Confluence in (Un)Typed Higher-Order Theories by means of Critical Pairs

Gaspard Ferey, Jean-Pierre Jouannaud

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

GASPARD FÉREY AND JEAN-PIERRE JOUANNAUD

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. In a series of papers, we develop techniques based on van Oostrom's decreasing diagrams 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.

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

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), 29 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 [7]; 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 and Agda, 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 [3], as is alreay the case in Dedukti [5], and several on-going developments in Agda are already using this facility. Investigating the preservation of confluence by user-defined rewrite rules in λ-calculus appears therefore to be very timely.

Dedukti 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. Let $\mathcal{R}$ be the set of user-defined rewrite rules, which come in addition to the β-rule. The rewrite relation underlying the type theory is then generated by both $\mathcal{R}$ and the β-rule. Studying the meta-theory of such a type theory implies investigating the confluence property of $\beta \cup \mathcal{R}$.

There are three main tools for analyzing confluence of a rewrite relation: Newman's Lemma [14], Hindley-Rosen's Lemma [9], and van Ostrom's Theorem which generalizes both previous ones [19]. Since beta rewrites are non-terminating in pure lambda calculus, Newman's Lemma does not apply. And if the rules have non-trivial critical pairs, then Hindley-Rosen's Lemma does not apply either. Even the subtle use of Hindley-Rosen's Lemma allowing for development-closed critical

2017. 2475-1421/2017/1-ART1 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

pairs [18] is too restrictive for practical usage. The way out is the use of van Oostrom's decreasing diagrams [17]. The fact that beta reductions do not terminate for pure lambda terms is no obstacle since we do not rely on termination for showing confluence when using decreasing diagrams. A further reason for considering pure lambda terms is that it is then easy to deduce confluence for any type system, including dependent type systems, for which the rules enjoy type preservation.

Van Oostrom's theorem is abstract, its application to term rewriting relations conceals many difficulties. Further, neither confluence nor termination are preserved by adding a confluent and terminating set of rewrite rules to a λ-calculus. A counter-example to termination in the simply typed λ-calculus is given in [15]. Numerous counter-examples to confluence in the pure λ-calculus are given in [11]. The problem we address is by no means simple.

Our untyped λ-calculus is intended to fit with the implementation of Dedukti. The format of rules is classical: left-hand sides must be patterns [12, 13], which are extremely useful for describing encodings of a type theory in another, a keen application to us. Considering untyped terms requires simple adaptations of the usual higher-order rewriting definitions. In particular, we shall consider that the meta-variables used in rules have an arity which is not fixed, but bound, hence allowing for implicit arguments. These adaptations impact unification: we shall precisely analyze unification of linear untyped patterns and show the existence of most general unifiers computable in linear time.

Our contribution is that a set $\mathcal{R}$ll of rules which is terminating on the set of pure λ-terms and whose left-hand sides are linear patterns, preserves confluence of the λ-calculus if its critical pairs are joinable by using rules in $\mathcal{R}$ll.

This result is then demonstrated with the example of a theory of global states due to Plotkin and Power [16], whose rules have overlapping linear higher-order patterns as left-hand sides. As we shall see, its critical pairs are not development closed. The confluence of this example had indeed been shown already in [8]. Hamana shows first that the (simply) typed rules are terminating, then that the higher-order critical pairs are joinable, using Newman's Lemma to deduce its confluence. Our methods apply independently of the typing system, hence we can deduce that the same example remains confluent when using a dependently typed discipline.

We recall the notion of labeled reduction and decreasing diagram in Section 2, and describe our higher-order setting in Section 3. Matching and unification of untyped patterns and basic properties of untyped higher-order rewriting is considered in Section 4. Local rewriting peaks are analyzed in Section 5. Our confluence result is stated and proved in Section 6. Significance of the framework, and of our result, is discussed in Conclusion.

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 $\longleftrightarrow$, $\longrightarrow\!\!\!\!\!\rightarrow$, and $\twoheadleftarrow\!\!\!\!\!\rightarrow$, 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 $s \longrightarrow\!\!\!\!\!\rightarrow t$. Termination is the impossibility of an infinite rewriting sequence $t_0 \longrightarrow t_1 \longrightarrow \ldots \longrightarrow t_n \longrightarrow \ldots$. Termination 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 $s \longrightarrow\!\!\!\!\!\rightarrow v \twoheadleftarrow\!\!\!\!\!\leftarrow t$ 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 [17]). Given a labeled relation $\longrightarrow$ on an abstract set, we denote by $DS(m,n)$ the set of *decreasing rewrite sequences* of the form $u \overset{\delta}{\longrightarrow\!\!\!\!\!\rightarrow} v$ or $u \overset{\gamma}{\longrightarrow\!\!\!\!\!\rightarrow} s \overset{n}{\longrightarrow} t \overset{\delta}{\longrightarrow\!\!\!\!\!\rightarrow} v$ such that the labels in γ and δ are strictly smaller than, respectively, m, and, m or n. The steps labeled by γ, n and δ, are called the *side* steps, *facing* step and *middle* steps of the decreasing sequence, respectively.

Given a local peak $v \overset{m}{\longleftarrow} u \overset{n}{\longrightarrow} w$, 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 at 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 is [19], we won't need it here.

Theorem 2.2 ([17]). *A labeled relation is Church-Rosser if all its local peaks have DDs.*

van Oostrom's theorem generalizes to rewriting modulo an equational theory, for example $=_\alpha$ in the λ-calculus, in which case $\triangleright$ must be compatible with the equational theory [10]. This is of course true of $=_\alpha$ which must be built-in any definition of reduction over lambda terms. Further, equational steps with $=_\alpha$ must have a minimal label, which is easy to achieve.

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 β-rule of the underlying λ-calculus by a set R of user-defined rewrite rules built over that vocabulary. Were we to analyze the confluence of R alone, then, the situation would be similar to the first-order case, at least when left-hand sides of rules are patterns [12]. Unfortunately, confluence of $R \cup \beta$ 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 [11], that fits with Dedukti [5]. Terms are those of the pure lambda calculus enriched with $\mathcal{F}$*-headed* terms of the form $f(\overline{u})$ with $f \in \mathcal{F}$ and *meta-terms* of the form $Z[\overline{v}]$ with $Z \in \mathcal{Z}$. Only variables can be abstracted over. Elements of the vocabulary have arities, denoted by vertical bars as in $|a|$. Variables have arity zero. The grammar of terms is the following:

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

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 $f, g, h, \ldots$ for function symbols and $x, y, z, \ldots$ for variables, and reserve capital letters $X, Y, Z, \ldots$ 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 $\overline{u}$, $\overline{u}[m..n]$ denotes the finite sublist $u_m, \ldots u_n$. The list $\overline{u}$ is omitted in case it is the list of natural numbers.

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 that allows meta-variables to take *implicit arguments*. This peculiarity 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. 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.

Positions in terms are words over the natural numbers (assuming $|\lambda x.| = 1$), using $\cdot$ for concatenation, $\wedge$ for the empty word, P/p for $\{q : p \cdot q \in P\}$, $\leq_{\mathcal{P}}$ for the prefix order (*above*), $\geq_{\mathcal{P}}$ for its inverse (*below*), $>_{\mathcal{P}}$ for the strict part of $\geq_{\mathcal{P}}$, and $p\#q$ for $\neg(>_{\mathcal{P}} \vee \leq_{\mathcal{P}})$ (*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 $\mathcal{P}os(M)$, $\mathcal{VP}os(M)$, $\mathcal{MP}os(M)$ for the following respective sets of positions of M: all positions, the positions of free variables, and of meta-variables, and $\mathcal{V}ar(M)$ and $\mathcal{MV}ar(M)$ for its sets of free variables and of meta-variables respectively. A term M is *ground* if $\mathcal{V}ar(M) = \emptyset$, *closed* if $\mathcal{MV}ar(M) = \emptyset$, and linear if $|\mathcal{MP}os(M)| = |\mathcal{MV}ar(M)|$. Given now a term M and a position $p \in \mathcal{P}os(M)$, we use $M(p)$ for its symbol at positions 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 notations extends to sets P of parallel positions in $M[\overline{N}]_P$ or $M[N]_P$ in case all terms in $\overline{N}$ 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 $Z[\overline{v}]$. 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.

A *substitution* is a *capture-avoiding* homomorphism written as $\sigma = \{x_1 \mapsto M_1, \ldots, x_n \mapsto M_n\}$, or $\sigma = \{\overline{x} \mapsto \overline{M}\}$, where $M_i = \lambda\overline{y_i}.N_i$ with $|\overline{y_i}| \geq |x_i|$. Note that x_i denotes here an element of $\mathcal{X} \cup \mathcal{Z}$, hence its arity may be non-null. $\mathcal{D}om(\sigma) = \{x_1, \ldots, x_n\} \subseteq \mathcal{X} \cup \mathcal{Z}$ is the *domain* of σ while $\mathcal{R}an(\sigma) = \bigcup_{i=1}^{i=n} \mathcal{V}ar(M_i)$ is its *image*. The substitution σ is *ground* (resp., *closed*) when so are all M_i's. A substitution σ can be *restricted to* or *deprived from* (meta-)variables in some set V, written $\sigma_{|V}$ and $\sigma_{\setminus V}$ respectively. As in λ-calculus, substituting in terms requires renaming bound variables to avoid capturing free ones. Using post-fixed notation, $x_i\sigma = t_i$ and $y\sigma = y$ if $y \notin \mathcal{D}om(\sigma)$; $f(\overline{t})\sigma = f(\overline{t}\sigma)$; $(u\ v)\sigma = (u\sigma\ v\sigma)$; and $(\lambda x.u)\sigma = \lambda x.u\sigma$ if $x \notin \mathcal{D}om(\sigma) \cup \mathcal{R}an(\sigma)$ (otherwise, as announced, x must be renamed away from $\mathcal{D}om(\sigma) \cup \mathcal{R}an(\sigma)$ in $\lambda x.u$.) The additional rule for meta-variables is as follows: if $Z \mapsto \lambda\overline{x}.s \in \sigma$, then $Z[\overline{u}]\sigma = \lambda\overline{x}[m{+}1..n].s\{\overline{x}[1..m] \mapsto \overline{u}\sigma\}$, where $|\overline{u}| = m \leq n = |Z|$, hence delaying the replacement of those arguments of Z that are missing. The result $t\sigma$ of substituting the term t is called an *instance* (of t) and the operation itself an *instantiation*.

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

Substitution of meta-variables was introduced by Klop in the case of a fixed arity [11]. Our definition ensures the straightforward properties that arities are non-decreasing under substitution,

hence provide enough abstractions for all meta-terms of the form $X[\overline{u}]$ encountered in a computation. (So does of course Klop's definition for the case of fixed arities.)

Example 3.1. Let X be a ternary meta-variable, $s = \lambda x.(X[x, f(x,y)]\ g(y))$ and $\sigma = \{X \mapsto \lambda xzz'.h(z',z), y \mapsto h(a,a)\}$. Then, $s\sigma = \lambda x.(\lambda z'.h(z', f(x,h(a,a)))\ g(h(a,a)))$. Then, $ar(s) = 1+(3-2) = 2 = ar(s\sigma)$.

Rewriting terms extends to substitutions as expected, while substitutions are extended to sequences of terms and to substitutions in the natural way, keeping the same post-fixed notation.

LEMMA 3.2. *Given u, σ, τ, $(u\sigma)\tau = u(\sigma\tau)$ (we write $u\sigma\tau$).*

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 $\underline{u}_P = u[Z_1[\overline{x_1}]]_{p_1} \ldots [Z_n[\overline{x_n}]]_{p_n}$ (u is cut below P) and its associated substitution by $\overline{u}^P = \{Z_i \mapsto \lambda\overline{x_i}.u|_{p_i}\}_{i=1}^{i=n}$ (u is cut above P), where, for all $i \in [1,n]$, $\overline{x_i}$ 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) $|\overline{x_i}|$. The definition of substitution for meta-variables ensures that $\underline{u}_P\overline{u}^P = u$, which justifies writing $u = u[u|_P]_P$ 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.

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

Two different kinds of reductions coexist in $\lambda\mathcal{F}$, 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 α-index stresses that renaming bound variables, called *α-conversion*, is built-in.

As is customary [13], the particular case for which v is a variable is denoted by β^0. Note that instantiating a β^0-step may yield a full β-step. For example, $s = (\lambda x.(\lambda y.g(y)\ x)\ a) \xrightarrow[\beta^0]{1\cdot 1} (\lambda x.g(x)\ a) \xrightarrow[\beta]{\Lambda} g(a)$ while $s \xrightarrow[\beta]{\Lambda} (\lambda y.g(y)\ a) \xrightarrow[\beta]{\Lambda} g(a)$. 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: $\lambda z.X[\overline{u}, z] =_{M\eta} X[\overline{u}]$ if $|X| > |\overline{u}|$, z fresh. When oriented from left to right, $M\eta$ is terminating and confluent. It has an even more important property: assume σ is a substitution replacing X by $\lambda\overline{x}z.v$. Then $\lambda z.X[\overline{u}, z]\sigma = \lambda z.v\{\overline{x} \mapsto \overline{u}\} = X[\overline{u}]\sigma$. So, $M\eta$-steps disappear when taking instantiations, a key property for us.

3.3 Higher-order reductions

Higher-order reductions result from rules whose left-hand sides are higher-order patterns in Miller's or Nipkow's sense [12], although they need not be typed:

Definition 3.3 (Pattern). A *pre-redex* of arity n in a term L is an unapplied meta-term $Z[\overline{x}]$ whose arguments $\overline{x}$ are n pairwise distinct variables. A *pre-pattern* is a ground β-normal term all of whose

meta-variables occur in pre-redexes. A *pattern* is a pre-pattern which is neither an abstraction nor a pre-redex.

It is important to assume, as does Nipkow, that pre-patterns are β-normal. Note that 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 isn't important until later, when we address the question of matching and unification of patterns.

The properties investigated below are true of pre-patterns, not only of patterns.

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

Definition 3.4 (Fringe). The *fringe* F_L of a pre-pattern L is the set of parallel positions of its pre-redexes. We denote by $\mathcal{FP}os(L) = \{p \in \mathcal{P}os(L) : p <_{\mathcal{P}} F_L\}$ the (non-empty) set of *functional positions* of the pre-pattern L, and by $\mathcal{MV}ar(L, o)$, for $o \in F_L$, the meta-variable Z such that $L|_o = Z[\overline{x}]$. We also define $F_\beta = \{1, 2\}$ for convenience.

Example 3.5. 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, we have:

Lemma 3.6. *Given a pre-pattern L, let $p \in F_L$ and $L|_p = Z[\overline{x}]$ be a pre-redex. Then, all variables in $\overline{x}$ are bound above p in L.*

We can now define higher-order rules and rewriting:

Definition 3.7 (Rule). A (higher-order) *rule* is a triple $i : L \to R$, whose (possibly omitted) *index* i is a natural number, left-hand side L is a pattern, and right-hand side R is a ground β-normal term such that $\mathcal{MV}ar(R) \subseteq \mathcal{MV}ar(L)$. 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 and splittings belong to the meta-language, which serves analyzing the properties of the language. The role taken by free variables in first-order rules is therefore taken here by meta-variables of arity zero.

Definition 3.8 (Higher-order untyped rewriting). Given an open term u, a position $p \in \mathcal{P}os(u)$, and a rule $i : L \to R$, then u rewrites with i at p, written $u \xrightarrow[i]{p} v$, iff $u|_p = L\gamma$ for some substitution γ, and $v = u[X[\overline{x}]]_p\{X \mapsto \lambda\overline{x}.R\gamma\} = u[R\gamma]_p$, where $\overline{x}$ is the list of variables of $u|_p$ which are bound above the position p in u. We write $u \xrightarrow[\mathcal{R}]{p} v$ for $\exists i \in \mathcal{R}.u \xrightarrow[i]{p} v$.

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

- $\underline{u}_p = u[X[\overline{x}]]_p$ and $\overline{u}^p = \{X \mapsto \lambda\overline{x}.u|_p\}$ with $\overline{x}$ variables bound above p in u
- $u = \underline{u}_p\overline{u}^p = \underline{u}_p\{X \mapsto \lambda\overline{x}.u|_p\} = \underline{u}_p\{X \mapsto \lambda\overline{x}.L\gamma\}$
- $v = \underline{u}_p\{X \mapsto \lambda\overline{x}.R\gamma\}$, hence $\underline{v}_p = \underline{u}_p$, $\overline{v}^p = \{X \mapsto \lambda\overline{x}.R\gamma\}$ and $v|_p = R\gamma$.

Example 3.9. Let $L = der(\lambda x.sin(F[x])) \to R = \lambda x.cos(F[x])$, and take for σ the identity substitution $\{F \mapsto \lambda x.x\}$. Then, $L\sigma = der(sin(x))$ and $R\sigma = cos(x)$, hence $der(sin(x)) \longrightarrow cos(x)$.

In sharp contrast with Nipkow [12], we observe that we do not need matching modulo β^0, since the corresponding β^0-steps are now hidden in the Klop's definition of substitution for meta-variables. We will however show that our main confluence result applies to Nipkow's definition: the use of Klop's definition of substitution for meta variable can be seen as a technical choice.

Besides, we do *not* assume that u, or v, is β-normal, or even β-normal up to position p. We cannot for two reasons: β-normal forms may not exists, and we need monotonicity and stability properties:

Lemma 3.10 (Splitting Above). *Let $s \xrightarrow[L\to R]{q} t$. Then, $\overline{s}^q \xrightarrow[L\to R]{} \sigma$ and $t = \underline{s}_q\sigma$.*

Lemma 3.11 (Monotonicity). *Let $s \xrightarrow[L\to R]{p} t$ and u a term such that $q \in \mathcal{P}os(u)$. Then, $u[s]_q \xrightarrow[L\to R]{q\cdot p} u[t]_q$.*

Monotonicity follows directly from definition and Lemma 3.10. Stability is just as easy.

Lemma 3.12 (Stability). *Let $s \xrightarrow[L\to R]{p} t$ and σ a substitution. Then $s\sigma \xrightarrow[L\to R]{p} t\sigma$.*

Proof. By definition of higher-order rewriting, $s|_p = L\gamma$ for some substitution γ, and $t = s[R\gamma]_p$. We have $s\sigma|_p = s|_p\sigma = L\gamma\sigma$ and $t\sigma = s[R\gamma]_p\sigma = s\sigma[R\gamma\sigma]_p$ yielding the result. □

Lemma 3.13 (Substitution Lemma). *Let $u \xrightarrow[\mathcal{R}ll]{p} v$ and $\sigma \xrightarrow[\mathcal{R}ll]{} \tau$. Then, $u\sigma \xrightarrow[\mathcal{R}ll]{} v\tau$.*

Proof. We first prove $u\sigma \xrightarrow[\mathcal{R}ll]{} u\tau$ by induction on u:

- $u = f(\overline{u})$ with $f \in \mathcal{V} \cup \{@, \lambda\}$ By induction hypothesis, $\overline{u}\sigma \longrightarrow\!\!\!\!\!\rightarrow \overline{u}\tau$. By monotonicity, $f(\overline{u}\sigma) \longrightarrow\!\!\!\!\!\rightarrow f(\overline{u}\tau)$. Conclusion follows.
- $u = x$. This case is straightforward.
- $u = X[\overline{u}]$ with $X \notin \mathcal{D}om(\sigma)$. Similar to the first case.
- $u = X[\overline{u}]$ with $X\sigma = \lambda\overline{x}.w$, hence $w \xrightarrow[\mathcal{R}ll]{} w'$ and $X\tau = \lambda\overline{x}.w'$. By induction hypothesis, $\overline{u}\sigma = \overline{u}\tau$. Hence $u\sigma = w\{\overline{x} \mapsto \overline{u}\sigma\} \longrightarrow\!\!\!\!\!\rightarrow w\{\overline{x} \mapsto \overline{u}\tau\}$ (by monotonicity) $\longrightarrow\!\!\!\!\!\rightarrow w'\{\overline{x} \mapsto \overline{u}\tau\}$ (by stability) $= u\tau$.

Since $u\tau \xrightarrow[\mathcal{R}ll]{} v\tau$ by stability, we conclude that $u\sigma \longrightarrow\!\!\!\!\!\rightarrow v\tau$. □

3.4 Rewrite theories

Definition 3.14. A *$\lambda\mathcal{F}$-rewrite theory* is a pair $(\mathcal{F}, \mathcal{R})$ made of a user's signature $\mathcal{F}$ and a set $\mathcal{R}$ of higher-order rewrite rules on that signature, defining the rewrite relation $\xrightarrow[\lambda\mathcal{F}]{}$ of $\lambda\mathcal{F}$ as $\xrightarrow[\mathcal{R}\cup\beta_\alpha]{}$.

Rewrite theories are used in Dedukti [1] to define the conversion rule of the calculus, which is, as is customary, untyped. The rewrite relation implemented in Dedukti is indeed the one we just described, Klop's notion of substitution for meta-variables being implemented via a priority mechanism.

The main question addressed in this paper is whether a $\lambda\mathcal{F}$-rewrite theory is confluent (Church-Rosser), and how to show its confluence by calculating and inspecting critical pairs of some form. We shall focus on rewrite theories for which the set of rules $\mathcal{R}$ satisfies linearity assumptions. We say that $\lambda\mathcal{F}$ is : a *left-linear* theory if $\mathcal{R}$ is a set of left-linear rules; a *right-linear* theory if $\mathcal{R}$ is a set of right-linear rules; a *semi-linear* theory if $\mathcal{R}$ is made of rules which are of either kind.

In this paper, we restrict our attention to left-linear rewrite theories $(\mathcal{F}, \mathcal{R}ll)$.

3.5 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 [16]. It is described by two types, Val for values and A for states, a unary operation lk for looking up a state, a binary operation 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:

$$lk \quad : \quad (Val \to A) \to A \quad \Big| \quad ud \quad : \quad Val, A \to A$$

$lk(\lambda v.t)$ looks up the state, binds its value to v, and continues with t while $ud(v,t)$ updates the state to v, and continues with t. Types are given for a better understanding, they do not play any role here. Let us now give the rules, using X (resp. Y) (resp. U, V, W) for meta variables of arity 1 (resp. 2) (resp. 0). We also use Z, whose arity will have to be given. 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.

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

This typed higher-order theory was studied by Hamana, who was interested in its confluence investigated with his Haskell-based analysis tool SOL [8]. Our presentation is a simplification of Hamana's, whose 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 η-expansions:

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

We could of course, eliminate the η-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

Firing rules requires pattern matching an arbitrary term against a pattern, while computing critical pairs, which play a key role in the analysis of overlapping peaks, requires unifying two patterns. Both algorithms are described by rewrite rules operating on equational problems.

Definition 4.1. A (matching or unification) *equational problems* $\mathcal{P}$ is a conjunction of elementary equations. An *elementary equation* is either the constant $\bot$ or is of the form $u = v$ in which u is a pre-pattern, v is either a pre-pattern (unification case), or an arbitrary term (matching case).

We now define solutions and unifiers of an equational problem, the unifiers being representations of their solutions. It is important to note here that patterns have no free variables. This implies that solutions and unifiers of unification problems can be restricted to be ground, since additional variables are not needed for expressing unifiers, we can use meta-variables of arity zero instead.

Definition 4.2. A *solution* of a matching problem $\mathcal{P}$ different from $\bot$ is a substitution γ such that $\mathcal{D}om(\gamma) \subseteq \mathcal{Z}$ and for all equations $u = v \in \mathcal{P}$, $u\gamma =_\alpha v$.

A *solution* of a unification problem $\mathcal{P}$ different from $\bot$ is a closed, ground substitution γ such that $\mathcal{D}om(\gamma) \subseteq \mathcal{Z}$, and for all equations $u = v \in \mathcal{P}$, then $u\gamma =_\alpha v\gamma$.

A *unifier* of a unification problem $\mathcal{P}$ different from $\bot$ is a ground substitution γ such that $\mathcal{D}om(\gamma) \subseteq \mathcal{Z}$, and for all equations $u = v \in \mathcal{P}$, then $u'\gamma =_{\alpha,M\eta} v'\gamma$.

The constant $\bot$ has no (matching or unification) solution nor unifier.

Unifiers equate terms of an equation modulo renaming, but also modulo extensionality for meta-variables. As we have seen, the latter steps will disappear by instantiation of the meta-variables: unifiers are not solutions but representations of solutions via their closed instances.

Definition 4.3. A unification problem $\mathcal{P}$ is *linear* is no meta-variable occurs more than once in $\mathcal{P}$. A matching problem $\mathcal{P}$ is *linear* is no meta-variable occurs more than once in the left-hand sides of the elementary equations of $\mathcal{P}$.

In the sequel, we will usually omit $=_\alpha$, and also restrict ourselves to linear equational problems.

Before to give the rules, we need the following preliminary definition:

Definition 4.4. A variable $x \in \mathcal{X}$ is *protected* in a pre-pattern u if all its occurrences in u belong to a pre-redex of u, that is, take place below F_u.

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 at 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 $\bot$ 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-Var*, the set of common 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. The role of *Meta-Var* is to ensure that the arity condition for meta-variables is met by the substitution that will be obtained if the algorithm succeeds.

The two green rules for matching are failure rules. *Fail-Arity* applies when the arity condition for meta-variables cannot be met, while *Fail-Protect* applies as soon as there is an equation whose right-hand side contains a free variable that does not occur in the left-hand side or in the initial problem. The two failure rules for unification require different conditions, in particular because unification is symmetric while matching is not. *Fail-Arity* treats equations which falsify the arity condition for meta-variables, while *Fail-Protect* deals with equations which cannot be unified because the right-hand side has an unprotected variable. Note that u cannot be a pre-redex in that case.

There are three remaining blue rules for unification. 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 there is some protected variable in the right-hand side that must be eliminated, as stipulated by the first condition. The other 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 $q = \wedge$, $|X| - |\overline{x}| = |Y| - |\overline{y}|$ and $\overline{x} \subsetneq \overline{y}$ then *Drop* applies to $X[\overline{x}] = Y[\overline{y}]$ and produces $X[\overline{x}] = Z[\overline{x}] \wedge Y[\overline{y}] = Z[\overline{x}]$ which could be improved in an implementation with a special instance of *Drop* to produce $Y[\overline{y}] = X[\overline{x}]$ only, as does *Flip*.

Dec-Fun	$f(\overline{u}) = f(\overline{v}) \longrightarrow \wedge_{i=1}^{i=\lvert f\rvert} u_i = v_i$	if $f \in \mathcal{F} \cup \mathcal{X}$
Dec-App	$(u\ s) = (v\ t) \longrightarrow u = v \wedge s = t$	if $(u\ s)$ and $(v\ t)$ are not pre-redexes
Dec-Abs	$\lambda x.u = \lambda y.v \longrightarrow u\{x \mapsto z\} = v\{y \mapsto z\}$	with z fresh
Conflict	$f(\overline{u}) = g(\overline{v}) \longrightarrow \bot$	if $f, g \in \mathcal{F} \cup \mathcal{X} \cup \{@, \lambda\},\ f \neq g$
Meta-Abs	$X[\overline{x}] = \lambda y.v \longrightarrow X[\overline{x}y] = v$	if $X \in \mathcal{Z},\ \lvert X\rvert > \lvert\overline{x}\rvert$
Fail-Arity	$X[\overline{x}] = u \longrightarrow \bot$	if $\lvert X\rvert - \lvert\overline{x}\rvert > ar(u)$
Fail-Protect	$X[\overline{x}] = u \longrightarrow \bot$	if $\exists z \in \mathcal{V}ar(u), z \notin \overline{x} \cup \mathcal{V}ar(P_0)$
Fail-Arity	$X[\overline{x}] = f(\overline{u}) \longrightarrow \bot$	if $\lvert X\rvert > \lvert\overline{x}\rvert \wedge f \in \{@\} \cup \mathcal{F} \cup \overline{x}$
Fail-Protect	$X[\overline{x}] = u \longrightarrow \bot$	if $\exists z \in \mathcal{V}ar(u), z \notin \overline{x}, z$ unprotected
Swap	$u = Y[\overline{y}] \longrightarrow Y[\overline{y}] = u$	if u is not a pre-redex
Flip	$X[\overline{x}] = Y[\overline{y}] \longrightarrow Y[\overline{y}] = X[\overline{x}]$	if $\lvert X\rvert - \lvert\overline{x}\rvert > \lvert Y\rvert - \lvert\overline{y}\rvert$
Drop	$X[\overline{x}] = u[Y[\overline{y}]]_q \longrightarrow X[\overline{x}] = u[Z[\overline{z}]]_q \wedge Y[\overline{y}] = Z[\overline{z}]$	if $\begin{cases} \overline{y} \not\subseteq \overline{x} \cup \mathcal{BV}ar(u) \\ u(\wedge) \notin \mathcal{F} \cup \{@, \lambda\} \vee \lvert X\rvert = \lvert\overline{x}\rvert \\ \text{unprotected variables of } u \text{ are in } \overline{x} \\ q \neq \wedge \vee \lvert Y\rvert - \lvert\overline{y}\rvert \geq \lvert X\rvert - \lvert\overline{x}\rvert \end{cases}$ with $\begin{cases} \overline{z} = \overline{y} \cap (\overline{x} \cup \mathcal{BV}ar(u)) \\ Z \text{ fresh s.t. } \lvert Z\rvert = \lvert Y\rvert - \lvert\overline{y}\rvert + \lvert\overline{z}\rvert \end{cases}$

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

Note that the set of rules can be easily transformed into a deterministic algorithm as no two rules apply to the same equation, except *Meta-Abs* and each of the failure rules but *Fail-Arity*.

Before to prove properties of these matching and unification rules, we show below examples of use of the unification rules that are useful for the reader's understanding:

Example 4.5. Let's illustrate some rules, using $\lvert X\rvert = 1, \lvert Y\rvert = 0, \lvert X'\rvert = 3, \lvert Y'\rvert = 2$ and $\lvert Z\rvert = 2$.

$$f(\lambda y.f(Y)) = f(X) \underset{Dec-Fun}{\longrightarrow} \lambda y.f(Y) = X \underset{Swap}{\longrightarrow} X = \lambda y.f(Y) \underset{Meta-Var}{\longrightarrow} X[y] = f(Y)$$
$$f(Y') = f(\lambda y.f(Y)) \underset{Dec-Fun}{\longrightarrow} Y' = \lambda y.f(Y) \underset{Meta-Var}{\longrightarrow} Y'[y] = f(Y) \underset{Fail-Arity}{\longrightarrow} \bot$$
$$X' = \lambda y.Y'[y] \underset{Meta-Var}{\longrightarrow} X'[y] = Y'[y] \underset{Flip}{\longrightarrow} Y'[y] = X'[y]$$
$$Y'[z] = \lambda x.X'[y, z] \underset{Dec-var}{\longrightarrow} Y'[z, x] = X'[y, z] \underset{Drop}{\longrightarrow} Y'[z, x] = Z[z] \wedge X'[y, z] = Z[z]$$

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 won't do all computations needed later on, only a few interesting ones originating from the first or second versions of that theory:

Example 4.6. We start with two matching problems, matching first the term $lk(\lambda w.ud(w, X'[w]))$ with the left-hand side $lk(\lambda w.ud(w, X[w]))$ of rule (lu) of the second set:

$$lk(\lambda w.ud(w, X[w])) = lk(\lambda w.ud(w, X'[w])) \underset{Dec-Fun}{\longrightarrow} \lambda w.ud(w, X[w])) = \lambda w.ud(w, X'[w])$$
$$\underset{Dec-Abs}{\longrightarrow} ud(w, X[w]) = ud(w, X'[w]) \underset{Dec-Fun}{\longrightarrow} w = w \wedge X[w] = X'[w] \underset{Dec-Fun}{\longrightarrow} X[w] = X'[w]$$

Now, the second, matching the term $lk(\lambda v.lk(Y'[v]))$ against the left-hand side $lk(\lambda v.lk(Y'[v]))$ of rule (ll) of the second set:

$$lk(\lambda v.lk(Y[v])) = lk(\lambda v.lk(Y'[v])) \underset{Dec-Fun}{\longrightarrow} \lambda v.lk(Y[v]) = \lambda v.lk(Y'[v]) \underset{Dec-Abs}{\longrightarrow}$$
$$lk(Y[v]) = lk(Y'[v]) \underset{Dec\text{-}Fun}{\longrightarrow} Y[v] = Y'[v]$$

It then follows that $lk(\lambda v.lk(Y'[v])) \underset{ll}{\longrightarrow} lk(\lambda v.Y'[v, v])$.

We go on with unification problems, first of the left-hand sides of rules (ll) and (l) from the first set:

$$lk(\lambda w.lk(\lambda v.Y[w, v])) = lk(\lambda w.U) \underset{Dec-Fun}{\longrightarrow} \lambda w.lk(\lambda v.Y[w, v]) = \lambda w.U \underset{Dec-Abs}{\longrightarrow}$$
$$lk(\lambda v.Y[w, v]) = U \underset{Swap}{\longrightarrow} U = lk(\lambda v.Y[w, v]) \underset{Drop}{\longrightarrow} U = lk(\lambda v.Z[v]) \wedge Y[w, v] = Z[v]$$
(with Z fresh of arity 1)

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

$$lk(\lambda v.ud(v, X[v])) = lk(\lambda v.U) \underset{Dec-Fun}{\longrightarrow} \lambda v.ud(v, X[v])) = \lambda v.U \underset{Dec-Abs}{\longrightarrow}$$
$$ud(v, X[v]) = U \underset{Swap}{\longrightarrow} U = ud(v, X[v]) \underset{Fail\text{-}Protect}{\longrightarrow} \bot$$

since v occurs unprotected as first argument of ud in $ud(v, X[v])$, making unification fail.

Finally comes unification of the left-hand sides of rule (l) with a subterm of the left-hand side of rule (ul) (still from the same set):

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

We can now carry out the same computations using the second set of rules. We get first:

$$lk(\lambda w.lk(Y[w])) = lk(\lambda w.U) \underset{Dec-Fun}{\longrightarrow} \lambda w.lk(Y[w]) = \lambda w.U \underset{Dec-Abs}{\longrightarrow} lk(Y[w]) = U \underset{Swap}{\longrightarrow} U = lk(Y[w])$$
$$\underset{Drop}{\longrightarrow} U = lk(Z) \wedge Y[w] = Z \qquad \text{(with } Z \text{ fresh of arity 1)}$$

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

$$lk(\lambda v.U) = lk(X) \underset{Dec-Fun}{\longrightarrow} \lambda v.U = X \underset{Swap}{\longrightarrow} X = \lambda v.U \underset{Meta-Var}{\longrightarrow} X[v] = U$$

We observe that the computations from the second set of rules are identical in the first two cases, but slightly different in the third case. On the contrary, the obtained solved form is the same in the third case, but slightly different in the first case for which the second one can be deduced from the first by an $M\eta$-step.

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 4.7. *Assume that en 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. □

The following sequence of properties shows that all rules are sound, that is preserve the solutions of equational problems.

We start with the three rules checking arities. We have seen that arities are non-decreasing under substitutions, and are even sometimes preserved. This is the basis for the soundness of the rules:

Lemma 4.8. *Assume that $|X| > |\overline{x}|$. Then the elementary unification problems $X[\overline{x}] = \lambda y.v$ and $X[\overline{x}y] = v$ have the same set of solutions.*

Proof. Without loss of generality, we assume that $y \notin \overline{x}$ and restrict our attention to solutions σ such that $y \notin \mathcal{D}om(\sigma)$. Since $|X| > |\overline{x}|$, a substitution for X must be of the form $\{X \mapsto \lambda\overline{x}z.u\}$. Such a substitution σ is a solution of $X[\overline{x}] = \lambda y.v$ iff $(\lambda z.u)\{\overline{x} \mapsto \overline{x}\} = \lambda z.u = (\lambda y.v)\sigma = \lambda y.v\sigma$, which holds true iff $u\{z \mapsto y\} = v\sigma$, which in turn holds true iff σ is a solution of $X[\overline{x}y] = v$. □

Lemma 4.9. *The elementary matching problem $X[\overline{x}] = u$ has no solution if $ar(X[\overline{x}]) > ar(u)$.*

Proof. By non-decreasingness, $ar(X[\overline{x}]\sigma) \geq ar(X[\overline{x}]) > ar(u)$. □

Lemma 4.10. *Assume that $|X| > |\overline{x}|$ and $f \in \mathcal{F} \cup \{@\} \cup \mathcal{X}$. Then, the elementary unification problem $X[\overline{x}] = f(\overline{u})$ has no solution.*

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

We now move to the case where extra-variables occur in right-hand sides, whether protected or not, starting with the cases of unprotected variables:

Lemma 4.11. *Let u be a term containing a variable $z \notin \overline{x} \cup \mathcal{V}ar(P)$. Then, the elementary matching problem $X[\overline{x}] = u$ has no solution in common with $\mathcal{P}$.*

Proof. Assume γ is a common solution for $X[\overline{x}] = u$ and $\mathcal{P}$. By definition of a solution of $\mathcal{P}$, $\gamma(X) = \lambda\overline{x}.v$ with $\mathcal{V}ar(v) \subseteq \overline{x} \cup \mathcal{V}ar(\mathcal{P})$. By assumption on z, $X[\overline{x}]\gamma$ and u have different sets of free variables, hence $X[\overline{x}]\gamma \neq_\alpha u$, hence contradicting our assumptions. □

Lemma 4.12. *Let u be a term containing an unprotected variable $z \notin \overline{x}$. Then, the elementary unification problem $X[\overline{x}] = u$ has no solution.*

Proof. Assume γ is a solution for $X[\overline{x}] = u$. By definition of a solution, $\gamma(X) = \lambda\overline{x}.v$ with $\mathcal{V}ar(v) \subseteq \overline{x}$, hence $z \notin \mathcal{V}ar(X[\overline{x}]\gamma)$. By definition again, $z \notin \mathcal{D}om(\gamma)$, hence $z\gamma = z$, and therefore $z \in \mathcal{V}ar(u\gamma)$. Since $X[\overline{x}]\gamma$ and $u\gamma$ have different sets of free variables, no α-renaming can make them equal, hence contradicting our assumption. □

Lemma 4.13. *Let E be the elementary unification problem $X[\overline{x}] = u[Y[\overline{y}]]_q$ and P the unification problem $X[\overline{x}] = u[Z[\overline{z}]]_q \wedge Y[\overline{y}] = Z[\overline{z}]$, where $\overline{z} = \overline{y} \cap (\overline{x} \cup \mathcal{BV}ar(u))$ and Z is a fresh variable of arity $|Y| - |\overline{y}| + |\overline{z}|$. Then, γ is a solution of P iff $\gamma_{\setminus Z}$ is a solution of E.*

Proof. Let $\overline{x'}$ (resp., $\overline{y'}$, $\overline{z'}$) be a vector of pairwise distinct fresh variables of length $|X| - |\overline{x}|$ (resp., $|Y| - |\overline{y}|$, $|Z| - |\overline{z}|$)), and $\gamma = \{X \mapsto \lambda\overline{x}\overline{x'}.w, Y \mapsto \lambda\overline{y}\overline{y'}.w', Z \mapsto \lambda\overline{z}\overline{y'}.w''\}$ be a solution of the generated problem. Remark that $\overline{y'}$ and $\overline{z'}$ have the same length so that we could actually identify them. By definition of a substitution, we get $\lambda\overline{x'}w = u\gamma[\lambda\overline{z'}.w'']_q$ and $\lambda\overline{y'}.w' = \lambda\overline{z'}.w''$, hence $\lambda\overline{x'}.w = u\gamma[\lambda\overline{y'}.w']_q$, showing that $\gamma_{\setminus Z}$ is a solution to the original problem.

Conversely, let $\gamma = \{X \mapsto \lambda\overline{x}\overline{x'}.w, Y \mapsto \lambda\overline{y}\overline{y'}.w'\}$ be a solution of the original problem. Using the previous remark, we can tentatively extend γ as γ' by letting $\gamma'(Z) = \lambda\overline{z}\overline{y'}.w'$. By definition of substitutions, $\lambda\overline{x'}.w = u\gamma[\lambda\overline{y'}.w']_q$. It follows that both sides of the equations have the same set of free variables, hence $\mathcal{V}ar(w') \subseteq \overline{y'} \cup \mathcal{BV}ar(u) \cup (\mathcal{V}ar(w) \setminus \overline{x'}) \subseteq \overline{y'} \cup \mathcal{BV}ar(u) \cup \overline{x} \cup \mathcal{V}ar(E) \subseteq$

$\overline{y'} \cup \overline{z} \cup \mathcal{V}ar(E)$. Hence $\mathcal{V}ar(\lambda\overline{z}\overline{y'}.w') \subseteq \mathcal{V}ar(E) = \mathcal{V}ar(P)$, and γ' satisfies the requirements to be a candidate solution of P. Showing that it satisfies P is routine. □

We can now conclude:

Lemma 4.14. *The matching/unification rules are terminating and preserve solutions of matching and unification problems.*

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's first component has decreased strictly. 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 $\overline{z} \subsetneq \overline{y}$.

Preservation of solutions follows from: for the first four rules, the fact that they apply above the fringe since they can't apply to an equation resulting from any other rule application; Lemma 4.8 for *Meta-Var*; Lemma 4.9 for *Fail-Arity*; Lemma 4.11 for *Fail-Protect*; Lemma 4.10 for *Fail-Arity*; Lemma 4.12 for *Fail-Protect*; commutativity of equality for *Swap* and *Flip*; and Lemma 4.13 for *Drop* (preservation is relative here to the variables of the initial problem). □

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

Definition 4.15 (Solved forms).

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

Lemma 4.16.

(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 $u = v$. Necessarily $u = X[\overline{x}]$ otherwise one of the *Dec* rules, *Conflict* or *Swap* rules would apply. Assuming $|\overline{x}| < |X|$, then v must be some pre-redex $Y[\overline{y}]$ otherwise *Meta-Abs*, *Fail-Arity* or *Fail-Arity* would apply. Because *Flip* doesn't apply by assumption, we conclude that $|X| - |\overline{x}| \leq |Y| - |\overline{y}|$, hence $u = v$ is in arity solved form.

(2) The application conditions of *Drop* imply that the *Dec* rules, *Conflict*, *Meta-Var*, *Swap* and *Flip* don't apply. From (1), it follows that *Drop* applies only on equations in arity solved form.

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

Lemma 4.17. *All rules preserve the following two properties of an unification problem P:*

(1) *For all equation not in arity solved form $u = v \in P$, meta-variables in $\mathcal{MV}ar(u) \cup \mathcal{MV}ar(v)$ are linear in P.*
(2) *For all equation in arity solved form $X[\overline{x}] = v \in P$, X is linear in P and all occurrences of a non-linear meta-variable Z in v are exclusively applied to variables in $\mathcal{V}ar(v) \cup \overline{x}$.*

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 $\overline{z} \subseteq \overline{y}$ and $\overline{z} \subseteq \overline{x} \cup \mathcal{BV}ar(u)$. □

Lemma 4.18. *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 4.16 (1), all equations in Q are in arity-solved form. By Lemma 4.17 (2), left-hand sides pre-redexes are linear in Q. We are left proving that for all $X[\overline{x}] = v \in Q$, $\mathcal{V}ar(v) \subseteq \overline{x}$. Assume it is not the case, and let $y \in \mathcal{V}ar(v) \setminus \overline{x}$. Either y occurs protected and *Drop* applies, or else *Fail-Protect* applies. □

Lemma 4.19. *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 bound by $|\mathcal{FP}os(u)|$, and yields a number of elementary equations whose whole size is bound 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 *Drop*s. Further the total number of applications of *Drop* to a given equation is bound by the number of protected variables to be eliminated, hence by its size. This shows that the whole unification process is linear. □

We are left extracting most-general unifiers from equational problems in solved form:

Lemma 4.20. *A solved form $P = \wedge_i X_i[\overline{x_i}] = v_i$ has a unique (up to renaming of bound variables) most general unifier $mgs(P) = \{X_i \mapsto \lambda\overline{x_i}\lambda\overline{z}.(v_i\ \overline{z})\}_i$, that is, every ground solution of P is a ground instance of σ.*

Proof. First, $X_i[\overline{x_i}]\sigma = \lambda\overline{z}.(v_i\ \overline{z})$. On the other hand, by definition of a solved form, $v_i\sigma = v_i$. Hence $X_i[\overline{x_i}]\sigma =_{\alpha\cup M\eta} v_i\sigma$, hence σ is a unifier of the solved form.

Instantiating the equation $v_i =_{M\eta} X_i[\overline{x_i}]\sigma$ with an arbitrary solution γ of P, we get $v_i\gamma = X_i[\overline{x_i}]\sigma\gamma$ ($M\eta$-steps disappear, as stressed in subsection 3.2). Using now the fact that γ is a solution yields $X_i[\overline{x_i}]\gamma = X_i[\overline{x_i}]\sigma\gamma$, showing that γ is an instance of σ by itself (as in the first-order case). □

Example 4.21. Consider the two solved forms obtained at example 4.6: $U = lk(\lambda v.Z[v]) \wedge Y[w, v] = Z[v]$ and $U = lk(Z) \wedge Y[w] = Z$, where $|Z| = 1$ for both cases. The most general solution is $\{U \mapsto lk(\lambda v.Z[v]), Y \mapsto \lambda wv.Z[v]\}$ for the first, and $\{U \mapsto lk(Z), Y \mapsto \lambda w.Z\}$ for the second. Consider now the matching solved form obtained at example 4.6: $X[w] = X'[w]$. The matching

687 substitution obtained is $\{X \mapsto \lambda w.X'[w]\}$. It would of course be possible to extract the better mgs
688 $\{X \mapsto X'\}$ to the price of some more technicalities.
689
690 Pattern matching and unification of linear patterns is therefore quite easy: first, reduce the initial
691 pattern matching problem $L = u$, or unification problem $L|_p = L'|_{p'}$, to a solved form. Then extract
692 the matching substitution or the most general unifier from the solved form. Therefore:
693
694 Theorem 4.22. *The matching problem results in a single matching substitution when it succeeds, computable in linear time. The unification problem results in a single (up to α) most general solution when it succeeds, computable in linear time.*
695
696
697 Note that there is no mention of types in this algorithm. A natural question is whether the most
698 general solution is typed when two linear patterns get unified, and whether it coincides with the
699 one obtained when unifying dependently typed linear patterns. This question will be addressed in
700 a forthcoming paper, in which the linearity restriction on patterns is removed.
701
702 ## 5 LOCAL PEAKS IN REWRITE THEORIES
703 Rewrite theories have two kinds of local ancestor peaks, *homogeneous* ones, between functional or
704 higher-order reductions, and *heterogeneous* ones, which mix both kinds of reductions. We analyze
705 here which local ancestor peaks enjoy *decreasing diagrams for free*, and which do not. Some results
706 in this section will be reused in forthcoming papers, those that do not rely on the left-linearity
707 assumption for the rewrite rules, nor on orthogonal functional reductions.
708
709 ### 5.1 Decreasing diagrams for free
710 A key property of plain first-order rewriting is that there are three possible kinds of local peaks
711 depending on the respective positions of the rewrites that define them. This property generalizes
712 trivially to higher-order rewrites with our definition of set of positions for patterns:
713
714 Lemma 5.1. *Given terms s, t such that $s \underset{i:L\to R}{\overset{p}{\longleftarrow}} u \underset{j:G\to D}{\overset{q}{\longrightarrow}} t$, then, there are three possibilities: (i) $p\#q$*
715 *(disjoint peak case); (ii) $q \geq_{\mathcal{P}} p \cdot F_L$ or $p \geq_{\mathcal{P}} q \cdot F_L$ (ancestor peak case); and (iii) $p = q \cdot o$ with*
716 *$o \in \mathcal{FP}os(L)$ or $q = p \cdot o$ with $o \in \mathcal{FP}os(G)$ (overlapping peak case).*
717
718 In the case of plain rewriting, two non-overlapping rewrite steps issuing from a same term
719 commute, a major component of any confluence proof. When the steps occur at disjoint positions,
720 this property, which holds for any monotonic relation, remains true for rewriting modulo a theory,
721 hence all disjoint peaks have decreasing diagrams for free. This is not the case, however, when the
722 steps occur at positions whose one is an ancestor of the other, because the modulo part of the above
723 rewrite may interact with the rewrite below. Our definition of higher-order rewriting, however,
724 enjoys a similar property, because the fringe of a rewrite step protects positions below it.
725 In the coming lemma, "LAP" stands for *linear ancestor peak*, and "a" for *above*, the β-step being
726 above a higher-order step. It applies to any higher-order rule, left-linear or not.
727
728 Lemma 5.2 (LAPβA). *Let u be a term, $p, q \in \mathcal{P}os(u)$ such that $q \geq_{\mathcal{P}} p \cdot F_\beta$ and $s \underset{\beta}{\overset{p}{\longleftarrow}} u \underset{j\in\mathcal{R}}{\overset{q}{\longrightarrow}} t$. Then*
729
730 *$s \underset{j}{\overset{Q}{\Longrightarrow}} \underset{\beta}{\overset{p}{\longleftarrow}} t$ for some set Q of parallel positions of s such that $Q \geq_{\mathcal{P}} p$.*
731
732 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
733 are two cases:
734
735

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

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 3.12, $u|_q\sigma \xrightarrow[j]{} t|_q\sigma$, hence, by Lemma 3.11, $s \xrightarrow[j]{} u[M\sigma[t|_q\sigma]_{q'}]_p$. On the other hand, $t|_p = (\lambda x.P\ N)$, where $P = M[t|_q]_{q'}$, therefore $t = u[(\lambda x.P\ N)]_p \xrightarrow[\beta]{p} u[P\sigma]_p = u[M\sigma[t|_q\sigma]_{q'}]_p$. We are done. □

The case of a local peak $s \xleftarrow[i]{p} u \xrightarrow[j]{q} t$, where the higher-order step with $i : L \to R$ 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 $u \xrightarrow[i:L\to R]{p} v$ with L linear and $q \in \mathcal{P}os(u)$ such that $q \geq_{\mathcal{P}} p \cdot F_L$. Then $\underline{u}_q = u[Z[\overline{z}]]_q \xrightarrow[i]{p} w$ for some w, where $\overline{z}$ is the list of variables bound above q in u, Z fresh, $|Z| = |\overline{z}|$, and $v = w\overline{u}^q = w\{Z \mapsto \lambda\overline{z}.u|_q\}$.*

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

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

By definition of higher-order rewriting, $u|_p = L\gamma$ for some substitution γ. By definition of a substitution, $\gamma(X) = \lambda\overline{x}.M$, and by the previous property, $X[\overline{x}]\gamma = M = u|_{p\cdot o}$, hence $M|_{q'} = u|_q$. As a consequence, $u|_{p\cdot o}[Z[\overline{z}]]_{q'} = M[Z[\overline{z}]]_{q'}$.

Let now θ be the substitution identical to γ except for the meta-variable X for which $\theta(X) = \lambda\overline{x}.M[Z[\overline{z}]]_{q'}$. We have $\theta(X)\tau = \lambda\overline{x}.M[u|_q]_{q'} = \lambda\overline{x}.M[M|_{q'}]_{q'} = \lambda\overline{x}.M = \gamma(X)$, hence $\gamma = \theta\tau$.

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

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

As already said, (LAP$\mathcal{R}$a) requires the linearity assumption.

Lemma 5.4 (LAP$\mathcal{R}$a). *Let $\mathcal{R}$ be a left-linear rewrite system, $i : L \to R \in \mathcal{R}$, $j \in \mathcal{R} \cup \beta$, u be a term, and $p, q \in \mathcal{P}os(u)$ such that $q \geq_{\mathcal{P}} p \cdot F_L$ and $s \xleftarrow[i]{p} u \xrightarrow[j]{q} t$. Then, $s \overset{\geq_{\mathcal{P}} p}{\underset{j}{\twoheadrightarrow}} \xleftarrow[i]{p} t$.*

Proof. Splitting u at q yields $u = v\sigma$, where $v = \underline{u}_q = u[Z[\overline{z}]]_q$ and $\sigma = \overline{u}^q = \{Z \mapsto \lambda\overline{z}.u|_q\}$ is preserving since $u|_q$ cannot be an abstraction by definition of a pattern and is normal as a subterm of $u|_p$. By assumption, $u|_q \xrightarrow[j]{} t|_q$, and by monotonicity, $\sigma(Z) = \lambda\overline{z}.u|_q \xrightarrow[j]{} \lambda\overline{z}.t|_q$. Let τ be σ with the exception $\tau(Z) = \lambda\overline{z}.t|_q$. Then $\sigma \xrightarrow[j]{} \tau$ and by Lemma 3.11, $u = v\sigma \xrightarrow[j]{} v\tau = t$. By Lemma 5.3, $v \xrightarrow[i]{p} w$ for some w such that $s = w\sigma$. By Lemma 3.13, $u = v\sigma \xrightarrow[j]{} w\tau$. The result follows. □

This ancestor peak property is more complex than for first-order computations since we need the assumption that the rewrite rules are left-linear, which is of course true of all first-order rewrite rules, and is true as well of all rules of the theory of global states.

Note that whenever X occurs self-nested in a right-hand-side, $R[X[\overline{u}]]_p$, such that $\overline{u_i}^q = \lambda\overline{z}.X[\overline{t}]$ for some i, this right-hand-side can be replaced with $R[\ (\lambda x.X[\overline{v}]\ \ \lambda\overline{z}.X[\overline{t}])\ ]_p$ with $v_j = u_j$ for $i \neq j$ and $v_i = u_i[(x\ \overline{z})]_q$. For instance, instead of using the rule $f(\lambda x.X[x]) \longrightarrow X[g(\lambda z.X[h(z)])]$ one may use $f(\lambda x.X[x]) \longrightarrow (\lambda x.X[g(\lambda z.(x\ z))]\ \ \lambda z.X[h(z)])$. In case the rule has critical pairs, they will most presumably require to be joined with β-steps which make this transformation of little interest.

5.2 Critical peaks

We can now define critical peaks:

Definition 5.5. Let $i : L \to R$ and $j : G \to D$ be two rules in $\mathcal{R}$ and $o \in \mathcal{FP}os(L)$ such that the equation $L|_o = G$ has a most general solution σ. Then, the peak $R\sigma \xleftarrow[i]{\Lambda} L\sigma \xrightarrow[j]{o} L\sigma[D\sigma]_o$ is called a *critical peak* of j onto i at position o. Its associated *critical pair* is $\langle R\sigma, L\sigma[D\sigma]_o\rangle$.

This definition makes sense: since $o \in \mathcal{FP}os(L)$, then $o <_{\mathcal{P}} F_L$, and therefore, $o \in \mathcal{FP}os(L\sigma)$. 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 5.6 (Critical pair lemma). *Assume $s \xleftarrow[i]{p} u \xrightarrow[j]{q} t$ is an overlapping peak of $j : G \to D$ onto $i : L \to R$ at position $o \in \mathcal{FP}os(L)$ such that $p = q \cdot o$. Then, there is a critical peak $s' \xleftarrow[i]{\Lambda} u' \xrightarrow[j]{o} t'$ and a substitution θ such that $u'\theta = u|_p$, $s'\theta = s|_p$ and $t'\theta = t|_p$.*

Thanks to our definition of higher-order rewriting, the proof is similar to the first-order case:

Proof. By definition of higher-order rewriting, there exists some substitution γ such that $L\gamma = u|_p$, $G\gamma = u|_q$, $s|_p = R\gamma$ and $t|_q = D\gamma$. Since $o \in \mathcal{FP}os(L)$, then $o <_{\mathcal{P}} F_L$, and since $L\gamma = u|_p$, then $u|_p[\ ]_o = L\gamma[\ ]_o$, hence $t|_p = u|_p[t|_q]_o = (L\gamma[D\gamma]_o) = L[D]_o\gamma$.

Since $o \in \mathcal{FP}os(L)$, $(L\gamma)|_o = L|_o\gamma$, hence $L|_o\gamma = G\gamma$. Therefore, γ is a solution of the equation $L|_o = G$. Let σ be its most general unifier. Then, there exist a substitution θ such that $\sigma\theta = \gamma$. Hence $u[R\sigma\theta]_p = s$ and $u[L[D]_o\sigma\theta] = t$. Since $s' = R\sigma$ and $t' = L\sigma[D\sigma]_o$, we get the result. □

5.3 Orthogonal functional reductions

The confluence proof will not be based on using β-rewrites, nor parallel β-rewrites, but, as in Tait's confluence proof of the lambda-calculus, orthogonal β-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 5.7. Given a set of parallel positions P and a family Q of sets of positions strictly below Λ 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 5.8. *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. $P = \{p \in S \mid \forall q \in S, p \not> q\}$ and $Q_p = \{q > \Lambda \mid p \cdot q \in S\}$ □

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

Definition 5.9 (Orthogonal reductions). Orthogonal rewriting, written $u \overset{O}{\underset{\beta}{\otimes\Longrightarrow}} v$, is the smallest reflexive relation on terms such that $u \overset{\overline{O}}{\underset{\beta}{\otimes\Longrightarrow}} s$ and $s \overset{\underline{O}}{\underset{\beta}{\Longrightarrow}} v$ imply $u \overset{O}{\underset{\beta}{\otimes\Longrightarrow}} v$.

The alternative choice of rewriting first at positions in $\underline{O}$, instead of first in $\overline{O}$, would yield a more complex calculation for O, explaining our 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: $t \underset{\beta}{\overset{P}{\otimes\!\!\Longrightarrow}} u \underset{\beta}{\overset{Q}{\otimes\!\!\Longrightarrow}} v$ implies $t \underset{\beta}{\overset{P\cup Q}{\otimes\!\!\Longrightarrow}} v$ if $Q \not\geq_{\mathcal{P}} P$, which is easily proved by induction on P.

We shall need several well-known properties of (Tait's) orthogonal β-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 5.10. *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 $u \underset{\beta}{\overset{O}{\otimes\!\!\Longrightarrow}} v$ iff $u \underset{\beta}{\overset{P}{\otimes\!\!\Longrightarrow}} \underset{\beta}{\overset{Q}{\otimes\!\!\Longrightarrow}} \underset{\beta}{\overset{R}{\otimes\!\!\Longrightarrow}} v$.*

Proof. It suffices to notice that $\underset{\beta}{\overset{P}{\otimes\!\!\Longrightarrow}}$ preserves Q and R and $\underset{\beta}{\overset{Q}{\otimes\!\!\Longrightarrow}}$ preserves R. □

Lemma 5.11. *$(\lambda x.M\ N) \underset{\beta}{\overset{O}{\otimes\!\!\Longrightarrow}} t$ with $\Lambda \in O$ if and only if $O = \{\Lambda\} \uplus 11\cdot P \uplus 2\cdot Q$, $M \underset{\beta}{\overset{P}{\otimes\!\!\Longrightarrow}} M'$, $N \underset{\beta}{\overset{Q}{\otimes\!\!\Longrightarrow}} N'$ and $t = M'\{x \mapsto N'\}$.*

5.4 Orthogonal decreasing diagrams for free

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

Lemma 5.12 (LAPOb). *Let $s \underset{\mathcal{R}ll}{\overset{q}{\longleftarrow}} u \underset{\beta}{\overset{O}{\otimes\!\!\Longrightarrow}} t$ for some set $O >_{\mathcal{P}} q$. Then, $s \underset{\beta}{\overset{\geq_{\mathcal{P}} q}{\otimes\!\!\Longrightarrow}} r \underset{\mathcal{R}ll}{\overset{q}{\longleftarrow}} t$.*
Besides $u|_q = L\gamma$ and $s = u[R\gamma]_q$ such that $\gamma \underset{\beta}{\otimes\!\!\Longrightarrow} \sigma$, $t = u[L\sigma]_q$ and $r = u[R\sigma]$.

Proof. Since patterns are β-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), $\underline{u}_{\underline{O}} \underset{\mathcal{R}ll}{\overset{q}{\longrightarrow}} s'$ for some s' such that $s = s'\overline{u}^{\underline{O}}$. By definition of orthogonal rewriting, $\overline{u}^{\underline{O}} \underset{\beta}{\otimes\!\!\Longrightarrow} \gamma$ for some γ such that $t = \underline{u}_{\underline{O}}\gamma$.

We conclude that both $s = s'\overline{u}^{\underline{O}} \underset{\beta}{\otimes\!\!\Longrightarrow} s'\gamma$ and, by stability, $t = \underline{u}_{\underline{O}}\gamma \underset{\mathcal{R}ll}{\overset{q}{\longrightarrow}} s'\gamma$. □

Definition 5.13. A non-empty set of position $O \subseteq \mathcal{P}os(u)$ is said to be *rigid* in u if there exists $q \leq_{\mathcal{P}} O$ such that $\mathcal{V}ar(u|_O) \subseteq \mathcal{V}ar(u|_q)$.

If Q is rigid in u, we can always choose $q = glb(O)$, the greatest lower bound of O w.r.t. $\leq_{\mathcal{P}}$.
Note also that a position $o \in \mathcal{P}os(u)$ is a singleton set of rigid positions in u.

Lemma 5.14 (LAPOa). *Let $s \underset{\beta}{\overset{P}{\Longleftarrow\!\!\otimes}} u \underset{\mathcal{R}ll}{\overset{q}{\longrightarrow}} t$, where $P \not\geq_{\mathcal{P}} q$. Then $s \underset{\mathcal{R}ll}{\overset{Q'}{\Longrightarrow}} v \underset{\beta}{\overset{P}{\Longleftarrow\!\!\otimes}} t$ where $Q' \geq_{\mathcal{P}} \underline{(P \cup \{q\})}$.*

This implies that $s = u'\overline{u}^q$, $v = u'\sigma$ and $t = \underline{u}_q\sigma$ such that $\underline{u}_q \underset{\beta}{\overset{P}{\otimes\!\!\Longrightarrow}} u'$, $\overline{u}^q \underset{\mathcal{R}ll}{\longrightarrow} \sigma$.

Note that $\underline{(P \cup \{q\})}$ 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 $Q' \geq_{\mathcal{P}} \underline{(P \cup \{q\})}$, which is therefore more precise.

Proof. *We prove a more general result for which* $u \overset{Q}{\underset{\mathcal{R}ll}{\Longrightarrow}} t$, *with* $P \not\geq_{\mathcal{P}} Q$ *and* Q *is a set of rigid positions of* u. *We then conclude that* $s \overset{Q'}{\underset{\mathcal{R}ll}{\Longrightarrow}} v$ *for some set* Q' *of positions of* s *such that* $Q' \geq_{\mathcal{P}} \underline{(P \cup Q)}$.

We prove the result by induction on the set of positions P *using the well-founded multiset extension* $>_{mul}$ *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* $u = f(\overline{u})$ *and* $f(\overline{s}) \overset{P}{\underset{\beta}{\Longleftarrow\!\!\otimes}} f(\overline{u}) \overset{Q}{\underset{\mathcal{R}ll}{\longrightarrow}} f(\overline{t})$ *with* $s_i \overset{P_i}{\underset{\beta}{\Longleftarrow\!\!\otimes}} u_i \overset{Q_i}{\underset{\mathcal{R}ll}{\Longrightarrow}} t_i$. *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,* $glb(Q) = \Lambda$ *and* $\mathcal{V}ar(u_i|_{Q_i}) \subseteq \mathcal{V}ar(u)$, *the latter property being preserved by rewriting. Since* $P_i \prec_{mul} P$, *by induction hypothesis,* $s_i \overset{Q'_i}{\underset{\mathcal{R}ll}{\Longrightarrow}} v_i \overset{P_i}{\underset{\beta}{\Longleftarrow\!\!\otimes}} t_i$, *where* $Q'_i \geq_{\mathcal{P}} \underline{P_i \cup Q_i}$. *The orthogonal* β*-steps can be grouped together into* $v \overset{P}{\underset{\beta}{\Longleftarrow\!\!\otimes}} t$. *For the* $\mathcal{R}ll$*-steps, let* $Q' = \bigcup_i i \cdot Q'_i$, *hence* $s = f(\overline{s}) \overset{Q'}{\underset{\mathcal{R}ll}{\Longrightarrow}} f(\overline{v}) = v$. *From* $Q'_i \geq_{\mathcal{P}} \underline{P_i \cup Q_i}$ *we deduce* $Q' \geq_{\mathcal{P}} \underline{P \cup Q}$, *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* $P_1, Q_1 \in \mathcal{P}os(M)$ *and* $P_2, Q_2 \in \mathcal{P}os(N)$ *such that* $M_w \overset{P_1}{\underset{\beta}{\Longleftarrow\!\!\otimes}} M \overset{Q_1}{\underset{\mathcal{R}ll}{\Longrightarrow}} M_t$ *and* $N_w \overset{P_2}{\underset{\beta}{\Longleftarrow\!\!\otimes}} N \overset{Q_2}{\underset{\mathcal{R}ll}{\Longrightarrow}} N_t$. *Since* $\underline{P \cup Q} = \{\Lambda\}$, *we only need to show that* $s \overset{Q'}{\underset{\mathcal{R}ll}{\Longrightarrow}} v \overset{P}{\underset{\beta}{\Longleftarrow\!\!\otimes}} t$ *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* $w = (\lambda x.M_w\ \ N_w) \overset{P'}{\underset{\beta}{\Longleftarrow\!\!\otimes}} u \overset{Q}{\underset{\mathcal{R}ll}{\Longrightarrow}} t$. *Since* $P' \not\geq_{\mathcal{P}} Q$ *and* $P' \prec_{mul} P$, *by induction hypothesis,* $w \overset{Q''}{\underset{\mathcal{R}ll}{\Longrightarrow}} (\lambda x.M_v\ N_v) \overset{P'}{\underset{\beta}{\Longleftarrow\!\!\otimes}} t$. *Hence* $M_w \overset{Q''_1}{\underset{\mathcal{R}ll}{\Longrightarrow}} M_v$ *and* $N_w \overset{Q''_2}{\underset{\mathcal{R}ll}{\Longrightarrow}} N_v$.

(a) $Q_1 \neq \emptyset$, *hence* $glb(Q) = \Lambda$. *Since* Q *is a set of rigid positions in* u, *no variable bound above* Q_1 *in* u *can occur in* $M|_{Q_1}$. *It follows that* $x \notin \mathcal{V}ar(M|_{Q_1})$ *and furthermore that* β*-reductions at* P_1 *do not instantiate terms at* Q_1, *and therefore* $x \notin \mathcal{V}ar(M_w|_{Q'_1})$. *By repeated applications of Lemma 3.11, it follows that* $s = M_w\{x \mapsto N_w\} \overset{Q'}{\underset{\mathcal{R}ll}{\Longrightarrow}} M_v\{x \mapsto N_v\}$, *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* $s \overset{Q'}{\underset{\mathcal{R}ll}{\Longrightarrow}} M_v\{x \mapsto N_v\}$, *where* $Q' = \{o \cdot Q''_2 : M_w|_o = x\}$ *is a set of parallel positions.*

Since $t \overset{P' \uplus \{\Lambda\}}{\underset{\beta}{\otimes\!\!\Longrightarrow}} v = M_v\{x \mapsto N_v\}$, *we are done in both cases.*

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

$$u = (\lambda x.M\ N) \overset{2 \cdot P_2}{\underset{\beta}{\otimes\!\!\Longrightarrow}} (\lambda x.M\ N_w) = w \overset{\Lambda}{\underset{\beta}{\longrightarrow}} M\{x \mapsto N_w\} = \overset{P_1}{\underset{\beta}{\otimes\!\!\Longrightarrow}} M_s\{x \mapsto N_w\} = s$$

rewrites at $1^2 \cdot Q_1 \ \#\ 2 \cdot P_2$, $(\lambda x.M\ N_w) \overset{1^2 \cdot Q_1}{\underset{\mathcal{R}ll}{\Longrightarrow}} (\lambda x.M_t\ N_w) \overset{2 \cdot P_2}{\underset{\beta}{\Longleftarrow\!\!\otimes}} (\lambda x.M_t\ N) = t$. *By stability Lemma 3.12 used at all positions in* Q_1, $M\{x \mapsto N_w\} \overset{Q_1}{\underset{\mathcal{R}ll}{\Longrightarrow}} M_t\{x \mapsto N_w\} \overset{\Lambda}{\underset{\beta}{\longleftarrow}} (\lambda x.M_t\ N_w)$. *Since* $P \not\geq_{\mathcal{P}} Q$, *then* $P_1 \not\geq_{\mathcal{P}} Q_1$. *Besides, since* Q *is rigid in* u *and* $Q_2 = \emptyset$, *then* Q_1 *must be rigid in* M *and since substitutions don't capture variables,* Q_1 *is rigid in* $M\{x \mapsto N_w\}$. *We can therefore*

apply the induction hypothesis to $M_s\{x \mapsto N_w\} \underset{\beta}{\overset{P_1}{\Longleftarrow\!\!\otimes}} M\{x \mapsto N_w\} \underset{\mathcal{R}ll}{\overset{Q_1}{\Longrightarrow}} M_t\{x \mapsto N_w\}$, *which gives* $s = M_s\{x \mapsto N_w\} \underset{\mathcal{R}ll}{\overset{Q'}{\Longrightarrow}} M'\{x \mapsto N_w\} \underset{\beta}{\overset{P_1}{\Longleftarrow\!\!\otimes}} M_t\{x \mapsto N_w\} = v$, *where* $P \cup Q \geq_{\mathcal{P}} Q'$. *Now* $t = (\lambda x.N_t\ N) \underset{\beta}{\overset{2\cdot P_2 \uplus \Lambda \uplus 1^2 \cdot P_1 = P}{\otimes\!\!\Longrightarrow}} v''$ *and we are done.* □

6 CONFLUENCE IN $\lambda\mathcal{F}$

We can now address the problem of confluence of a higher-order rewrite theory $\lambda\mathcal{F}$.

We assume given a set $\mathcal{R}$ll of left-linear rewrite rules, and will consider the case where the relation generated by $\mathcal{R}$ll is terminating. The other rewrite relations to be considered are $=_\alpha$ and β. For β, we shall need orthogonal reductions, as previously defined, introduced by Tait under the name of parallel reductions for showing confluence of the λ-calculus.

Theorem 6.1. *Let* **($\mathcal{R}$ll)** *be a terminating, left-linear rewrite system whose all critical pairs are joinable with* **$\mathcal{R}$ll**. *Then* $\lambda\mathcal{F}$ *is confluent.*

Proof. *Let* $\longrightarrow = \underset{\mathcal{R}ll}{\longrightarrow} \cup \underset{\beta}{\otimes\!\!\Longrightarrow}$. *The relations* $\longrightarrow$ *and* $\underset{\lambda\mathcal{F}}{\longrightarrow}$ *verify the assumptions of the following well-known property: assume* $\overset{1}{\longleftarrow\!\!\!\!\rightarrow} \subseteq \overset{2}{\longleftarrow\!\!\!\!\rightarrow}$ *and* $\overset{2}{\longrightarrow} \subseteq \overset{1}{\longrightarrow}$, *then Church-Rosser of* $\overset{2}{\longrightarrow}$ *implies Church-Rosser of* $\overset{1}{\longrightarrow}$. *We will therefore apply van Oostrom's decreasing diagram method to the relation* $\longrightarrow$ *and conclude that* $\underset{\lambda\mathcal{F}}{\longrightarrow}$ *is confluent. To this end, we use for labels ordered pairs defined as follows:* $\langle 0, u\rangle$ *for* $u \underset{\mathcal{R}ll}{\longrightarrow} v$; *and* $\langle 1, \bot\rangle$ *for* $u \underset{\beta}{\otimes\!\!\Longrightarrow} v$, $\bot$ *being a don't care constant. Labels are compared lexicographically, the first argument in the order on natural numbers, the second in the order* $\underset{\mathcal{R}ll}{\longrightarrow\!\!\!\!\rightarrow}$, $\bot$ *being chosen minimal.*

We now show that all local peaks have decreasing diagrams.

Let $s \overset{P}{\longleftarrow} u \overset{Q}{\longrightarrow} t$ *be an arbitrary local peak, where* P, Q *are either a set of orthogonal positions (for* β*) or a single position (for* $\mathcal{R}ll$*).*

(1) *First, rewrite steps of monotonic relations always commute when* $P\#Q$, *yielding a DD.*

We are now left with all peaks for which $\neg(P\#Q)$, *which we consider in turn:*

(2) $s \underset{\beta}{\overset{P}{\Longleftarrow\!\!\otimes}} u \underset{\beta}{\overset{Q}{\otimes\!\!\Longrightarrow}} t$. *Orthogonal* β*-reductions are known to be joinable in at most one step from each side, hence* $s \underset{\beta}{\overset{P'}{\otimes\!\!\Longrightarrow}} v \underset{\beta}{\overset{Q'}{\Longleftarrow\!\!\otimes}} t$ *for some* P', Q', v, *a DD.*

(3) $s \underset{i}{\overset{p}{\longleftarrow}} u \underset{j}{\overset{q}{\longrightarrow}} t$ *with* $i : L \to R \in \mathcal{R}ll$, $j \in \mathcal{R}ll$ *and* $q \geq_{\mathcal{P}} p \cdot F_L$. *Lemma 5.4 (LAP*$\mathcal{R}$*a) yields a decreasing diagram without facing steps because labels decrease along reductions.*

(4) $s \underset{i}{\overset{p}{\longleftarrow}} u \underset{j}{\overset{q}{\longrightarrow}} t$ *with* $i : L \to R \in \mathcal{R}ll$, $j : G \to D \in \mathcal{R}ll$, $q \in p \cdot o$ *and* $o \in \mathcal{FP}os(F_L)$. *By Lemma 5.6, there is a critical peak obtained by overlapping* G *onto* L *at position* o. *By assumption, this peak is joinable with rules of* $\mathcal{R}ll$, *hence the pair* s, t *is joinable by the monotonicity lemma 3.11 and the stability Lemma 3.12. Note that there are no facing steps here, since labels decrease strictly along* $\mathcal{R}ll$*-reductions.*

(5) $s \underset{\beta}{\overset{O}{\Longleftarrow\!\!\otimes}} u \underset{i}{\overset{q}{\longrightarrow}} t$, *where* $i : L \to R \in \mathcal{R}ll$. *The proof of this case is shown at Figure 4.*

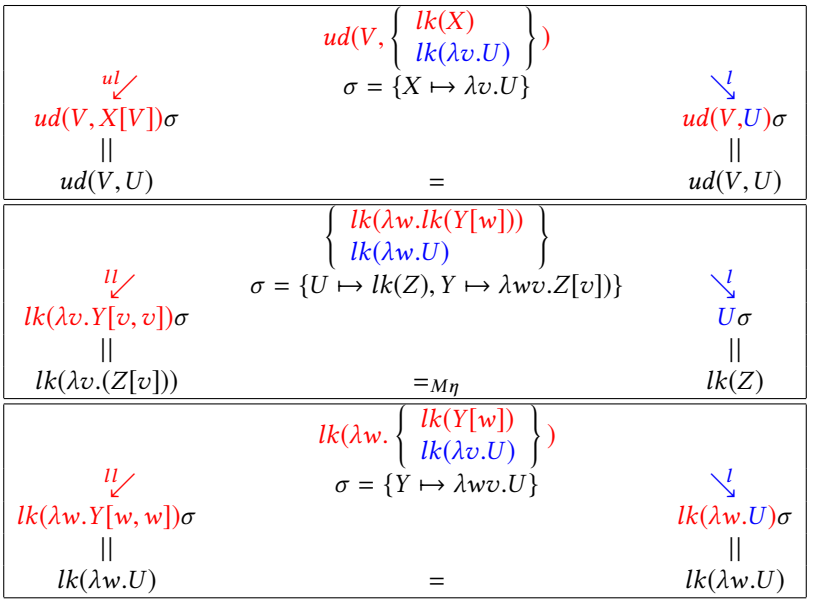
From Lemma 5.10 we have $s \underset{\beta}{\overset{R}{\Longleftarrow\!\!\otimes}} v \underset{\beta}{\overset{Q}{\Longleftarrow\!\!\otimes}} u' \underset{\beta}{\overset{P}{\Longleftarrow\!\!\otimes}} u$ *with* $O = P \uplus Q \uplus R$ *such that* $P\#q$, $Q \geq_{\mathcal{P}} q$ *and* $R <_{\mathcal{P}} q$. *Besides, since* $u(q) \in \mathcal{F}$, *then* $q \notin O$ *and* $Q >_{\mathcal{P}} q$. *By commutation we easily get* $u' \underset{i}{\overset{q}{\longrightarrow}} t' \underset{\beta}{\overset{P}{\Longleftarrow\!\!\otimes}} t$. *By Lemma 5.12 we have* $v \underset{i}{\overset{q}{\longrightarrow}} r \underset{\beta}{\overset{\geq_{\mathcal{P}} q}{\Longleftarrow\!\!\otimes}} t'$ *and by Lemma 5.14 we have* $s \underset{i}{\Longrightarrow} w \underset{\beta}{\overset{R}{\Longleftarrow\!\!\otimes}} r$. *From Lemma 5.10 again, all three* β*-steps can be merged into a single orthogonal facing step* $w \underset{\beta}{\Longleftarrow\!\!\otimes} t$. *The step* $\underset{i}{\Longrightarrow} w$ *can then be linearized, hence we get a DD.*

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

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

Example 6.2. SOL 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 it's untyped version is terminating, and then, that the critical pairs are joinable. In the absence of $\beta^{\neq 0}$, first-order termination techniques can do. We are left with verifying the joinability of critical pairs, 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 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.

Since the most general unifiers are identical for both choices of rules, we choose the second set. The computations are not identical for both sets, since they will actually 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 exactly.



$$
\begin{array}{ccc}
 & lk(\lambda w.\left\{\begin{array}{l} lk(Y[w]) \\ lk(\lambda v.lk(Y'[v])) \end{array}\right.) & \\
 & \sigma = \{Y \mapsto \lambda w v.lk(Y'[v])\} & \\
\overset{ll}{\swarrow} & & \overset{ll}{\searrow} \\
lk(\lambda v.Y[v,v])\sigma & & lk(\lambda w.lk(\lambda v.Y'[v,v]))\sigma \\
\| & & \| \\
lk(\lambda v.lk(Y'[v])) & \overset{ll}{\longrightarrow}\ lk(\lambda v.Y'[v,v])\ \overset{l}{\longleftarrow} & lk(\lambda w.lk(\lambda v.Y'[v,v]))
\end{array}
$$

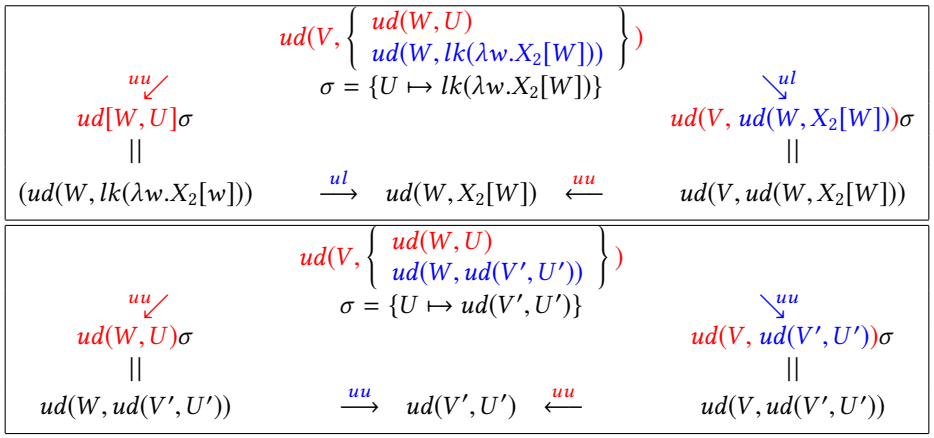
$$
\begin{array}{ccc}
 & lk(\lambda w.\left\{\begin{array}{l} lk(Y[w]) \\ lk(\lambda v.ud(v,X[v])) \end{array}\right.) & \\
 & \sigma = \{Y \mapsto \lambda w v.ud(v,X[v])\} & \\
\overset{ll}{\swarrow} & & \overset{lu}{\searrow} \\
lk(\lambda v.Y[v,v])\sigma & & lk(\lambda w.lk(X))\sigma \\
\| & & \| \\
lk(\lambda v.ud(v,X[v])) & \overset{lu}{\longrightarrow}\ lk(\lambda v.X[v]) =_{M\eta} lk(X)\ \overset{l}{\longleftarrow} & lk(\lambda w.lk(X))
\end{array}
$$

$$
\begin{array}{ccc}
 & lk(\lambda w.\left\{\begin{array}{l} ud(w,X[w]) \\ ud(V,lk(X')) \end{array}\right.) & \\
 & \sigma = \{V \mapsto w, X \mapsto \lambda w.lk(\lambda v.X'[v])\} & \\
\overset{lu}{\swarrow} & & \overset{ul}{\searrow} \\
lk(X)\sigma & & lk(\lambda w.ud(V,X'[V]))\sigma \\
\| & & \| \\
lk(\lambda w.lk(X')) & \overset{l}{\longrightarrow}\ lk(\lambda v.X'[v]) =_{\alpha} lk(\lambda w.X'[w])\ \overset{lu}{\longleftarrow} & lk(\lambda w.ud(w,X'[w]))
\end{array}
$$

$$
\begin{array}{ccc}
 & lk(\lambda w.\left\{\begin{array}{l} ud(w,X[w]) \\ ud(U,ud(V,W)) \end{array}\right.) & \\
 & \sigma = \{U \mapsto w, X \mapsto \lambda w.ud(V,W)\} & \\
\overset{lu}{\swarrow} & & \overset{uu}{\searrow} \\
lk(\lambda w.X[w])\sigma & & lk(\lambda w.ud(V,W))\sigma \\
\| & & \| \\
lk(\lambda w.lk(\lambda w.ud(V,W))) & \overset{ll}{\longrightarrow} & lk(\lambda w.ud(V,W))
\end{array}
$$

$$
\begin{array}{ccc}
 & ud(V,\left\{\begin{array}{l} lk(\lambda v.X_1[v]) \\ lk(\lambda v.ud(v,X_2[v])) \end{array}\right.) & \\
 & \sigma = \{X_1 \mapsto \lambda v.ud(v,X_2[v])\} & \\
\overset{ul}{\swarrow} & & \overset{lu}{\searrow} \\
ud(V,X_1[V])\sigma & & ud(V,\ lk(\lambda v.X_2[v]))\sigma \\
\| & & \| \\
ud(V,ud(V,X_2[V])) & \overset{uu}{\longrightarrow}\ ud(V,X_2[V])\ \overset{ul}{\longleftarrow} & ud(V,lk(\lambda v.X_2[v]))
\end{array}
$$

$$
\begin{array}{ccc}
 & ud(V,\left\{\begin{array}{l} lk(\lambda v.X_1[v]) \\ lk(\lambda v.lk(\lambda w.X_2[w,v])) \end{array}\right.) & \\
 & \sigma = \{X_1 \mapsto \lambda v.lk(\lambda w.X_2[w,v])\} & \\
\overset{ul}{\swarrow} & & \overset{ll}{\searrow} \\
ud(V,X_1[V])\sigma & & ud(V,\ lk(\lambda w.X_2[w,w]))\sigma \\
\| & & \| \\
ud(V,lk(\lambda w.X_2[w,V]))) & \overset{ul}{\longrightarrow}\ ud(V,X_2[V,V])\ \overset{ul}{\longleftarrow} & ud(V,lk(\lambda w.X_2[w,w]))
\end{array}
$$



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 β-rule in the pure λ-calculus.

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

6.1 Ralationship 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 β-reduction is strongly normalizing as well as η-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 $u{\updownarrow^\eta_\beta}$, the β-normal form, the η-expanded form and the β-normal η-expanded form, respectively, possibly omitting indices and exponents when convenient.

A rule "à la Nipkow" assumes η-expanded left-hand side patterns and η-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 $(X\ \overline{x})$ in a Nipkow's pattern will correspond in our syntax to the pre-redex $X[\overline{x}]$ in which $ar(X) = |\overline{x}|$. 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 η-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 will therefore depend whether the rule $L \longrightarrow R$ 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 6.3. $u \xrightarrow[L\to R]{p} v$ for $L \longrightarrow R \in \mathcal{R}_{nw}$ iff $u = u{\updownarrow}$, $u|_p =_{\beta^0} L\sigma$ for some β-normal η-expanded substitution σ and $v = u[R\sigma]_p{\updownarrow}$.

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

Lemma 6.4. *If* $u \underset{\mathcal{R}_{nw}}{\overset{p}{\longrightarrow}} v$ *then* $u = u{\updownarrow}$ *and* $u \underset{\mathcal{R}_{kp}}{\overset{p}{\longrightarrow}} \underset{\beta}{\longrightarrow\!\!\!\!\!\rightarrow} v = v{\updownarrow}$.

Lemma 6.5. *For all* $u \in \mathcal{T}$ *such that* $u \underset{\mathcal{R}_{kp}}{\overset{p}{\longrightarrow}} v$ *and* $v \in \mathcal{T}$ *then* $u{\updownarrow} \underset{\mathcal{R}_{nw}}{\overset{*}{\longrightarrow}} v{\updownarrow}$.

Proof. We have $u = u[L\sigma]_p = \overline{u}^p\{Z \mapsto \lambda\overline{z}.L\sigma\} = \overline{u}^p\gamma$ and $v = u[R\sigma]_p = \overline{u}^p\{Z \mapsto \lambda\overline{z}.R\sigma\} = \overline{u}^p\theta$ with $\gamma \underset{\mathcal{R}}{\longrightarrow} \theta$. Since L is a pattern $(L\sigma){\downarrow} = L(\sigma{\downarrow})$ and since it cannot be an abstraction we have, $u{\updownarrow} = ((\overline{u}^p\gamma){\downarrow}){\uparrow} = (\overline{u}^p{\downarrow}\,(\gamma{\downarrow})){\uparrow} \underset{\mathcal{R}_{nw}}{\overset{*}{\longrightarrow}} (\overline{u}^p{\downarrow}\,\{Z \mapsto \lambda\overline{z}.R(\sigma{\downarrow})\}){\updownarrow} = ((\overline{u}^p\{Z \mapsto \lambda\overline{z}.R\sigma\}){\downarrow}){\uparrow} = v{\updownarrow}$ □

Corollary 6.6. *If* $u \underset{\beta\eta\mathcal{R}_{kp}}{\longrightarrow\!\!\!\!\!\rightarrow} v$ *then* $u{\updownarrow} \underset{\mathcal{R}_{nw}}{\overset{*}{\longrightarrow}} v{\updownarrow}$.

Lemma 6.7. *Assume a rewrite system* $\mathcal{R}$ *such that* $\underset{\beta}{\longrightarrow} \cup \underset{\mathcal{R}_{kp}}{\longrightarrow}$ *is Church-Rosser. Then* $\underset{\mathcal{R}_{nw}}{\longrightarrow}$ *is Church-Rosser on* β*-normal* η*-expanded terms of* $\mathcal{T}$.

Proof. Assume $u \underset{\mathcal{R}_{nw}}{\overset{*}{\longleftarrow}} v$. Then by Lemma 6.5, $u = u{\updownarrow}$, $v = v{\updownarrow}$ and $u \underset{\beta\eta\mathcal{R}_{kp}}{\longleftarrow\!\!\!\!\!\rightarrow} v$. By assumption, $u\ (\underset{\mathcal{R}_{kp}}{\longrightarrow} \cup \underset{\beta}{\longrightarrow})^*(\underset{\mathcal{R}_{kp}}{\longleftarrow} \cup \underset{\beta}{\longleftarrow})^*\ v$ and by Corollary 6.6, $u \underset{\mathcal{R}_{nw}}{\overset{*}{\longrightarrow}}\ \underset{\mathcal{R}_{nw}}{\overset{*}{\longleftarrow}} v$. □

Theorem 6.8. *Assume* $\mathcal{R}$ *is a left-linear system such that all critical pairs are joinable using* $\underset{\mathcal{R}_{kp}}{\longrightarrow}$ *and* $\mathcal{R}$ *is strongly normalizing. Then* $\underset{\mathcal{R}_{nw}}{\longrightarrow}$ *is Church-Rosser on* β*-normal* η*-expanded terms of* $\mathcal{T}$.

Proof. $\underset{\mathcal{R}}{\longrightarrow}$ satisfies the hypotheses of Theorem 6.1. We conclude that $\underset{\beta}{\longrightarrow} \cup \underset{\mathcal{R}}{\longrightarrow}$ is Church-Rosser and by Lemma 6.7, $\underset{\mathcal{R}_{nw}}{\longrightarrow}$ is too on β-normal η-expanded terms of $\mathcal{T}$. □

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

7 CONCLUSION

Confluence of first-order rewrite rules is well understood, in both the terminating and non-terminating cases. Confluence of left-linear higher-order rules on simply-typed λ-terms is well understood too [12]. This is true as well of confluence of first-order rules in presence of β-reductions for any type discipline for which β is terminating [4].

In this paper, we have described a condition on critical pairs which ensures preservation of confluence in the untyped λ-calculus by a set $\mathcal{R}$ll of rewrite rules whose left-hand sides are left-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 β-rule itself, on the other hand, cannot be used to join the critical pairs. In that case, confluence can be obtained by joining *nested critical pairs*, as will be shown in the second paper of this series. This other result does not subsume the present one, since nested critical pairs may be infinitely many, as is the case with the theory of global states for a unique location [6], which is shown here to preserve confluence of β-reductions by computing its finitely many critical pairs.

In order to define critical pairs, we had to unify left-linear patterns, where patterns are specific untyped λ-terms whose definition ensures that erasing types from a simply-typed pattern in Miller's sense yields a pattern in our sense. Untyped patterns enjoy most general unifiers, in the same way as Miller's patterns do. Note that unification here looks purely syntactic, thanks to a definition of substitution which incorporates β-reductions. The linearity restriction should not be essential:

having multiple occurrences of meta-variables requires using a merge rule in the unification case, and checking terms for equality in the matching case.

One may wonder why we did not consider a well-known setting, like Klop's combinatory reduction systems [11] or Nipkow's higher-order rewriting, or van Oostrom's higher-order rewriting systems [20], and then encode our notion of higher-order rewriting within their's. One main reason is that we always insist, in Dedukti, in using shallow encodings, hence do not want to encode the λ-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 unsual. Although one could fear that the present setting becomes too specific for a wide application, we believe that this is not the case, and that it can be used to show confluence of rewrite rules in other dependent type theories without difficulty, as well as for other, related rewrite relations, as we have shown with Nipkow's higher-order rewriting.

One may also wonder whether considering parallel higher-order critical pairs could improve our results. The difficulty here is that one of the decreasing diagrams for free, Lemma 5.4, breaks down. It can of course be repaired, to the price of imposing that meta-variables do not occur embedded in one another in the right-hand sides of the rules. This restriction looks of course very strong. However, any expression such as $X[Y]$ can be transformed into $(X\ Y)$, hence eliminating this embedding. There is of course a general transformation that will eliminate all embeddings, making the use of parallel rewriting (and therefore parallel critical pairs) look attractive. The problem however, is that right-hand sides such as $(X\ Y)$ may result in the use of β-steps to join the critical pairs, hence the joinability diagrams would not be decreasing. This may or may not happen, of course. It is certainly possible to exhibit examples for which this transformation would work. We have not encountered such a natural example so far. A forthcoming paper will therefore adress directly an even more general left-linear case, by using orthogonal higher-order rewriting.

The case of non-left-linear rules is not touched at all here, it is indeed much more difficult since adding such rules to the untyped λ-calculus results, in general, in loosing confluence, as shown by Klop [11]. We however show in another forthcoming paper that for all Klop's counter-examples, confluence is preserved on appropriate subsets of λ-terms, hence showing a way to get around this difficulty. Finally, mixing left-linear rules with right-linear ones, is a problem which is important to us, because encodings of complex type theories in Dedukti are not purely left-linear, and of course not purely right-linear either. We do not know yet whether we can obtain meaningful results for this combination.

Acknowledgments: to Gilles Dowek for many discussions, Jiaxiang Liu for a chary reading, and Vincent van Oostrom for his many suggestions and corrections to an earlier draft.

REFERENCES

[1] Ali Assaf, Guillaume Burel, RaphaÃńl Cauderlier, Gilles Dowek, Catherine Dubois, FrÃĺdÃĺric Gilbert, Pierre Halmagrand, Olivier Hermant, and Ronan Saillard. Dedukti: a Logical Framework based on the lambda-pi-Calculus Modulo Theory. draft, INRIA, 2019.

[2] Hendrik Pieter Barendregt. *The lambda calculus : its syntax and semantics.* Studies in logic and the foundations of mathematics. North-Holland, Amsterdam, New-York, Oxford, 1981.

[3] Jesper Cockx, Nicolas Tabareau, and Théo Winterhalter. How to tame your rewrite rules, 2018. Draft.

[4] Daniel J. Dougherty. Adding algebraic rewriting to the untyped lambda calculus. *Inf. Comput.*, 101(2):251–267, 1992.

[5] Gilles Dowek at all. The Dedukti system, 2016. Available from http://dedukti.gforge.inria.fr/.

[6] Gilles Dowek, Jean-Pierre Jouannaud and Jiaxiang Liu. Confluence in untyped higher-order theories. draft hal-, INRIA, january 2019. Full version of a work presented at HOR 2016.

[7] Healfdene Goguen. The metatheory of UTT. In Peter Dybjer, Bengt Nordström, and Jan M. Smith, editors, *Types for Proofs and Programs, International Workshop TYPES'94, Båstad, Sweden, June 6-10, 1994, Selected Papers*, volume 996 of *Lecture Notes in Computer Science*, pages 60–82. Springer, 1994.

[8] Makoto Hamana. How to prove your calculus is decidable: practical applications of second-order algebraic theories and computation. *PACMPL*, 1(ICFP):22:1–22:28, 2017.
[9] J. R. Hindley. An abstract form of the Church-Rosser theorem. i. *J. Symb. Log.*, 34(4):545–560, 1969.
[10] Jean-Pierre Jouannaud and Jiaxiang Liu. From diagrammatic confluence to modularity. *Theor. Comput. Sci.*, 464:20–34, 2012.
[11] Jan Willem Klop. *Combinatory reduction systems*. PhD thesis, CWI tracts, 1980.
[12] Richard Mayr and Tobias Nipkow. Higher-order rewrite systems and their confluence. *Theoretical Computer Science*, 192:3–29, 1998.
[13] Dale Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. *Journal of Logic and Computation*, 1(4):497–536, 1991.
[14] Maxwell H. A. Newman. On theories with a combinatorial definition of 'equivalence'. *Ann. Math.*, 43(2):223–243, 1942.
[15] Mitsuhiro Okada. Strong normalizability for the combined system of the typed lambda calculus and an arbitrary convergent term rewrite system. In Gaston H. Gonnet, editor, *Proceedings of the ACM-SIGSAM 1989 International Symposium on Symbolic and Algebraic Computation, ISSAC '89, Portland, Oregon, USA, July 17-19, 1989*, pages 357–363. ACM, 1989.
[16] Gordon D. Plotkin and John Power. Algebraic operations and generic effects. *Applied Categorical Structures*, 11(1):69–94, 2003.
[17] Vincent van Oostrom. Confluence by decreasing diagrams. *Theor. Comput. Sci.*, 126(2):259–280, 1994.
[18] Vincent van Oostrom. Developing developments. *Theor. Comput. Sci.*, 175(1):159–181, 1997.
[19] Vincent van Oostrom. Confluence by decreasing diagrams converted. In Voronkov A., editor, *RTA*, volume 5117 of *Lecture Notes in Computer Science*, pages 306–320. Springer, 2008.
[20] Vincent van Oostrom and Femke van Raamsdonk. Comparing combinatory reduction systems and higher-order 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, Selected Papers*, volume 816 of *Lecture Notes in Computer Science*, pages 276–304. Springer, 1993.

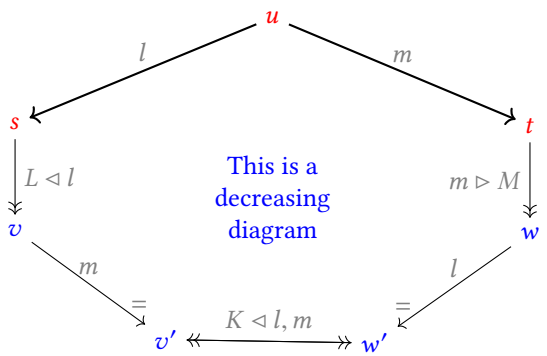


Fig. 2. Decreasing diagram

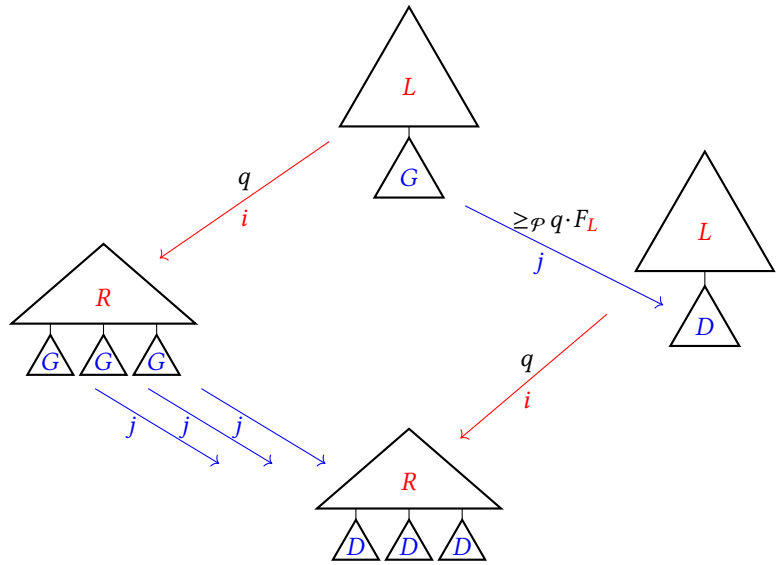


Fig. 3. Ancestor peaks in rewrite theories. L, G stand for terms rewriting to R, D, using a red rule in $\mathcal{R}$ll and a blue rule in $\mathcal{R}$ll $\cup \beta$.

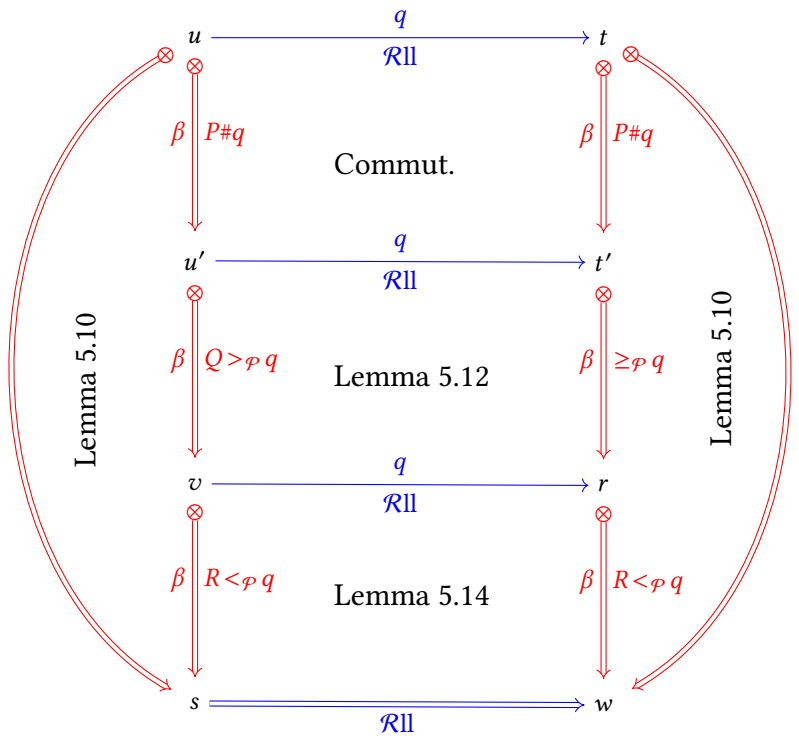


Fig. 4. Construction of a decreasing diagram for heterogeneous local peaks.

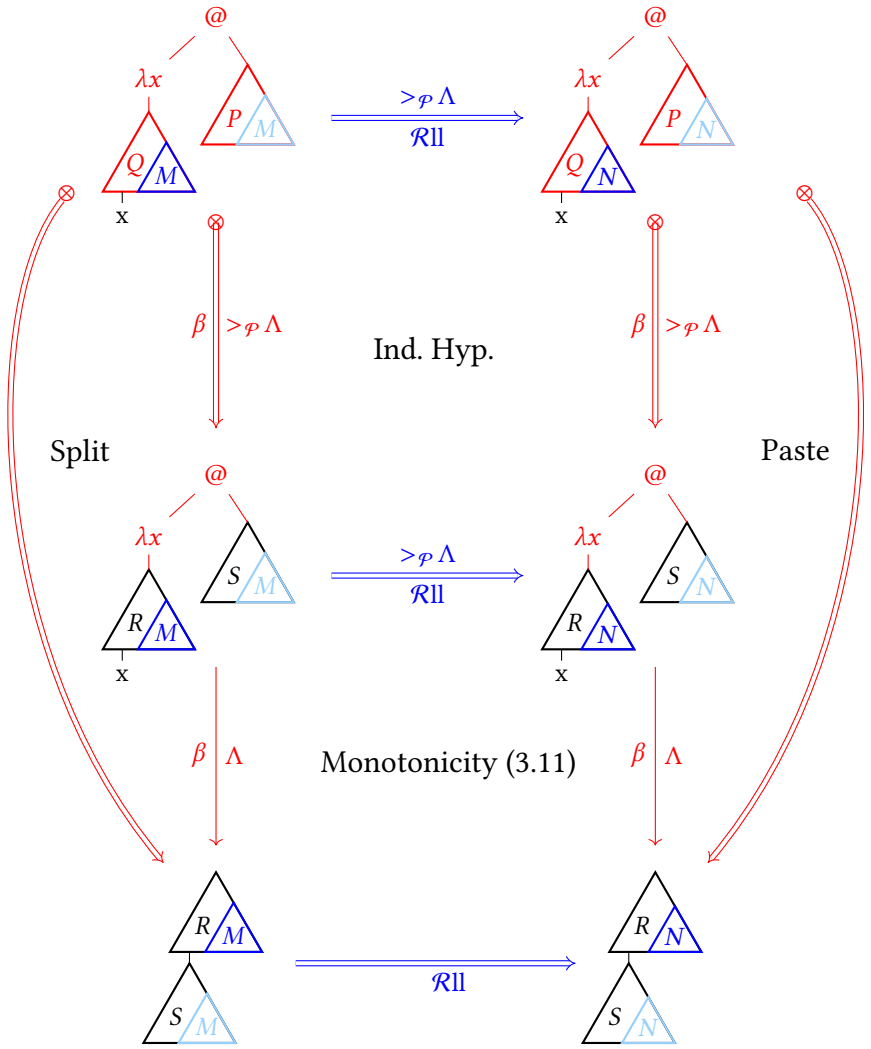


Fig. 5. Construction of a decreasing diagram for peak: Lemma 5.14

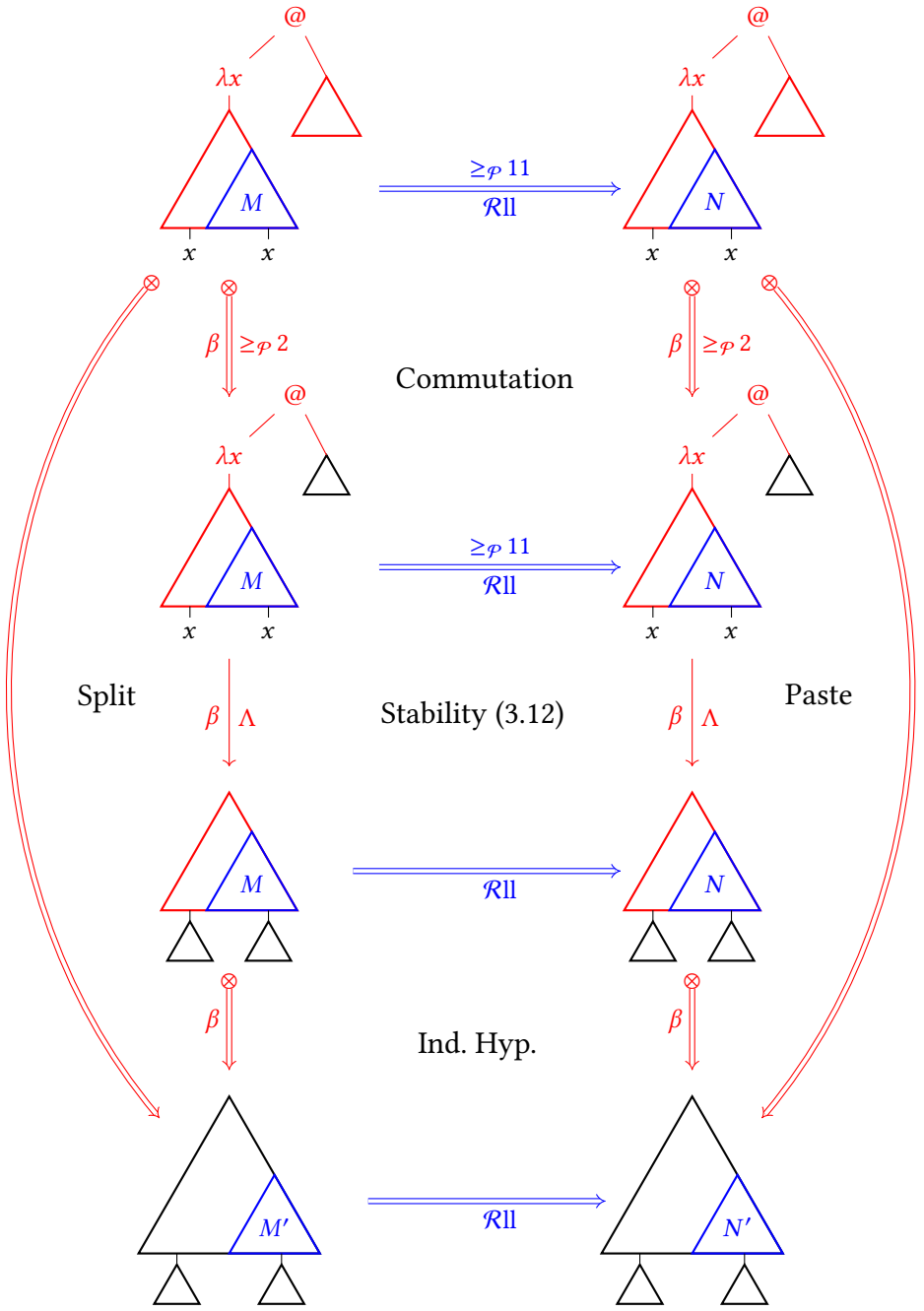


Fig. 6. Construction of a decreasing diagram for peak: Lemma 5.14