  $u$  is a pre-redex  $X[\bar{x}]$  such that  $ar(v) \geq |X| - |\bar{x}|$ ;
- (2) An equational problem is in *solved form* if it is either the constant  $\perp$ , or a conjunction  $\bigwedge X_i[\bar{x}_i] = v_i$  of equations in *arity-solved form* such that for all  $i$ ,  $\mathcal{V}ar(v_i) \subseteq \bar{x}_i$  and for all  $i, j$ ,  $X_i \notin \mathcal{M}\mathcal{V}ar(v_j)$  and for all  $i \neq j$ ,  $X_i \neq X_j$ .

LEMMA 4.22.

- (1) An equation is in *arity-solved form* iff it is irreducible by all rules but *Fail-Protect*, *Fail-Protect*, *Drop* and *Meta-Abs*;
- (2) *Drop* and *Meta-Abs* preserve *arity-solved forms*.

PROOF. (1) The only if case being clear, let us assume that no rule other than *Fail-Protect*, *Fail-Protect*, *Drop* and *Meta-Abs* can apply to  $u = v$ . Necessarily  $u = X[\bar{x}]$  otherwise one of the *Dec* rules, *Conflict* or *Swap* rules would apply. Assuming  $|\bar{x}| < |X|$ , then  $v$  must be some pre-redex  $Y[\bar{y}]$  otherwise *Meta-Abs*, *Fail-Arity* or *Fail-Arity* would apply. Because *Flip* does not apply by assumption, we conclude that  $|X| - |\bar{x}| \leq |Y| - |\bar{y}|$ , hence  $u = v$  is in *arity-solved form*.

(2) The case of *Meta-Abs* follows from the definition of arity. Consider now *Drop*. If  $q = \wedge$ , then  $|X| - |\bar{x}| \leq |Y| - |\bar{y}| = |Z| - |\bar{z}|$  and the first generated equation is in arity-solved form. Otherwise  $|X| = |\bar{x}|$ . The first generated equation is in arity-solved form, as was the input equation. The second generated equation is in arity-solved form because  $|Z| - |\bar{z}| = |Y| - |\bar{y}|$ .  $\square$

LEMMA 4.23. *All rules preserve the following two properties of an unification problem P:*

- (1) *For all equations  $u = v \in P$  not in arity-solved form, all meta-variables in  $\mathcal{MVar}(u) \cup \mathcal{MVar}(v)$  are linear in P.*
- (2) *For all equations  $X[\bar{x}] = v \in P$  in arity-solved form, the meta-variable X is linear in P and for all pre-redexes  $Z[\bar{z}]$  in v, either the meta-variable Z is linear in P or  $\bar{z} \subseteq \bar{y} \cup \mathcal{BVar}(v) \cup \bar{x}$ .*

PROOF. All rules but *Meta-Abs*, *Fail-protect* and *Drop* preserve linearity of elementary equations, hence both properties (1) and (2) are preserved in that case. The case of *Meta-Abs* and *Fail-Protect* is clear. We are left with *Drop* which operates on equations in arity-solved form. From (2), non-linear meta-variables in right-hand sides are applied to locally bound or left-hand side variables, which prevents the application of *Drop* in that case. On the other, any application of *Drop* to linear meta-variable produces equations that satisfy the property since  $\bar{z} \subseteq \bar{y}$  and  $\bar{z} \subseteq \bar{x} \cup \mathcal{BVar}(v)$ .  $\square$

LEMMA 4.24. *Let P be a linear (matching or unification) equational problem: its normal form is in solved form.*

PROOF. Let Q be the normal form of P. By Lemma 4.22 (1), all equations in Q are in arity-solved form. By Lemma 4.23 (2), pre-redexes in their left-hand sides are linear in Q. We are left proving that for all  $X[\bar{x}] = v \in Q$ ,  $\mathcal{Var}(v) \subseteq \bar{x}$ . Assume it is not the case, and let  $y \in \mathcal{Var}(v) \setminus \bar{x}$ . Either y occurs protected and *Drop* applies, or else *Fail-Protect* applies.  $\square$

LEMMA 4.25. *Solved forms of equational problems are computed in linear time.*

PROOF. First, note that the matching and unification rules check a fixed number of symbols belonging to the head of the left-hand and right-hand side of each equation belonging to a given equational problem in turn. It is therefore enough to show that the total number of matching or unification steps is linear in the size of the problem. Finally, because meta-variables appear linearly in a given equational problem, it is enough to consider every elementary equation separately.

The set of rules common to unification and matching applies to an elementary equation  $u = v$  a number of times bounded by  $|\mathcal{FPos}(u)|$ , and yields a number of elementary equations whose whole size is bounded by  $|u| + |v|$ . Failure rules may apply only once. This concludes the case of matching, we continue with the remaining unification rules. *Swap* and *Flip* may apply only once to a given equation, and leave the size of the problem invariant. Finally, the conditions for applying *Drop* ensure that no other rule will ever apply after any sequence of *Drops*. Further the total number of applications of *Drop* to a given equation is bounded by the number of protected variables to be eliminated, hence by its size. This shows that the whole unification process is linear.  $\square$

#### 4.6 Matching solutions and unifiers

We are left extracting matching substitutions and most general unifiers from equational problems in solved form. The case of a matching solved form follows quite easily:

LEMMA 4.26. *A matching solved form  $P = \wedge_i X_i[\bar{x}_i] = v_i$  has a matching solution  $\text{match}(P) = \{X_i \mapsto \lambda \bar{x}_i. v_i\}_i$ , which is unique up to the equivalence  $=_{\alpha \cup M \eta}$ .*

PROOF. It is clear that  $\text{match}(P)$  is a matching solution. Let  $\gamma = \{X_i \mapsto \lambda \bar{z}_i. t_i\}_i$  be another matching solution. If  $|\bar{z}_i| \geq |\bar{x}_i|$ , then necessarily  $(X_i[\bar{x}_i])\gamma =_{\alpha} v_i = (X_i[\bar{x}_i])\sigma$ . As noted after the

definition of a matching solution,  $\mathcal{R}an(\gamma) \cap \bar{x} = \emptyset$ , hence  $\sigma(X_i) =_{\alpha} \gamma(X_i)$ . Otherwise  $\bar{x}_i = \bar{z}_i \bar{y}_i$  and  $t_i = Y[\bar{u}]$  such that  $(X_i[\bar{x}_i])\gamma = Y[\bar{u}, \bar{y}_i] =_{\alpha} v_i$ , hence  $\sigma(X_i) = \lambda \bar{z}_i \lambda \bar{y}_i . Y[\bar{u}, \bar{y}_i] =_{\alpha \cup M\eta} \gamma(X_i)$ .  $\square$

*Example 4.27.* We continue here the calculations of Example 4.7.

Consider the first matching solved form obtained,  $X[w] = X'[w]$ , whose corresponding matching solution is therefore  $\{X \mapsto \lambda w . X'[w]\}$ . We can therefore rewrite the term  $\text{lk}(\lambda w . ud(w, X'[w]))$  with rule (lu), resulting in the term  $\text{lk}(\lambda w . X'[w])$ .

Note that  $\{X \mapsto X'\}$  is another matching solution which is  $M\eta$ -equivalent to the previous one. Using that solution yields the reduct  $\text{lk}(X')$  which is again  $M\eta$ -equivalent to the previous one.

For the second matching problem, the reader can verify that the matching solution becomes  $\lambda v w . Y'[v, w]$  and therefore  $\text{lk}(\lambda v . \text{lk}(Y'[v]))$  rewrites to  $\text{lk}(\lambda v . Y'[v, v])$ .

**LEMMA 4.28.** *A unification solved form  $P = \wedge_i X_i[\bar{x}_i] = v_i$  has a most general unifier  $\text{mgu}(P) = \{X_i \mapsto \lambda \bar{x}_i . v_i\}_i$ , which is unique up to the equivalence  $=_{\alpha \cup M\eta}$ .*

**PROOF.** Let  $\sigma$  be  $\text{mgu}(P)$ . We need to show that  $\sigma$  is a unifier, that it is most general, and that it is unique up to equivalence.

By definition of a unification solved form,  $\mathcal{M}\mathcal{V}ar(v_i) \cap \mathcal{D}om(\sigma) = \emptyset$  and  $\mathcal{V}ar(v_i) \subseteq \bar{x}_i$ . Hence  $\sigma$  is ground as required, and  $X_i[\bar{x}_i]\sigma = v_i = v_i\sigma$ . Therefore,  $\sigma$  is a unifier of  $P$ .

Let  $\theta$  be a unifying solution. For all  $i$ ,  $X_i[\bar{x}_i]\theta = v_i\theta = (X_i[\bar{x}_i]\sigma)\theta = X_i[\bar{x}_i](\sigma\theta)$ . Since  $\theta$  is ground,  $X_i\theta = X_i(\sigma\theta)$ . And for all  $Y \in \mathcal{D}om(\theta) \setminus \mathcal{D}om(\sigma)$ ,  $Y\theta = (Y\sigma)\theta = Y(\sigma\theta)$ . Hence  $\theta = \sigma\theta$ .

Finally, let  $\tau$  be another most general unifier. We assume  $\mathcal{D}om(\tau) \subseteq \mathcal{L}\mathcal{M}\mathcal{V}ar(P)$  and prove  $\sigma =_{\alpha \cup M\eta} \tau$  from  $X_i[\bar{x}_i]\sigma = X_i[\bar{x}_i]\tau$  as in the proof of Lemma 4.26.  $\square$

*Example 4.29.* We continue here the calculations of Example 4.8. Consider first the two solved forms obtained when unifying the left-hand sides of rules belonging to the first set of rules of the theory of global states. First, for the rules (ll) and (l), we get the solved form  $U = \text{lk}(\lambda v . Z[v]) \wedge Y[w, v] = Z[v]$  where  $|Z| = 1$ . The most general unifier is therefore  $\{U \mapsto \text{lk}(\lambda v . Z[v]), Y \mapsto \lambda w v . Z[v]\}$ . Then, for the rules (lu) and (l), we get the solved form  $U = Z \wedge X[v] = Z$ , where  $|Z| = 1$ , hence the mgu is  $\{U \mapsto Z, X \mapsto \lambda v . Z\}$ .

Moving now to the unifications between the same rules of the second set, we got as solved forms  $U = \text{lk}(Z) \wedge Y[w] = Z$  which yields the mgu  $\{U \mapsto \text{lk}(Z), Y \mapsto \lambda w . Z\}$ , and  $X[v] = U$  which yields the mgu  $X \mapsto \lambda v . U$ . These mgus differ slightly: the first two are the same up to extensionality of the meta-variables; the other two differ in a different but more usual way, the use of an extra variable that can be dispensed with. Altogether, the second set of rules yields more economic mgus.

Pattern matching and unification of linear patterns is therefore quite easy: first, reduce the initial pattern matching problem  $L = u$ , or unification problem  $L|_p = L'|_{p'}$ , to a solved form. Then extract the matching substitution or the most general unifier from the solved form. Therefore:

**THEOREM 4.30.** *The matching problem results in a single matching substitution when it succeeds, computable in linear time and space. The unification problem results in a single (up to  $\alpha$ ) most general solution when it succeeds, computable in linear time and space.*

Note that there is no mention of types in this algorithm. A natural question is whether the most general solution is typed when two linear patterns get unified, and whether it coincides with the one obtained when unifying dependently typed linear patterns.

## 5 LOCAL PEAKS IN REWRITE THEORIES

Rewrite theories have two kinds of local ancestor peaks, *homogeneous* ones, between functional or higher-order reductions, and *heterogeneous* ones, which mix both kinds of reductions. We analyze

here which local ancestor peaks enjoy *decreasing diagrams for free*, and which do not. Some results in this section will be reused in forthcoming papers, those that do not rely on the left-linearity assumption for the rewrite rules, nor on orthogonal functional reductions.

### 5.1 Decreasing diagrams for free

A key property of plain first-order rewriting is that there are three possible kinds of local peaks depending on the respective positions of the rewrites that define them. This property generalizes trivially to higher-order rewrites with our definition of set of positions for patterns:

LEMMA 5.1. *Given terms  $s, t$  such that  $s \xleftarrow[p]{i:L \rightarrow R} u \xrightarrow[q]{j:G \rightarrow D} t$ , there are three possibilities: (i)  $p \# q$  (disjoint peak case); (ii)  $q \geq_{\mathcal{P}} p \cdot F_L$  or  $p \geq_{\mathcal{P}} q \cdot F_G$  (ancestor peak case); and (iii)  $p = q \cdot o$  with  $o \in \mathcal{FPos}(G)$  or  $q = p \cdot o$  with  $o \in \mathcal{FPos}(L)$  (overlapping peak case).*

In the case of plain rewriting, two non-overlapping rewrite steps issuing from a same term commute, a major component of any confluence proof. When the steps occur at disjoint positions, this property, which holds for any monotonic relation, remains true for rewriting modulo a theory, hence all disjoint peaks have decreasing diagrams for free. This is not the case, however, when the steps occur at positions such that one is an ancestor of the other, because the modulo part of the above rewrite may interact with the rewrite below. Our definition of higher-order rewriting, however, enjoys a similar property, because the fringe of a rewrite step protects positions below it.

In the coming lemma, “LAP” stands for *linear ancestor peak*, and “a” for *above*, the  $\beta$ -step being above a higher-order step. It applies to any higher-order rule, left-linear or not.

LEMMA 5.2 (LAP $\beta$ A). *Let  $u$  be a term,  $p, q \in \mathcal{Pos}(u)$  such that  $q \geq_{\mathcal{P}} p \cdot F_{\beta}$  and  $s \xleftarrow[p]{i:L \rightarrow R} u \xrightarrow[q]{j:G \rightarrow D} t$ . Then  $s \xrightarrow[Q]{j} \xleftarrow[p]{\beta} t$  for some set  $Q$  of parallel positions of  $s$  such that  $Q \geq_{\mathcal{P}} p$ .*

PROOF. By assumption,  $\underline{u}_p = \underline{s}_p = \underline{t}_p$ ,  $u|_p = (\lambda x.M N)$ , and  $s|_p = M\sigma$ , where  $\sigma = \{x \mapsto N\}$ . There are two cases:

The case where  $q = p \cdot 2 \cdot q'$  and  $t|_p = (\lambda x.M P)$  with  $N \xrightarrow[q']{j} P$ . This requires several  $j$ -steps at the parallel positions of  $x$  in  $M$ . Then  $s = u[M\sigma]_p \xrightarrow[p]{Q} u[M\{x \mapsto P\}]_p \xleftarrow[p]{\beta} u[(\lambda x.M P)]_p = t$ .

Otherwise,  $q = p \cdot 1^2 \cdot q'$ , that is,  $M|_{q'} = u|_q \xrightarrow[j]{} t|_q$ . Then,  $s = u[M\sigma[u|_q\sigma]_{q'}]_p$ . By Lemma 3.13,  $u|_q\sigma \xrightarrow[j]{} t|_q\sigma$ , hence, by Lemma 3.12,  $s \xrightarrow[j]{} u[M\sigma[t|_q\sigma]_{q'}]_p$ . On the other hand,  $t|_p = (\lambda x.P N)$ , where  $P = M[t|_q]_{q'}$ , therefore  $t = u[(\lambda x.P N)]_p \xrightarrow[p]{\beta} u[P\sigma]_p = u[M\sigma[t|_q\sigma]_{q'}]_p$ .  $\square$

The case of a local peak  $s \xleftarrow[p]{i} u \xrightarrow[q]{j} t$ , where the higher-order step with  $i : L \rightarrow R$  applies above another step belonging to  $\mathcal{R} \cup \beta$ , a situation called (LAP $\mathcal{R}$ a), is shown at Figure 3 (left). (LAP $\mathcal{R}$ a) does not apply to non-left-linear rules. Its proof requires an important preliminary result:

LEMMA 5.3 (PRESERVATION). *Let  $u \xrightarrow[p]{i:L \rightarrow R} v$  with  $L$  linear and  $q \in \mathcal{Pos}(u)$  such that  $q \geq_{\mathcal{P}} p \cdot F_L$ . Then  $\underline{u}_q = u[Z[\bar{z}]]_q \xrightarrow[p]{i} w$  for some  $w$ , where  $\bar{z}$  is the list of variables bound above  $q$  in  $u$ ,  $Z$  fresh,  $|Z| = |\bar{z}|$ , and  $v = w\bar{u}^q = w\{Z \mapsto \lambda \bar{z}.u|_q\}$ .*

PROOF. By definition of splitting,  $u = \tau\tau$ , where  $t = \underline{u}_q = u[Z[\bar{z}]]_q$  and  $\tau = \bar{u}^q = \{Z \mapsto \lambda \bar{z}.u|_q\}$ .

883 Since  $q \geq_P p \cdot F_L$ , then  $q = p \cdot o \cdot q'$ , where  $o \in F_L$  is the position of a pre-redex in  $L$ . Hence  
 884  $L|_o = X[\bar{x}]$  for some meta-variable  $X$  and variables  $\bar{x}$  bound above  $o$  in  $L$ .

885 By definition of higher-order rewriting,  $u|_p = L\gamma$  for some substitution  $\gamma$  and we call  $M = u|_p \cdot o$ .  
 886 We have  $M = (L\gamma)|_o = L|_o\gamma = X[\bar{x}]\gamma$ .

887 Let now  $\theta$  be the substitution identical to  $\gamma$  except for the meta-variable  $X$  for which  $\theta(X) =$   
 888  $\lambda\bar{x}.M[Z[\bar{z}]]_{q'}$ . We have  $X[\bar{x}]\theta\tau = M[u|_q]_{q'} = M[M|_{q'}]_{q'} = M = X[\bar{x}]\gamma$ , since  $L$  is linear, the only  
 889 occurrence of the meta-variable  $X$  in  $L$  is at position  $p \cdot o$  and therefore  $L\gamma = L\theta\tau$ .

890 Since  $L$  is linear, there is a single pre-redex containing the meta-variable  $X$ . As a consequence,  $L\theta =$   
 891  $u|_p[X[\bar{x}]\theta]_o = u|_p[(M[Z[\bar{z}]]_{q'})_o]u|_p[u|_p \cdot o[Z[\bar{z}]]_{q'}]_o = u|_p[Z[\bar{z}]]_{o \cdot q'}$ , and therefore  $\underline{u}_q = u[L\theta]_p$ . By  
 892 definition of higher-order rewriting,  $L\theta \xrightarrow{\Lambda} R\theta$ , and by monotonicity,  $\underline{u}_q \xrightarrow{p} u[R\theta]_p = w$ .  
 893

894 By definition of higher-order rewriting again,  $v = u[R\gamma]_p = u[R\theta\tau]_p = (u[R\theta]_p)\tau = w\bar{u}^q$ .  $\square$

895 As already said, (LAPRA) requires the linearity assumption.

897 LEMMA 5.4 (LAPRA). Let  $\mathcal{R}$  be a left-linear rewrite system,  $i : L \rightarrow R \in \mathcal{R}$ ,  $j \in \mathcal{R} \cup \beta$ ,  $u$  be a term, and  
 898  $p, q \in \text{Pos}(u)$  such that  $q \geq_P p \cdot F_L$  and  $s \xleftarrow{p} u \xrightarrow{q} t$ . Then,  $s \xrightarrow{\geq p p} \xleftarrow{p} t$ .  
 899

900 PROOF. Splitting  $u$  at  $q$  yields  $u = v\sigma$ , where  $v = \underline{u}_q = u[Z[\bar{z}]]_q$  and  $\sigma = \bar{u}^q = \{Z \mapsto \lambda\bar{z}.u|_q\}$  is  
 901 preserving since  $u|_q$  cannot be an abstraction by definition of a pattern and is normal as a subterm  
 902 of  $u|_p$ . By assumption,  $u|_q \xrightarrow{j} t|_q$ , and by monotonicity,  $\sigma(Z) = \lambda\bar{z}.u|_q \xrightarrow{j} \lambda\bar{z}.t|_q$ . Let  $\tau$  be  $\sigma$  with the  
 903 exception  $\tau(Z) = \lambda\bar{z}.t|_q$ . Then  $\sigma \xrightarrow{j} \tau$  and by Lemma 3.12,  $u = v\sigma \xrightarrow{j} v\tau = t$ . By Lemma 5.3,  $v \xrightarrow{p} w$   
 904 for some  $w$  such that  $s = w\sigma$ . By Lemma 3.14,  $u = v\sigma \xrightarrow{j} w\tau$ . The result follows.  $\square$   
 905  
 906  
 907

908 Note that whenever  $X$  occurs self-nested in a right-hand-side,  $R[X[\bar{u}]]_p$ , such that  $\bar{u}_i^q = \lambda\bar{z}.X[\bar{t}]$   
 909 for some  $i$ , this right-hand-side can be replaced with  $R[(\lambda x.X[\bar{v}]\ \lambda\bar{z}.X[\bar{t}])]_p$  with  $v_j = u_j$  for  
 910  $i \neq j$  and  $v_i = u_i[(x\ \bar{z})]_q$ . For instance, instead of using the rule  $f(\lambda x.X[x]) \rightarrow X[g(\lambda z.X[h(z)])]$   
 911 one may use  $f(\lambda x.X[x]) \rightarrow (\lambda x.X[g(\lambda z.(x\ z))] \ \lambda z.X[h(z)])$ . In case the rule has critical pairs, this  
 912 transformation is likely to force the use of  $\beta$ -steps to joined them which won't be supported by our  
 913 criteria.  
 914

## 915 5.2 Critical peaks

916 Higher-order critical pairs have been introduced in [20], [17]. As always, a critical peak is created  
 917 by overlapping a left-hand side of rule  $G$  with the subterm of a left-hand side  $L$  at some position  $p$ .  
 918 The problem is that  $L|_p$  may have free variables that are bound above  $p$  in  $L$ . In order to restore the  
 919 dependencies of the meta-variables of  $L|_p$  upon these free variables, Nipkow introduces a lifting  
 920 operation:  
 921

922 *Definition 5.5 (Lifting).* Given a term  $L$  and a list  $\bar{x}$  of pairwise different variables, let  $L\uparrow^{\bar{x}} = L\sigma_L^{\bar{x}}$   
 923 be the *lifting* of  $L$  with respect to  $\bar{x}$ , where  $\sigma_L^{\bar{x}} = \{Y \mapsto Y'[\bar{x}] : Y \in \mathcal{M}\mathcal{V}\text{ar}(L), |Y'| = |Y| + |\bar{x}|\}$ .  
 924

925 LEMMA 5.6. We have the following properties

- 926 (1) If  $L$  is a pattern or a pre-pattern or ground, then so is  $L\uparrow^{\bar{x}}$ .  
 927 (2) If  $L\uparrow^{\bar{x}} \theta = t$  then there exists  $\sigma$  such that  $L\sigma = t$  and  $\mathcal{R}\text{an}(\sigma) \subseteq \mathcal{R}\text{an}(\theta) \cup \bar{x}$ .  
 928 (3) If  $L\sigma = t$  then there exists  $\theta$  such that  $L\uparrow^{\bar{x}} \theta = t$  and  $\mathcal{R}\text{an}(\theta) = \mathcal{R}\text{an}(\sigma) \setminus \bar{x}$ .  
 929

930 PROOF. (1) By definition. (2)  $\sigma = \sigma_x^L$ . (3) If  $\sigma = \{Y_i \mapsto v_i\}$  then take  $\theta = \{Y'_i \mapsto \lambda\bar{x}.v_i\}$ .  $\square$   
 931

We can now define untyped higher-order critical peaks:

*Definition 5.7.* Let  $i : L \rightarrow R$  and  $j : G \rightarrow D$  be two rules in  $\mathcal{R}$ ,  $o \in \mathcal{FPos}(L)$  and  $\mathcal{Var}(L|_o) = \bar{x}$  such that the equation  $L|_o = G\uparrow^{\bar{x}}$  has a most general solution  $\sigma$ . Then,  $R\sigma \leftarrow_i^{\Lambda} L\sigma \xrightarrow{o} L\sigma[D\uparrow^{\bar{x}} \sigma]_o$  is called a *critical peak of  $j$  onto  $i$*  at position  $o$ . Its associated *critical pair* is  $\langle R\sigma, L\sigma[D\uparrow^{\bar{x}} \sigma]_o \rangle$ .

Nipkow considers the unification of the (ground) patterns  $\lambda\bar{x}.L|_o = \lambda\bar{x}.G\uparrow^{\bar{x}}$  instead. Note that this problem has the same (ground) solutions as the one we consider, since the later can be obtained from the former by applying *Meta-Abs*  $|\bar{x}|$ -times.

Our definition makes sense: since  $o \in \mathcal{FPos}(L)$ , then  $o <_{\mathcal{P}} F_L$ , and therefore,  $o \in \mathcal{FPos}(L\sigma)$ . Using standard techniques, we then get the analog of Nipkow's critical pair lemma developed for the case of simply typed higher-order rewrite rules:

**LEMMA 5.8 (CRITICAL PEAK LEMMA).** Assume  $s \leftarrow_i^p u \xrightarrow{q} t$  is an overlapping peak of  $j : G \rightarrow D$  onto  $i : L \rightarrow R$  at position  $o \in \mathcal{FPos}(L)$  such that  $q = p \cdot o$ . Then there is a critical peak  $s' \leftarrow_i^{\Lambda} u' \xrightarrow{o} t'$  and a closed substitution  $\theta$  such that  $u'\theta = u|_p$ ,  $s'\theta = s|_p$  and  $t'\theta = t|_p$ .

**PROOF.** By definition of higher-order rewriting, there exists some substitutions  $\gamma$  and  $\sigma$  such that  $L\gamma = u|_p$ ,  $G\sigma = u|_q = (L\gamma)|_o$ ,  $s = u[R\gamma]_p$  and  $t = u[D\sigma]_q$ . Let  $\bar{x}$  be the set of variables bound above  $o$  in  $L$ . We cannot merge  $\gamma$  and  $\sigma$  into a substitution that can be applied to  $L$  as the  $\bar{x}$  variables may be in  $\mathcal{Ran}(\sigma)$ . However, by Lemma 5.6, there exists  $\sigma'$  such that  $G\uparrow^{\bar{x}} \sigma' = (L\gamma)|_o$ . Since  $\sigma'$  and  $\gamma$  have disjoint domains we can define  $\gamma' = \gamma\sigma'$  and have both  $L\gamma' = u|_p$  and  $G\uparrow^{\bar{x}} \gamma' = (L\gamma')|_o = u|_q$ . Besides, since  $o \in \mathcal{FPos}(L)$  and  $\mathcal{Ran}(\gamma') \cap \bar{x} = \emptyset$  we have  $(L\gamma')|_o = L|_o\gamma'$ . Therefore,  $\gamma'$  is a solution of the equation  $L|_o = G\uparrow^{\bar{x}}$ . Let  $\omega$  be its most general unifier, there exist a substitution  $\theta$  such that  $\omega\theta = \gamma'$ . The critical peak is  $s' = R\omega \leftarrow_i^{\Lambda} L\omega = u' \xrightarrow{o} L\omega[D\uparrow^{\bar{x}} \omega]_o = t|_p$ . We check that  $s'\theta = R\omega\theta = R\gamma' = R\gamma = s|_p$ ,  $u'\theta = L\omega = u|_p$  and  $t'\theta = L\gamma[D\uparrow^{\bar{x}} \gamma']_o = u|_p[D\sigma]_o = t|_p$ .  $\square$

### 5.3 Orthogonal functional reductions

The confluence proof will not be based on using  $\beta$ -rewrites, nor parallel  $\beta$ -rewrites, but, as in Tait's confluence proof of the lambda-calculus, orthogonal  $\beta$ -rewrites (called parallel reductions in [2]). Our definition is essentially Tait's, but makes the rewriting positions explicit.

To this end, we first define the product of sets of positions:

*Definition 5.9.* Given a set of parallel positions  $P$  and a family  $Q$  of sets of positions strictly below  $\Lambda$  indexed by  $P$ , we define the *orthogonal product*  $P \otimes Q$  as the set  $P \uplus \biguplus_{p \in P} p \cdot Q_p$ .

**LEMMA 5.10.** Given a set of positions  $O$  there exist a unique set of parallel positions  $P \subseteq O$  and family  $Q >_{\Lambda}$  of sets of positions such that  $O = P \otimes Q$ .

**PROOF.**  $P = \{p \in O \mid \forall q \in O, p \not> q\}$  and  $Q_p = \{q >_{\Lambda} \mid p \cdot q \in O\}$   $\square$

$P$  is called the *parallel part* of  $O$ , written  $\underline{O}$ , while  $\biguplus_{p \in P} p \cdot Q_p$  is called the *residual part* of  $O$ , written  $\bar{O}$ . Note that  $O = \underline{O} \uplus \bar{O}$ ,  $\bar{O} >_{\mathcal{P}} \underline{O}$  and that whenever  $O \neq \emptyset$ ,  $\underline{O} \neq \emptyset$  and  $\bar{O} \subset O$ .

*Definition 5.11 (Orthogonal reductions).* Orthogonal rewriting, written  $u \xrightarrow[\beta]{O} v$ , is the smallest reflexive relation on terms such that  $u \xrightarrow[\beta]{\bar{O}} s$  and  $s \xrightarrow[\beta]{O} v$  imply  $u \xrightarrow[\beta]{O} v$ , assuming  $\bar{O} \geq_{\mathcal{P}} \underline{O} \cdot F_{\beta}$ .

The alternative choice of rewriting first at positions in  $\underline{Q}$ , instead of first in  $\overline{O}$ , would yield a more complex calculation for  $O$ , explaining our formulation of Tait's parallel rewriting definition.

Note that orthogonal rewriting contains parallel rewriting. Furthermore it is easy to show that our definition of orthogonal reduction coincides with Tait's parallel rewriting. This follows from the property:  $t \xrightarrow[\beta]{P} u \xrightarrow[\beta]{Q} v$  implies  $t \xrightarrow[\beta]{P \cup Q} v$  if  $\forall p \in P, \forall q \in Q, q \not\leq_{\mathcal{P}} p$ , which is easily proved by induction on  $P$ .

We shall need several well-known properties of (Tait's) orthogonal  $\beta$ -reductions: monotonicity, stability, commutation with any other monotonic rewrite relation, and strong confluence. Besides the following properties will be needed for the coming analysis of orthogonal ancestor peaks.

LEMMA 5.12. Assume  $q \in \mathcal{P}os(u)$  and  $O = P \uplus Q \uplus R \subseteq \mathcal{P}os(u)$  with  $P \# q, Q \geq_{\mathcal{P}} q$  and  $R <_{\mathcal{P}} q$ .

Then  $u \xrightarrow[\beta]{O} v$  iff  $u \xrightarrow[\beta]{P} u \xrightarrow[\beta]{Q} u \xrightarrow[\beta]{R} v$ .

PROOF. It suffices to notice that  $\xrightarrow[\beta]{P}$  preserves  $Q$  and  $R$  and  $\xrightarrow[\beta]{Q}$  preserves  $R$ .  $\square$

LEMMA 5.13.  $(\lambda x.M N) \xrightarrow[\beta]{O} t$  with  $\Lambda \in O$  if and only if  $O = \{\Lambda\} \uplus 11 \cdot P \uplus 2 \cdot Q, M \xrightarrow[\beta]{P} M', N \xrightarrow[\beta]{Q} N'$  and  $t = M'\{x \mapsto N'\}$ .

PROOF. By definition using the fact that  $\overline{O} \geq_{\mathcal{P}} \underline{O} \cdot F_{\beta}$ .  $\square$

## 5.4 Orthogonal decreasing diagrams for free

We investigate here the linear ancestor peak properties of orthogonal  $\beta$ -reductions. Unlike the "above case", the "below case" listed first follows easily from Lemma 5.2 (LAP $\beta$ a).

LEMMA 5.14 (LAPOB). Let  $s \xleftarrow[\mathcal{R}ll]{q} u \xrightarrow[\beta]{O} t$  for some set  $O >_{\mathcal{P}} q$ . Then,  $s \xleftarrow[\mathcal{R}ll]{\geq_{\mathcal{P}} q} r \xleftarrow[\mathcal{R}ll]{q} t$ .

Besides  $u|_q = L\gamma$  and  $s = u[R\gamma]_q$  such that  $\gamma \xrightarrow[\beta]{} \sigma, t = u[L\sigma]_q$  and  $r = u[R\sigma]$ .

PROOF. Since patterns are  $\beta$ -normal,  $O \geq_{\mathcal{P}} q \cdot F_L$  and from Lemma 5.3 (extended to the set  $\underline{O}$  of parallel positions below the fringe),  $\underline{u}_O \xrightarrow[\mathcal{R}ll]{q} s'$  for some  $s'$  such that  $s = s'\overline{u}^O$ . By definition of orthogonal rewriting,  $\overline{u}^O \xrightarrow[\beta]{} \gamma$  for some  $\gamma$  such that  $t = \underline{u}_O \gamma$ .

We conclude that both  $s = s'\overline{u}^O \xrightarrow[\beta]{} s'\gamma$  and, by stability,  $t = \underline{u}_O \gamma \xrightarrow[\mathcal{R}ll]{q} s'\gamma$ .  $\square$

Definition 5.15. A non-empty set of parallel positions  $Q \subseteq \mathcal{P}os(u)$  is said to be *rigid* in  $u$  if there exists  $q \leq_{\mathcal{P}} Q$  such that  $\mathcal{V}ar(u|_Q) \subseteq \mathcal{V}ar(u|_q)$ .

If  $Q$  is rigid in  $u$ , we can always choose  $q = glb(Q)$ , the greatest lower bound of  $Q$  w.r.t.  $\leq_{\mathcal{P}}$ .

Note also that a position  $q \in \mathcal{P}os(u)$  is a singleton set of rigid positions in  $u$ .

Example 5.16. If  $u = \lambda x.f(\lambda y.y, \lambda z.x)$ , then  $\{11, 12\}$  and  $\{11, 121\}$  are rigid in  $u$  while  $\{111, 121\}$  is not since  $\mathcal{V}ar(u|_{111}) = \{y\} \not\subseteq \mathcal{V}ar(u|_{11}) = \{x\}$ .

LEMMA 5.17 (LAPOA). Let  $s \xleftarrow[\beta]{P} u \xrightarrow[\mathcal{R}ll]{q} t$ , where  $\forall p \in P, p \not\leq_{\mathcal{P}} q$ . Then  $s \xrightarrow[\mathcal{R}ll]{Q'} v \xleftarrow[\beta]{P} t$  for some set

$Q' \geq_{\mathcal{P}} (P \cup \{q\})$ . Besides  $s = u'\overline{u}^q, v = u'\sigma$  and  $t = \underline{u}_q \sigma$  such that  $\underline{u}_q \xrightarrow[\beta]{P} u', \overline{u}^q \xrightarrow[\mathcal{R}ll]{} \sigma$ .

Note that  $(P \cup \{q\})$  are positions in  $s$ , hence the condition on  $Q'$  makes sense. We could of course conclude  $P \not\geq_{\mathcal{P}} Q'$ , since it is implied by  $Q' \geq_{\mathcal{P}} (P \cup \{q\})$ , which is therefore more precise.

PROOF. We prove a more general result for which  $u \xrightarrow[\mathcal{R}_{\text{ll}}]{Q} t$ , with  $P \not\geq_{\mathcal{P}} Q$  and  $Q$  is a rigid set of positions of  $u$ . We then conclude that  $s \xrightarrow[\mathcal{R}_{\text{ll}}]{Q'} v$  for some set  $Q'$  of positions of  $s$  such that  $Q' \geq_{\mathcal{P}} (P \cup Q)$ .

We prove the result by induction on the set of positions  $P$  using the well-founded multiset extension  $>_{\text{mul}}$  of the size ordering on positions (a set is of course a multiset).

If  $P$  or  $Q$  is empty, the result is trivial. Otherwise, there are two cases depending whether  $\Lambda \in P$ .

If  $\Lambda \notin P$ , then  $u = f(\bar{u})$  and  $f(\bar{s}) \xleftarrow[\beta]{P} f(\bar{u}) \xrightarrow[\mathcal{R}_{\text{ll}}]{Q} f(\bar{t})$  with  $s_i \xleftarrow[\beta]{P_i} u_i \xrightarrow[\mathcal{R}_{\text{ll}}]{Q_i} t_i$ . Since  $Q$  is rigid in  $u$ , then  $Q_i$  is obviously rigid in  $u_i$ . Note further that in case two different subsets  $Q_i$  and  $Q_j$  are non-empty,  $\text{glb}(Q) = \Lambda$  and  $\mathcal{V}\text{ar}(u_i|_{Q_i}) \subseteq \mathcal{V}\text{ar}(u)$ , the latter property being preserved by rewriting. Since

$P_i <_{\text{mul}} P$ , by induction hypothesis,  $s_i \xrightarrow[\mathcal{R}_{\text{ll}}]{Q'_i} v_i \xleftarrow[\beta]{P_i} t_i$ , where  $Q'_i \geq_{\mathcal{P}} P_i \cup Q_i$ . The orthogonal  $\beta$ -steps can be grouped together into  $v \xleftarrow[\beta]{P} t$ . For the  $\mathcal{R}_{\text{ll}}$ -steps, let  $Q' = \bigcup_i i \cdot Q'_i$ , hence  $s = f(\bar{s}) \xrightarrow[\mathcal{R}_{\text{ll}}]{Q'} f(\bar{v}) = v$ .

From  $Q'_i \geq_{\mathcal{P}} P_i \cup Q_i$  we deduce  $Q' \geq_{\mathcal{P}} P \cup Q$ , which concludes this case.

If  $\Lambda \in P$ , then  $u = (\lambda x.M N)$ ,  $P = \{\Lambda\} \uplus P'$ ,  $P' = 1^2 \cdot P_1 \uplus 2 \cdot P_2$  and  $Q = 1^2 \cdot Q_1 \uplus 2 \cdot Q_2$ , where  $P_1, Q_1 \in \mathcal{P}\text{os}(M)$  and  $P_2, Q_2 \in \mathcal{P}\text{os}(N)$  such that  $M_w \xleftarrow[\beta]{P_1} M \xrightarrow[\mathcal{R}_{\text{ll}}]{Q_1} M_t$  and  $N_w \xleftarrow[\beta]{P_2} N \xrightarrow[\mathcal{R}_{\text{ll}}]{Q_2} N_t$ . Since

$P \cup Q = \{\Lambda\}$ , we only need to show that  $s \xrightarrow[\mathcal{R}_{\text{ll}}]{Q'} v \xleftarrow[\beta]{P} t$  for some  $Q'$  and  $v$  yet to be defined.

There are two cases, depending whether  $Q_2 = \emptyset$ , the first one being itself split into two:

(1)  $Q_2 \neq \emptyset$ , a case depicted at Figure 5. Let  $w = (\lambda x.M_w N_w) \xleftarrow[\beta]{P'} u \xrightarrow[\mathcal{R}_{\text{ll}}]{Q} t$ . Since  $P' \not\geq_{\mathcal{P}} Q$  and

$P' <_{\text{mul}} P$ , by induction hypothesis,  $w \xrightarrow[\mathcal{R}_{\text{ll}}]{Q''} (\lambda x.M_v N_v) \xleftarrow[\beta]{P'} t$ . Hence  $M_w \xrightarrow[\mathcal{R}_{\text{ll}}]{Q'_1} M_v$  and  $N_w \xrightarrow[\mathcal{R}_{\text{ll}}]{Q'_2} N_v$ .

(a)  $Q_1 \neq \emptyset$ , hence  $\text{glb}(Q) = \Lambda$ . Since  $Q$  is a rigid set of positions in  $u$ , no variable bound above  $Q_1$  in  $u$  can occur in  $M|_{Q_1}$ . It follows that  $x \notin \mathcal{V}\text{ar}(M|_{Q_1})$  and furthermore that  $\beta$ -reductions at  $P_1$  do not instantiate terms at  $Q_1$ , and therefore  $x \notin \mathcal{V}\text{ar}(M_w|_{Q'_1})$ . By repeated applications of

Lemma 3.12, it follows that  $s = M_w\{x \mapsto N_w\} \xrightarrow[\mathcal{R}_{\text{ll}}]{Q'} M_v\{x \mapsto N_v\}$ , where  $Q' := Q'_1 \uplus \{o \cdot Q'_2 : M_w|_o = x\}$  is a set of parallel positions.

(b)  $Q_1 = \emptyset$ . Then  $M_v = M_w$  and  $s \xrightarrow[\mathcal{R}_{\text{ll}}]{Q'} M_v\{x \mapsto N_v\}$ , where  $Q' = \{o \cdot Q'_2 : M_w|_o = x\}$  is a set of parallel positions.

Since  $t \xleftarrow[\beta]{P' \uplus \{\Lambda\}} v = M_v\{x \mapsto N_v\}$ , we can conclude in both cases.

(2)  $Q_2 = \emptyset$ , hence  $N_t = N$ , a case depicted in Figure 6. This time, the variable  $x$  may occur below the  $\mathcal{R}_{\text{ll}}$ -redexes in  $M$ , but there are no redexes in  $N$ . We split the orthogonal step into three parts:

$$u = (\lambda x.M N) \xrightarrow[\beta]{2 \cdot P_2} (\lambda x.M N_w) = w \xrightarrow[\beta]{\Lambda} M\{x \mapsto N_w\} = \xrightarrow[\beta]{P_1} M_s\{x \mapsto N_w\} = s$$



(4)  $s \xleftarrow[i]{p} u \xrightarrow{j} t$  with  $i : L \rightarrow R \in \mathcal{Rll}$ ,  $j : G \rightarrow D \in \mathcal{Rll}$ ,  $q \in p \cdot o$  and  $o \in \mathcal{FPos}(F_L)$ . By Lemma 5.8,

there is a critical peak  $s' \xleftarrow[i]{\Lambda} u' \xrightarrow{j} t'$  obtained by overlapping  $G$  onto  $L$  at position  $o$  and such that  $u|_p = u'\theta$ ,  $s|_p = s'\theta$  and  $t|_p = t'\theta$ . By assumption, this peak is joinable with rules of  $\mathcal{Rll} \cup \mathcal{M}\eta$ . By the stability Lemma 3.13 and Corollary 3.4, since  $\theta$  is ground, the pair  $s'\theta, t'\theta$  is joinable with rules of  $\mathcal{Rll}$ . By monotonicity lemma 3.12, so is the pair  $s, t$ . Note that there are no facing steps here, since labels decrease strictly along  $\mathcal{Rll}$ -reductions.

(5)  $s \xleftarrow[i]{O} u \xrightarrow{i} t$ , where  $i : L \rightarrow R \in \mathcal{Rll}$ . The proof of this case is shown in Figure 4.

From Lemma 5.12 we have  $s \xleftarrow{\otimes R} v \xleftarrow{\otimes Q} u' \xleftarrow{\otimes P} u$  with  $O = P \uplus Q \uplus R$  such that  $P \# q$ ,  $Q \geq_p q$  and  $R <_p q$ . Besides, since  $u(q) \in \mathcal{F}$ , then  $q \notin O$  and  $Q >_p q$ . By commutation we easily get  $u' \xrightarrow{i} t' \xleftarrow[\beta]{P} t$ . By Lemma 5.14 we have  $v \xrightarrow{i} r \xleftarrow[\beta]{\geq_p q} t'$  and by Lemma 5.17 we have  $s \xrightarrow{i} w \xleftarrow[\beta]{R} r$ . From Lemma 5.12 again, all three  $\beta$ -steps can be merged into a single orthogonal facing step  $w \xleftarrow[\beta]{\otimes} t$ . The step  $\xrightarrow{i} w$  can then be linearized, hence we get a DD.

By Lemma 5.1, all cases have been considered, we are therefore done.  $\square$

Note that the last case in the proof is actually a generalization of (LAPOa) and (LAPOb) to an arbitrary local peak between  $\beta$ - and  $\mathcal{Rll}$ -rewrites, which we could have singled out.

*Example 6.2.* SOL [11] shows the confluence of the theory of global states for the case of simple types with prenex polymorphism. We show below that it is confluent for any type discipline. To this end, we need to show first that its untyped version is terminating, and then, that the critical pairs are joinable. Unfortunately, it is not terminating, as exemplified by the following reduction with (ul): taking  $V = \text{lk}(\lambda v.v)$  and  $X = \lambda v.v$ , we get  $\text{ud}(\text{lk}(\lambda v.v), \text{lk}(\lambda v.v)) \xrightarrow{\text{ul}} \text{ud}(\text{lk}(\lambda v.v), \text{lk}(\lambda v.v))$ . We therefore cannot apply our theorem to the theory of global states as it is. However, the meta-variable  $V$  represents values, it can never be instantiated by an abstraction in any typed version of the rules. We can therefore restrict the rules so that  $V$  is a value, belonging to some given, possibly infinite set of values. The obtained set of rules can then be proved terminating by interpreting a term  $s$  with the pair  $\langle m, n \rangle$ , where  $m$  and  $n$  are the number of occurrences of  $\text{lk}$  and  $\text{ud}$  in  $s$ , respectively.

We are left with verifying the joinability of critical pairs of the obtained set of rules. Because the first argument of  $\text{ud}$ , a value, is now a constant, overlapping two  $\text{ud}$ -symbols is only possible if these constants are equal. This is what happens when overlapping the original rules, in which the first argument of  $\text{ud}$  is always a meta-variable of arity zero, hence unification can only produce equations between the meta-variables of arity zero that occur as first argument of  $\text{ud}$  in the rules. We can therefore compute the critical pairs of the original set of rules (and pretend that these meta-variables are constants).

These computations are presented inside individual boxes. In the upper middle of each box appear two left-hand sides of rules whose superposition is inside braces. The upper left-hand side is displayed in red, the lower one in blue. Next comes the most general unifier, then the colored right-hand sides, then the reduced right-hand sides, and finally the joinability verification itself, sometimes just an equality test. Colored rule names label the arrows.

The most general unifiers obtained at Example ?? are not identical for both choices of rules, although very similar since they capture the same sets of ground instances. The joinability computations are not identical for both sets either, since they depend upon the number of arguments of the meta-variables in the right-hand sides of the rules. Actually, all right-hand sides are the same except for (lu), this will impact four critical pairs only.

1177 Our choice for making the calculations is the second set.

1178

1179

1180

1181

1182

1183

1184

1185

1186

1187

1188

1189

1190

1191

1192

1193

1194

1195

1196

1197

1198

1199

1200

1201

1202

1203

1204

1205

1206

1207

1208

1209

1210

1211

1212

1213

1214

1215

1216

1217

1218

1219

1220

1221

1222

1223

1224

1225

1226

1227

1228

1229

1230

1231

1232

1233

1234

1235

1236

1237

1238

1239

1240

1241

1242

1243

1244

1245

1246

1247

1248

1249

1250

1251

1252

1253

1254

1255

1256

1257

1258

1259

1260

1261

1262

1263

1264

1265

1266

1267

1268

1269

1270

1271

1272

1273

1274

$\begin{array}{c} \text{ul} \swarrow \\ \text{ud}(V, X[V])\sigma \\ \parallel \\ \text{ud}(V, U) \end{array}$	$\begin{array}{c} \text{ud}(V, \left\{ \begin{array}{l} \text{lk}(X) \\ \text{lk}(\lambda v. U) \end{array} \right\}) \\ \sigma = \{X \mapsto \lambda v. U\} \\ = \\ \text{ud}(V, U) \end{array}$	$\begin{array}{c} \downarrow l \\ \text{ud}(V, U)\sigma \\ \parallel \\ \text{ud}(V, U) \end{array}$
$\begin{array}{c} \text{ll} \swarrow \\ \text{lk}(\lambda v. Y[v, v])\sigma \\ \parallel \\ \text{lk}(\lambda v. Z[v]) \end{array}$	$\begin{array}{c} \left\{ \begin{array}{l} \text{lk}(\lambda w. \text{lk}(Y[w])) \\ \text{lk}(\lambda w. U) \end{array} \right\} \\ \sigma = \{U \mapsto \text{lk}(Z), Y \mapsto \lambda w. Z\} \\ =_{M\eta} \\ \text{lk}(\lambda w. Z[v]) \end{array}$	$\begin{array}{c} \downarrow U \\ U\sigma \\ \parallel \\ \text{lk}(Z) \end{array}$
$\begin{array}{c} \text{ll} \swarrow \\ \text{lk}(\lambda w. Y[w, w])\sigma \\ \parallel \\ \text{lk}(\lambda w. U) \end{array}$	$\begin{array}{c} \text{lk}(\lambda w. \left\{ \begin{array}{l} \text{lk}(Y[w]) \\ \text{lk}(\lambda v. U) \end{array} \right\}) \\ \sigma = \{Y \mapsto \lambda w v. U\} \\ = \\ \text{lk}(\lambda w. U) \end{array}$	$\begin{array}{c} \downarrow U \\ \text{lk}(\lambda w. U)\sigma \\ \parallel \\ \text{lk}(\lambda w. U) \end{array}$
$\begin{array}{c} \text{ll} \swarrow \\ \text{lk}(\lambda v. Y[v, v])\sigma \\ \parallel \\ \text{lk}(\lambda v. \text{lk}(Y'[v])) \end{array}$	$\begin{array}{c} \text{lk}(\lambda w. \left\{ \begin{array}{l} \text{lk}(Y[w]) \\ \text{lk}(\lambda v. \text{lk}(Y'[v])) \end{array} \right\}) \\ \sigma = \{Y \mapsto \lambda w v. \text{lk}(Y'[v])\} \\ \xrightarrow{\text{ll}} \text{lk}(\lambda v. Y'[v, v]) \xleftarrow{l} \text{lk}(\lambda v. \text{lk}(Y'[v])) \end{array}$	$\begin{array}{c} \text{ll} \downarrow \\ \text{lk}(\lambda w. \text{lk}(\lambda v. Y'[v, v]))\sigma \\ \parallel \\ \text{lk}(\lambda w. \text{lk}(\lambda v. Y'[v, v])) \end{array}$
$\begin{array}{c} \text{ll} \swarrow \\ \text{lk}(\lambda v. Y[v, v])\sigma \\ \parallel \\ \text{lk}(\lambda v. \text{ud}(v, X[v])) \end{array}$	$\begin{array}{c} \text{lk}(\lambda w. \left\{ \begin{array}{l} \text{lk}(Y[w]) \\ \text{lk}(\lambda v. \text{ud}(v, X[v])) \end{array} \right\}) \\ \sigma = \{Y \mapsto \lambda w v. \text{ud}(v, X[v])\} \\ \xrightarrow{\text{ll}} \text{lk}(\lambda v. X[v]) =_{M\eta} \text{lk}(X) \xleftarrow{l} \text{lk}(\lambda w. \text{lk}(X)) \end{array}$	$\begin{array}{c} \text{ll} \downarrow u \\ \text{lk}(\lambda w. \text{lk}(X))\sigma \\ \parallel \\ \text{lk}(\lambda w. \text{lk}(X)) \end{array}$
$\begin{array}{c} \text{ll} \swarrow u \\ \text{lk}(X)\sigma \\ \parallel \\ \text{lk}(\lambda w. \text{lk}(X')) \end{array}$	$\begin{array}{c} \text{lk}(\lambda w. \left\{ \begin{array}{l} \text{ud}(w, X[w]) \\ \text{ud}(V, \text{lk}(X')) \end{array} \right\}) \\ \sigma = \{V \mapsto w, X \mapsto \lambda w. \text{lk}(\lambda v. X'[v])\} \\ \xrightarrow{l} \text{lk}(\lambda v. X'[v]) =_{\alpha} \text{lk}(\lambda w. X'[w]) \xleftarrow{\text{ll}} \text{lk}(\lambda w. \text{ud}(w, X'[w])) \end{array}$	$\begin{array}{c} \text{ll} \downarrow u \\ \text{lk}(\lambda w. \text{ud}(V, X'[V]))\sigma \\ \parallel \\ \text{lk}(\lambda w. \text{ud}(w, X'[w])) \end{array}$
$\begin{array}{c} \text{ll} \swarrow u \\ \text{lk}(\lambda w. X[w])\sigma \\ \parallel \\ \text{lk}(\lambda w. \text{lk}(\lambda w. \text{ud}(V, W))) \end{array}$	$\begin{array}{c} \text{lk}(\lambda w. \left\{ \begin{array}{l} \text{ud}(w, X[w]) \\ \text{ud}(U, \text{ud}(V, W)) \end{array} \right\}) \\ \sigma = \{U \mapsto w, X \mapsto \lambda w. \text{ud}(V, W)\} \\ \xrightarrow{\text{ll}} \text{lk}(\lambda w. \text{ud}(V, W)) \end{array}$	$\begin{array}{c} \text{ll} \downarrow uu \\ \text{lk}(\lambda w. \text{ud}(V, W))\sigma \\ \parallel \\ \text{lk}(\lambda w. \text{ud}(V, W)) \end{array}$

1275

1276

1277

1278

1279

1280

1281

1282

1283

1284

1285

1286

1287

1288

1289

1290

1291

1292

1293

1294

1295

1296

1297

1298

1299

1300

1301

1302

1303

1304

1305

1306

1307

1308

1309

1310

1311

1312

1313

1314

1315

1316

1317

1318

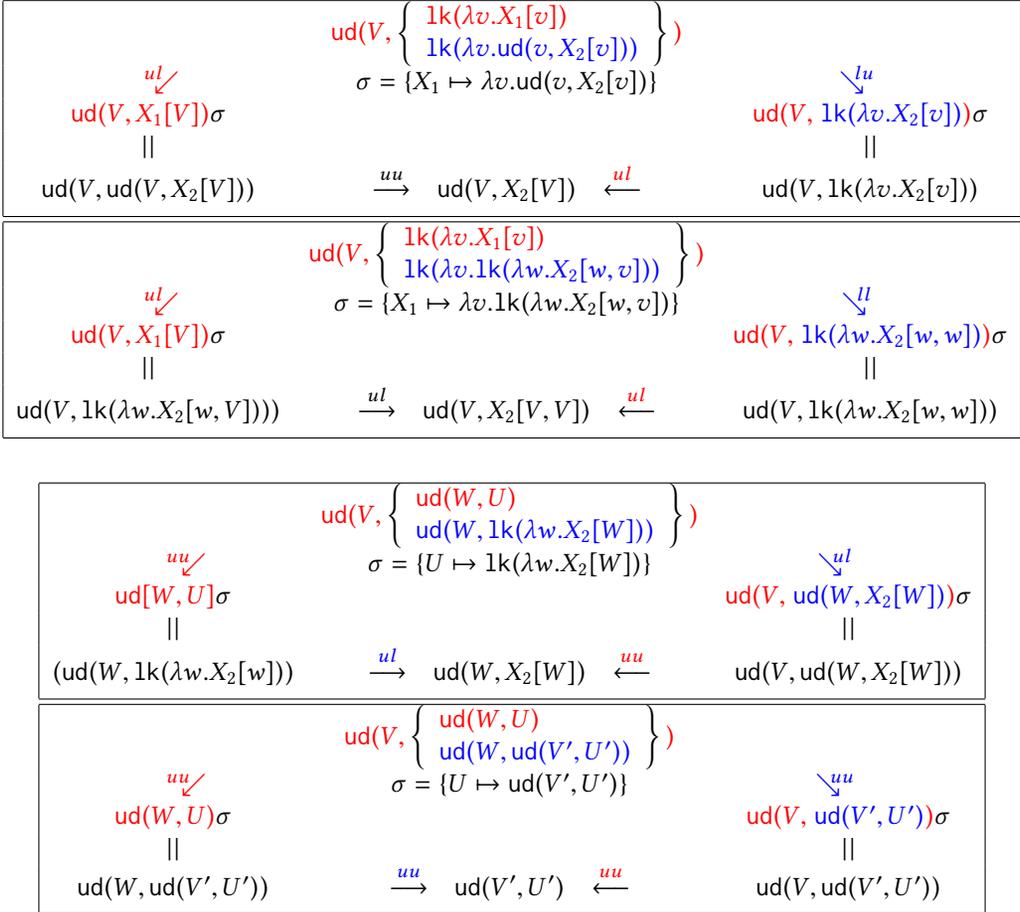
1319

1320

1321

1322

1323



Hence, all critical pairs are joinable, or joinable modulo  $M\eta$  for two of them. It follows that the theory of global states for a single location preserves confluence of the  $\beta$ -rule in the pure  $\lambda$ -calculus.

Note finally that most of these critical pairs are not development closed, since they need to be joined from both sides.

## 7 IMPACT AND RELATED WORK

One may wonder why we did not consider a well-known setting, like Klop's combinatory reduction systems (CRS, [15]) or Nipkow's higher-order rewriting (HRS), or van Oostrom and van Raamsdonk's higher-order rewriting systems (HORS, [27]), and then encode our setting within theirs. One main reason is that we always insist, in DEDUKTI, in using shallow encodings, hence do not want to encode the  $\lambda$ -calculus itself as a higher-order calculus in such a setting. Further, our notion of meta-variable has a fixed arity but may have missing arguments, which is unusual: encoding our notion of higher-order rewriting in their notion is certainly possible but not exactly trivial. Finally, encoding our setting into either HRS or HORS would break the monotonicity and stability properties of our own notion of rewriting which are critical for developing our results.

Two main questions however remain: do our results apply to these settings? and do they apply to dependent type theories. These are the questions we investigate here.

## 7.1 Relationship to Nipkow's higher-order rewriting

Nipkow's rewriting assumes terms to be simply typed, but it can be easily extended to other typing disciplines. The major requirement is indeed that  $\beta$ -reduction is strongly normalizing as well as  $\eta$ -expansion. The latter is obtained by restricting its application to functionally typed terms (which can be obtained in our case by controlling the arity of expressions).

Assuming a subset  $\mathcal{T}$  of the set of terms that satisfies these assumptions, we denote by  $u \downarrow_\beta$ ,  $u \uparrow^\eta$  and  $u \downarrow_\beta^\eta$ , the  $\beta$ -normal form, the  $\eta$ -expanded form and the  $\beta$ -normal  $\eta$ -expanded form, respectively, possibly omitting indices and exponents when convenient.

A rule "à la Nipkow" assumes  $\eta$ -expanded left-hand side patterns and  $\eta$ -expanded right-hand sides, as well as fully applied meta-variables of arity zero. To have both Nipkow's rewriting relation and ours defined in our setting, the pre-redex  $(X \bar{x})$  in a Nipkow's pattern will correspond in our syntax to the pre-redex  $X[\bar{x}]$  in which  $|X| = |\bar{x}|$ . It follows that a rule will have two different writings dubbed Klop and Nipkow, respectively. We will denote by  $\mathcal{R}_{kp}$  the set of higher-order Klop rules, corresponding to a set  $\mathcal{R}_{nw}$  of Nipkow rules, which must therefore be in  $\eta$ -expanded form.

In the Nipkow case, because meta-variables have arity zero, Klop's notion of substitution is nothing but the usual higher-order substitution. The meaning of the same expression  $L\sigma$  for some left-hand side of rule  $L \rightarrow R$  will therefore depend on whether the rule  $L \rightarrow R$  belongs to  $\mathcal{R}_{nw}$  or  $\mathcal{R}_{kp}$ : the equality  $u = L\sigma$  when  $L$  is a Klop left-hand side of rule becomes  $u =_{\beta^0} L'\sigma'$  for the corresponding Nipkow left-hand side of rule  $L'$ . The same applies to unification of left-hand sides.

We now make these remarks formal:

*Definition 7.1.*  $u \xrightarrow{p}_{L \rightarrow R} v$  for  $L \rightarrow R \in \mathcal{R}_{nw}$  iff  $u = u \downarrow$ ,  $u|_p =_{\beta^0} L\sigma$  for some  $\beta$ -normal  $\eta$ -expanded substitution  $\sigma$  and  $v = u[R\sigma]_p \downarrow$ .

We write  $u \xrightarrow{p}_{\mathcal{R}_{nw}} v$  when  $u \xrightarrow{p}_{L \rightarrow R} v$  for some  $L \rightarrow R \in \mathcal{R}_{nw}$ . As it is known that  $\eta$ -expanded forms are closed under  $\beta$ -reduction and substitution, we have:

LEMMA 7.2. If  $u \xrightarrow{p}_{\mathcal{R}_{nw}} v$  then  $u = u \downarrow$  and  $u \xrightarrow{p}_{\mathcal{R}_{kp}} \xrightarrow{\beta} v = v \downarrow$ .

LEMMA 7.3. For all  $u \in \mathcal{T}$  such that  $u \xrightarrow{p}_{\mathcal{R}_{kp}} v$  and  $v \in \mathcal{T}$ ,  $u \downarrow \xrightarrow{\mathcal{R}_{nw}} v \downarrow$ .

PROOF. We have  $u = u[L\sigma]_p = \bar{u}^p \{Z \mapsto \lambda \bar{z}. L\sigma\} = \bar{u}^p \gamma$  and  $v = u[R\sigma]_p = \bar{u}^p \{Z \mapsto \lambda \bar{z}. R\sigma\} = \bar{u}^p \theta$  with  $\gamma \xrightarrow{\mathcal{R}} \theta$ . Since  $L$  is a pattern  $(L\sigma) \downarrow = L(\sigma \downarrow)$  and since it cannot be an abstraction we have,

$$u \downarrow = ((\bar{u}^p \gamma) \downarrow) \downarrow = ((\bar{u}^p \downarrow (\gamma \downarrow)) \downarrow) \downarrow \xrightarrow{\mathcal{R}_{nw}} ((\bar{u}^p \downarrow \{Z \mapsto \lambda \bar{z}. R(\sigma \downarrow)\}) \downarrow) \downarrow = ((\bar{u}^p \{Z \mapsto \lambda \bar{z}. R\sigma\}) \downarrow) \downarrow = v \downarrow \quad \square$$

COROLLARY 7.4. If  $u \xrightarrow{\beta \eta \mathcal{R}_{kp}} v$  then  $u \downarrow \xrightarrow{\mathcal{R}_{nw}} v \downarrow$ .

LEMMA 7.5. Assume a rewrite system  $\mathcal{R}$  such that  $\xrightarrow{\beta} \cup \xrightarrow{\mathcal{R}_{kp}}$  is Church-Rosser. Then  $\xrightarrow{\mathcal{R}_{nw}}$  is Church-Rosser on  $\beta$ -normal  $\eta$ -expanded terms of  $\mathcal{T}$ .

PROOF. Assume  $u \xleftrightarrow{\mathcal{R}_{nw}} v$ . Then by Lemma 7.3,  $u = u \downarrow$ ,  $v = v \downarrow$  and  $u \xleftrightarrow{\beta \eta \mathcal{R}_{kp}} v$ . By assumption,  $u (\xrightarrow{\beta} \cup \xrightarrow{\mathcal{R}_{kp}})^* (\xleftarrow{\beta} \cup \xleftarrow{\mathcal{R}_{kp}})^* v$  and by Corollary 7.4,  $u \xrightarrow{\mathcal{R}_{nw}} \xleftarrow{\mathcal{R}_{nw}} v$ .  $\square$

THEOREM 7.6. Assume  $\mathcal{R}$  is a left-linear system such that all critical pairs are joinable using  $\xrightarrow{\mathcal{R}_{kp}}$  and  $\mathcal{R}$  is strongly normalizing. Then  $\xrightarrow{\mathcal{R}_{nw}}$  is Church-Rosser on  $\beta$ -normal  $\eta$ -expanded terms of  $\mathcal{T}$ .

1373 PROOF.  $\xrightarrow{\mathcal{R}}$  satisfies the hypotheses of Theorem 6.1. We conclude that  $\xrightarrow{\beta} \cup \xrightarrow{\mathcal{R}}$  is Church-Rosser  
 1374 and by Lemma 7.5,  $\xrightarrow{\mathcal{R}_{nw}}$  is too on  $\beta$ -normal  $\eta$ -expanded terms of  $\mathcal{T}$ .  $\square$   
 1375  
 1376

1377 This result is of course not really surprising, since it is known to hold for various type systems  
 1378 for which  $\beta$ -reductions terminate, in which case  $\mathcal{T}$  is the set of typable terms. What's new here is  
 1379 only that we abstract from a particular typing discipline.  
 1380

## 1381 7.2 Modularity of higher-order confluence

1382 In [28], van Oostrom and van Raamsdonk show that higher-order rewriting enjoys Toyama's  
 1383 modularity property when rules are left-linear: the union of two confluent higher-order linear  
 1384 rewriting systems (in their sense) that have no non-trivial critical pairs is confluent. This suggests  
 1385 an alternative way for proving that  $\mathcal{R} \cup \beta$  is confluent: first, encode  $\beta$  as a (linear) higher-order  
 1386 rewriting system made of one rule  $beta : app(lam(\lambda x.X[x]), U) \rightarrow X[U]$ . It is well-known that  
 1387 this encoding is a faithful (simply typed) encoding of the lambda-calculus [15]. Assuming now  
 1388 a higher-order rewriting system  $\mathcal{R}$  in our sense, that is, in particular, left-linear and having no  
 1389 abstraction as left-hand side, then  $\mathcal{R}$  and  $beta$  have no critical pair. Assuming now that the result  
 1390 of van Oostrom and van Raamsdonk applies, we could deduce the confluence property of the  
 1391 union from the confluence of  $beta$  (a well-know fact) and of  $\mathcal{R}$ . Now, it is easy to deduce the  
 1392 confluence property of  $\mathcal{R}$  from our assumptions: since  $\mathcal{R}$  is terminating, it suffices to prove its  
 1393 local confluence, which follows from Lemmas 5.1, 5.4, 5.8, 3.12 and 3.13, the last two being used to  
 1394 show that the joinability of critical pairs can be lifted to all overlapping peaks, as is done in the  
 1395 proof of Theorem 6.1.  
 1396

1397 It would therefore make sense to show that the modularity result of van Oostrom and van  
 1398 Raamsdonk applies to our setting. Note however that our technique is more informative: it tells us  
 1399 more precisely how local confluence diagrams are decreasing via the order used on proofs [14, 16].  
 1400

## 1401 7.3 Applicability to theories with annotated abstractions

1402 Although one could fear that the present setting becomes too specific for a wide application, we  
 1403 believe that this is not the case, and that it can be used to show confluence of rewrite rules in  
 1404 several dependent type theories without difficulty, as well as for other, related rewrite relations, as  
 1405 we have shown with Nipkow's higher-order rewriting.  
 1406

1407 In dependent type theories, lambda abstractions sometimes need to be annotated in order to  
 1408 guarantee typing properties such as decidability of type checking. Our setting supports type  
 1409 annotations through encodings, for instance using a dedicated symbol  $a \in \mathcal{X}$  and considering  
 1410  $\lambda x : A. t$  to be syntactical sugar for  $a(A, \lambda x. t)$  or  $\lambda x. a(A, t)$ . In the first case, the  $\beta$  rule itself becomes  
 1411 a rewrite rule headed with the  $a$  symbol which is not only non-terminating, preventing the use  
 1412 of our criteria, but also defies its very purpose which was based on the study of the interactions  
 1413 between the first-class  $\beta$  reduction and well-behaved rewrite rules. In the second case  $\beta$  reduction  
 1414 leaves a dangling “ $a$ ” symbol which can freeze further use of the  $\beta$  rule, for instance as in  $((\lambda x : B. \lambda y : A. x) t) u = ((\lambda x. a(B, \lambda y. a(A, x)) t) u) \xrightarrow{\beta} (a(B, \lambda y. a(A, u)) t)$  which is no longer a  $\beta$ -redex  
 1415 and requires some annotation erasure to proceed. This type erasure cannot simply be stated as an  
 1416 extra rewrite rule as it needs to be controlled :  $\lambda x : A. t$  and  $\lambda x : B. t$  should remain convertible only  
 1417 if  $A$  and  $B$  are.  
 1418

1419 Note that in some dependently typed systems, such as the Pure Type Systems [5] or the Calculus  
 1420 of Inductive Constructions [4], type erasure was proven to be safe and allows faster conversion  
 1421 check of well-typed terms. It is our understanding however that properties of the type system  
 1422

1422 are critical to ensure this irrelevance of type annotation which can only be used for computation  
 1423 optimizations once the type system has been proven to be well-behaved, which usually requires to  
 1424 prove first the confluence of (annotated) reduction. As we study confluence on untyped terms, we  
 1425 cannot rely on this kind of type erasure.

1426 Our solution would be to simply extend the definition of terms to add a type annotation to the  
 1427 syntax of  $\lambda$  abstractions. This requires to redefine the  $\beta$  rule while keeping it first-class. In particular,  
 1428 the fringe  $F_\beta = \{11, 12, 2\}$  is extended, orthogonal reduction may occur simultaneously in the body  
 1429 and the annotation of an abstraction, the unification and matching algorithm need to take the  
 1430 annotation into account All properties of our development remain true in this extended setting, in  
 1431 particular:  $\otimes_\beta \Rightarrow$  still has the diamond property, the definition of patterns keeps forbidding overlap  
 1432 between rewrite rules and  $\beta$ -redexes which is the key property for our theorem, commutation  
 1433 lemmas such as Lemma 5.2 (LAP $\beta$ a), Lemma 5.4 (LAP $\mathcal{R}$ a), Lemma 5.14 (LAPOb) and Lemma 5.17  
 1434 (LAPOa). This is the approach taken in the PhD thesis of the first author (to appear in the fall of  
 1435 2020).  
 1436

## 1437 8 CONCLUSION

1438 Confluence of first-order rewrite rules is well understood, in both the terminating and non-  
 1439 terminating cases. Confluence of left-linear higher-order rules on simply-typed  $\lambda$ -terms is well  
 1440 understood too [17]. This is true as well of confluence of first-order rules in presence of  $\beta$ -reductions  
 1441 for any type discipline for which  $\beta$  is terminating [7].  
 1442

1443 A dependently typed system, as we have seen, imposes the use of meta-variables whose arity  
 1444 is not fixed, a need which is not encountered in non-dependent type systems. Our language of  
 1445 untyped expressions must therefore cope with that peculiarity, and include meta-variables whose  
 1446 number of arguments can vary. This makes the use of existing frameworks difficult, as we have  
 1447 argued in Section 7. This paper includes therefore our own variant of an untyped  $\lambda$ -calculus.

1448 In this paper, we have described a condition on critical pairs which ensures preservation of  
 1449 confluence in the untyped  $\lambda$ -calculus by a set  $\mathcal{R}ll$  of rewrite rules whose left-hand sides are left-  
 1450 linear patterns: if  $\mathcal{R}ll$  is terminating and the critical pairs of  $\mathcal{R}ll$  are joinable by rules of  $\mathcal{R}ll$ . The  
 1451  $\beta$ -rule itself, on the other hand, cannot be used to join the critical pairs. In that case, confluence  
 1452 can be obtained by joining *nested critical pairs*, as will be shown in forthcoming paper. This other  
 1453 result does not subsume the present one, since nested critical pairs may be infinitely many, as is  
 1454 the case with the theory of global states for a unique location [? ], for which four infinite families of  
 1455 critical pairs are generated whose joinability uses a proof by (a simple) induction.

1456 In order to define critical pairs, we had to unify left-linear patterns, where patterns are specific  
 1457 untyped  $\lambda$ -terms whose definition ensures that erasing types from a simply-typed pattern in Miller's  
 1458 sense yields a pattern in our sense. Untyped patterns enjoy most general unifiers, in the same way  
 1459 as Miller's patterns do. Note that unification here looks purely syntactic, thanks to a definition  
 1460 of substitution which incorporates  $\beta$ -reductions. The linearity restriction should not be essential:  
 1461 having multiple occurrences of meta-variables requires using a merge rule in the unification case,  
 1462 and checking terms for equality in the matching case.

1463 One may wonder whether considering parallel higher-order critical pairs could improve our  
 1464 results. The difficulty here is that one of the decreasing diagrams for free, Lemma 5.4, breaks down.  
 1465 It can of course be repaired, at the price of imposing that meta-variables do not occur nested in  
 1466 one another in the right-hand sides of the rules. This restriction looks very strong. However, any  
 1467 expression such as  $X[Y]$  can be transformed into  $(X Y)$ , hence eliminating the nesting. There is of  
 1468 course a general transformation that will eliminate all nestings, making the use of parallel rewriting  
 1469 (and therefore parallel critical pairs) look attractive. The problem however, is that right-hand sides  
 1470

1471 such as  $(X \ Y)$  may result in the use of  $\beta$ -steps to join the critical pairs, hence the joinability diagrams  
 1472 would not be decreasing. This may or may not happen. It is certainly possible to exhibit examples  
 1473 for which this transformation would work. We have not encountered such a natural example so far.

1474 The case of non-left-linear rules is not touched at all here, it is indeed much more difficult since  
 1475 adding such rules to the untyped  $\lambda$ -calculus results, in general, in losing confluence, as shown by  
 1476 Klop [15]. We however show in another forthcoming paper that for all Klop's counter-examples,  
 1477 confluence is preserved on appropriate subsets of  $\lambda$ -terms, hence showing a way to get around this  
 1478 difficulty.

1479 **Acknowledgments:** to Frédéric Blanqui for useful remarks, Gilles Dowek for many discussions,  
 1480 Jiaxiang Liu for a chary reading, Vincent van Oostrom for his many suggestions and corrections to  
 1481 an earlier draft, as well as the referees for their many relevant questions about the first version of  
 1482 the manuscript.  
 1483

## 1484 REFERENCES

- 1485 [1] Ali Assaf, Guillaume Burel, Raphaël Cauderlier, Gilles Dowek, Catherine Dubois, Frédéric Gilbert, Pierre Halma-  
 1486 grand, Olivier Hermant, and Ronan Saillard. *Dedukti: a Logical Framework based on the lambda-pi-Calculus Modulo  
 1487 Theory*. draft, INRIA, 2019.
- 1488 [2] Hendrik Pieter Barendregt. *The lambda calculus : its syntax and semantics*. Studies in logic and the foundations of  
 1489 mathematics. North-Holland, Amsterdam, New-York, Oxford, 1981.
- 1490 [3] Bruno Barras, Samuel Boutin, Cristina Cornes, Judicaël Courant, Jean-Christophe Filliâtre, Eduardo Giménez, Hugo  
 1491 Herbelin, Gábor Huet, César Muñoz, Chetan Murthy, Catherine Parent, Christine Paulin-Mohring, Amokrane  
 1492 Saïrbi, and Benjamin Werner. *The coq proof assistant reference manual : Version 6.1*. 06 1997.
- 1493 [4] Bruno Barras and Benjamin Grégoire. On the role of type decorations in the calculus of inductive constructions. In  
 1494 C.-H. Luke Ong, editor, *Computer Science Logic, 19th International Workshop, CSL 2005, 14th Annual Conference of the  
 1495 EACSL, Oxford, UK, August 22-25, 2005, Proceedings*, volume 3634 of *Lecture Notes in Computer Science*, pages 151–166.  
 1496 Springer, 2005.
- 1497 [5] Gilles Barthe and Morten Heine Sørensen. Domain-free pure type systems. *J. Funct. Program.*, 10(5):417–452, 2000.
- 1498 [6] Jesper Cockx, Nicolas Tabareau, and Théo Winterhalter. How to tame your rewrite rules, 2018. Draft.
- 1499 [7] Daniel J. Dougherty. Adding algebraic rewriting to the untyped lambda calculus. *Inf. Comput.*, 101(2):251–267, 1992.
- 1500 [8] Gilles Dowek et al. The Dedukti system, 2016. Available from <http://dedukti.gforge.inria.fr/>.
- 1501 [9] Gilles Dowek, Jean-Pierre Jouannaud and Jiaxiang Liu. Confluence in untyped higher-order theories. draft hal-, INRIA,  
 1502 january 2019. Full version of a work presented at HOR 2016.
- 1503 [10] Healfdene Goguen. The metatheory of UTT. In Peter Dybjer, Bengt Nordström, and Jan M. Smith, editors, *Types for  
 1504 Proofs and Programs, International Workshop TYPES'94, Båstad, Sweden, June 6-10, 1994, Selected Papers*, volume 996 of  
 1505 *Lecture Notes in Computer Science*, pages 60–82. Springer, 1994.
- 1506 [11] Makoto Hamana. How to prove your calculus is decidable: practical applications of second-order algebraic theories  
 1507 and computation. *PACMPL*, 1(ICFP):22:1–22:28, 2017.
- 1508 [12] J. R. Hindley. An abstract form of the Church-Rosser theorem. i. *J. Symb. Log.*, 34(4):545–560, 1969.
- 1509 [13] Jean-Pierre Jouannaud and Jiaxiang Liu. From diagrammatic confluence to modularity. *Theor. Comput. Sci.*, 464:20–34,  
 1510 2012.
- 1511 [14] Jean-Pierre Jouannaud and Vincent van Oostrom. Diagrammatic confluence and completion. In Susanne Albers,  
 1512 Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikolettas, and Wolfgang Thomas, editors, *ICALP (2)*, volume  
 1513 5556 of *Lecture Notes in Computer Science*, pages 212–222. Springer, 2009.
- 1514 [15] Jan Willem Klop. *Combinatory reduction systems*. PhD thesis, CWI tracts, 1980.
- 1515 [16] Jiaxiang Liu and Jean-Pierre Jouannaud. Confluence: The unifying, expressive power of locality. In Shusaku Iida, José  
 1516 Meseguer, and Kazuhiro Ogata, editors, *Specification, Algebra, and Software - Essays Dedicated to Kokichi Futatsugi*,  
 1517 volume 8373 of *Lecture Notes in Computer Science*, pages 337–358. Springer, 2014.
- 1518 [17] Richard Mayr and Tobias Nipkow. Higher-order rewrite systems and their confluence. *Theoretical Computer Science*,  
 1519 192:3–29, 1998.
- 1520 [18] Dale Miller. A logic programming language with lambda-abstraction, function variables, and simple unification.  
 1521 *Journal of Logic and Computation*, 1(4):497–536, 1991.
- 1522 [19] Maxwell H. A. Newman. On theories with a combinatorial definition of 'equivalence'. *Ann. Math.*, 43(2):223–243, 1942.
- 1523 [20] Tobias Nipkow. Higher-order critical pairs. In *Proceedings of the 6th annual IEEE Symposium on Logic in Computer  
 1524 Science (LICS '91)*, pages 342–349, Amsterdam, The Netherlands, July 1991.

1520 [21] Ulf Norell. Dependently typed programming in agda. In Pieter W. M. Koopman, Rinus Plasmeijer, and S. Doaitse  
 1521 Swierstra, editors, *Advanced Functional Programming, 6th International School, AFP 2008, Heijen, The Netherlands, May*  
 1522 *2008, Revised Lectures*, volume 5832 of *Lecture Notes in Computer Science*, pages 230–266. Springer, 2008.

1523 [22] Mitsuhiro Okada. Strong normalizability for the combined system of the typed lambda calculus and an arbitrary  
 1524 convergent term rewrite system. In Gaston H. Gonnet, editor, *Proceedings of the ACM-SIGSAM 1989 International*  
 1525 *Symposium on Symbolic and Algebraic Computation, ISSAC '89, Portland, Oregon, USA, July 17-19, 1989*, pages 357–363.  
 ACM, 1989.

1526 [23] Gordon D. Plotkin and John Power. Algebraic operations and generic effects. *Applied Categorical Structures*, 11(1):69–94,  
 1527 2003.

1528 [24] Vincent van Oostrom. Confluence by decreasing diagrams. *Theor. Comput. Sci.*, 126(2):259–280, 1994.

1529 [25] Vincent van Oostrom. Developing developments. *Theor. Comput. Sci.*, 175(1):159–181, 1997.

1530 [26] Vincent van Oostrom. Confluence by decreasing diagrams converted. In Voronkov A., editor, *RTA*, volume 5117 of  
 1531 *Lecture Notes in Computer Science*, pages 306–320. Springer, 2008.

1532 [27] Vincent van Oostrom and Femke van Raamsdonk. Comparing combinatory reduction systems and higher-order  
 1533 rewrite systems. In Jan Heering, Karl Meinke, Bernhard Möller, and Tobias Nipkow, editors, *Higher-Order Algebra,*  
*Logic, and Term Rewriting, First International Workshop, HOA '93, Amsterdam, The Netherlands, September 23-24, 1993,*  
 1534 *Selected Papers*, volume 816 of *Lecture Notes in Computer Science*, pages 276–304. Springer, 1993.

1535 [28] Vincent van Oostrom and Femke van Raamsdonk. Weak orthogonality implies confluence: The higher order case. In  
 1536 Anil Nerode and Yuri V. Matiyasevich, editors, *Logical Foundations of Computer Science, Third International Symposium,*  
 1537 *LFCS'94, St. Petersburg, Russia, July 11-14, 1994, Proceedings*, volume 813 of *Lecture Notes in Computer Science*, pages  
 379–392. Springer, 1994.

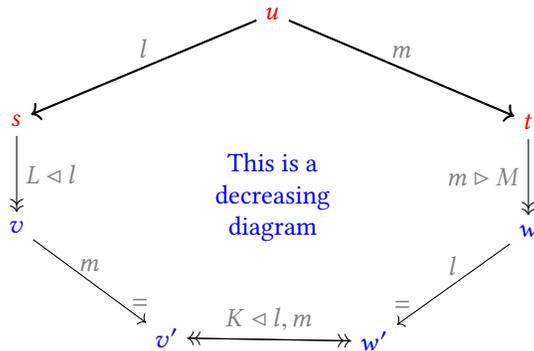


Fig. 2. Decreasing diagram

1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617

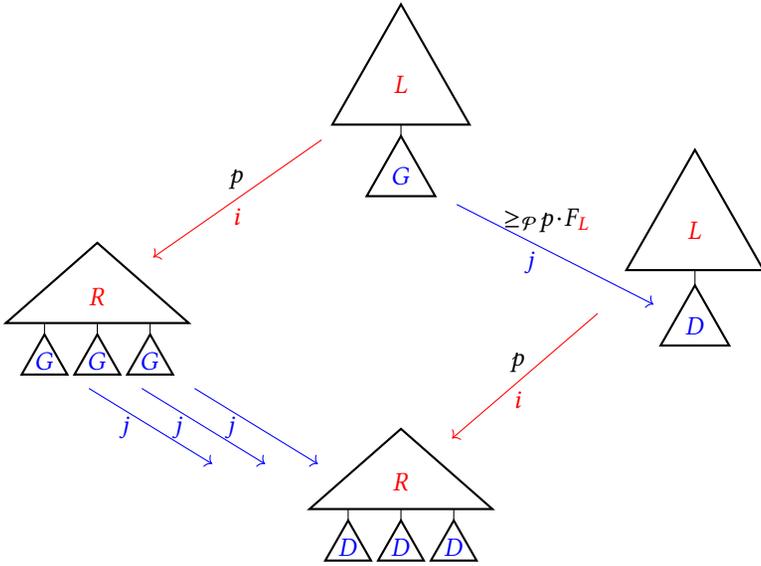


Fig. 3. Ancestor peaks in rewrite theories.  $L, G$  stand for terms rewriting to  $R, D$ , using a red rule in  $\mathcal{R}ll$  and a blue rule in  $\mathcal{R}ll \cup \beta$ .

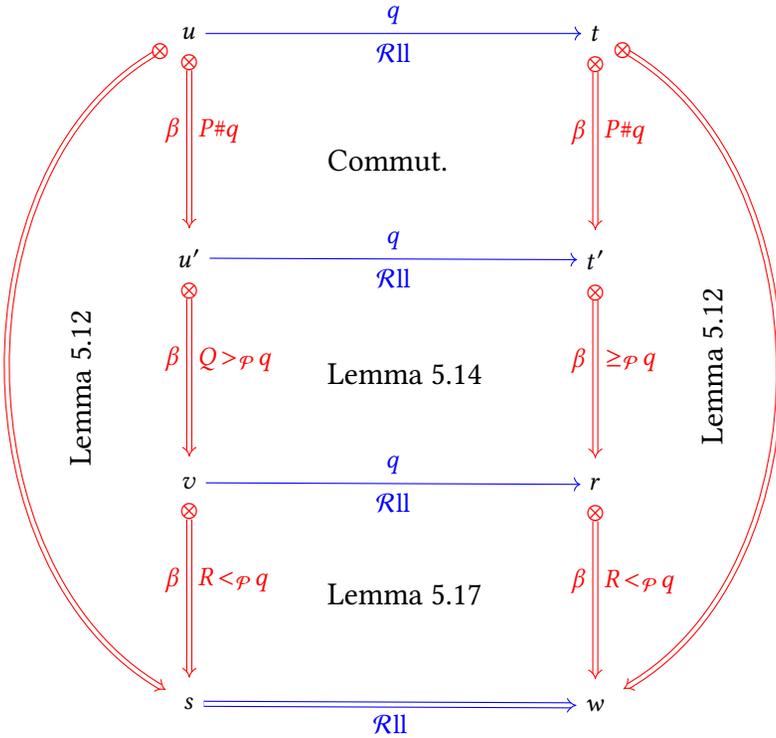


Fig. 4. Construction of a decreasing diagram for heterogeneous local peaks.

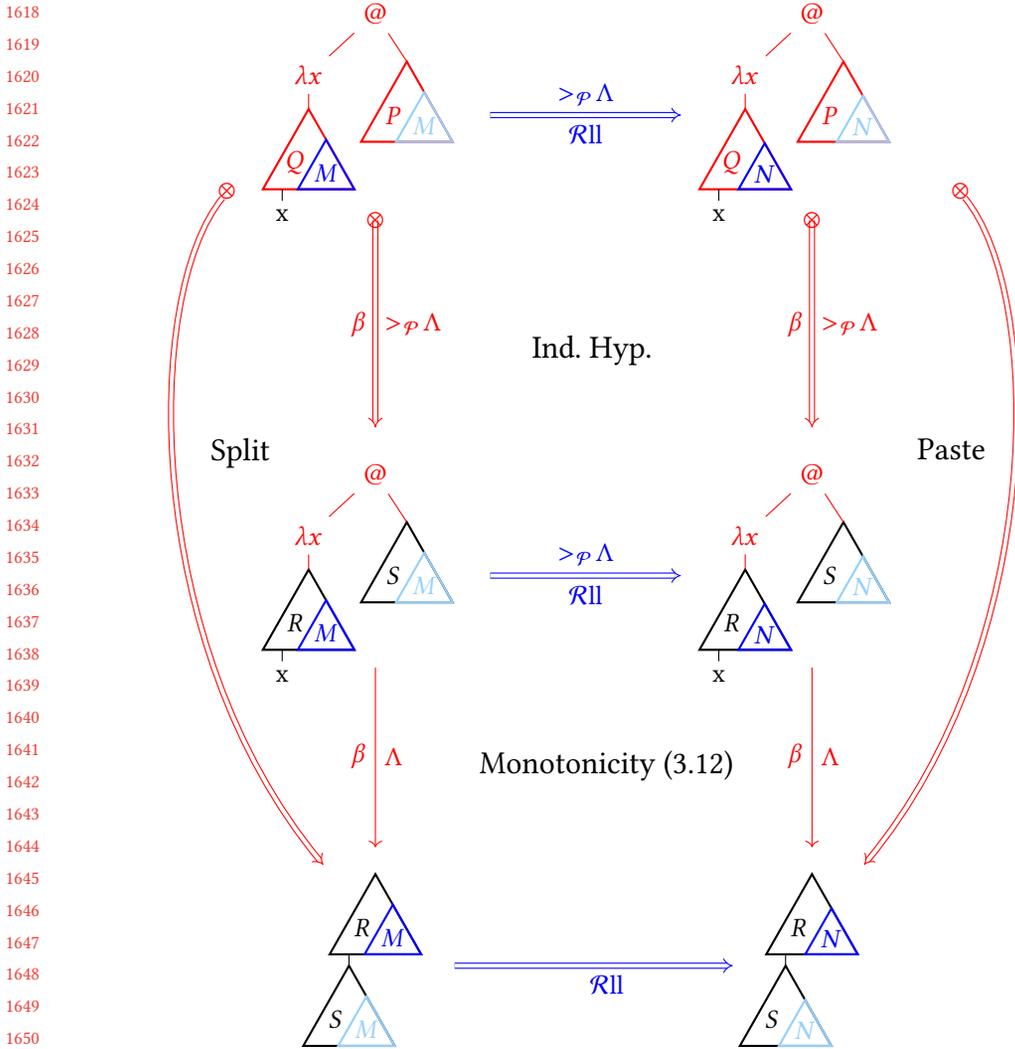


Fig. 5. Construction of a decreasing diagram for peak: Lemma 5.17

1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666

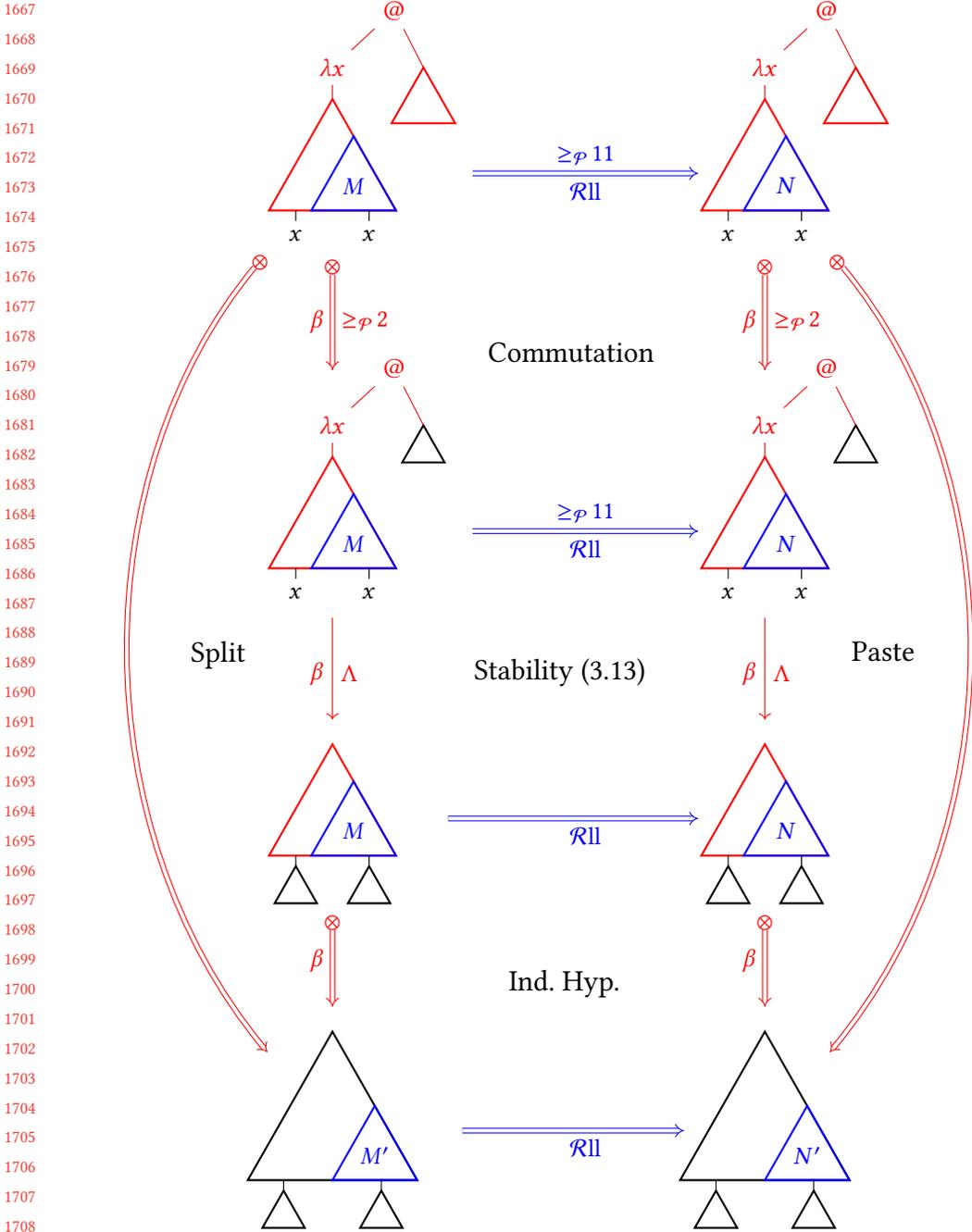


Fig. 6. Construction of a decreasing diagram for peak: Lemma 5.17