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

Gaspard Férey, Jean-Pierre Jouannaud

▶ **To cite this version:**

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

HAL Id: hal-03126102

https://inria.hal.science/hal-03126102v1

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

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

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

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

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

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

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

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

1 INTRODUCTION

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

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

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 $\beta \cup \mathcal{R}$. Since our prime target is Dedukti, which is a calculus without extentionality, we do not consider here the η rule in addition to the β-rule.

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

There are three main tools for analyzing confluence of a rewrite relation: Newman's Lemma [19], Hindley-Rosen's Lemma [12], and van Oostrom's Theorem which generalizes both previous ones [26]. 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 pairs [25] is too restrictive for practical usage. The way out is the use of van Oostrom's decreasing diagrams [24]. 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 [22]. Numerous counter-examples to confluence in the pure λ-calculus are given in [15]. 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 [17, 18], which are extremely useful for describing encodings of a type theory in another, a keen application to us. Considering untyped terms requires some 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 bounded. 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 main contribution is that a set $\mathcal{R}$ 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 $\mathcal{R} \cup M\eta$, where $M\eta$ is the extensionality rule for the meta-variables.

This result is then demonstrated with the example of a theory of global states due to Plotkin and Power [23], whose rules have overlapping linear higher-order patterns as left-hand sides. The confluence of this example had indeed been shown already in [11]. 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 method applies independently of the typing system, hence we can deduce that the same example remains confluent when using a dependently typed discipline. As we shall see, the critical pairs are not development closed, hence this result does not follow from existing techniques.

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

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 are considered in Section 4. Local rewriting peaks are analyzed in Section 5. Our confluence result is stated and proved in Section 6. Related work and applicability of our results is discussed in Section 7. Concluding remarks come last.

2 LABELED REDUCTIONS

2.1 Reductions

Given a binary relation $\longrightarrow$ on terms, called *rewriting*, we use: $\longleftarrow$ for its inverse, $\Longrightarrow$ for its parallelization, allowing one to rewrite at once several subterms of a given term, when none is a subterm of another, and $\longleftrightarrow$, $\longrightarrow\!\!\!\!\!\rightarrow$, 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* (or *strong normalization*) is the impossibility of an infinite rewriting sequence $t_0 \longrightarrow t_1 \longrightarrow \ldots \longrightarrow t_n \longrightarrow \ldots$ and guarantees the existence of normal forms for every term. A *local peak* is a triple of terms (s, u, t) such that $s \longleftarrow u \longrightarrow t$; u is the *source* and s, t are its *reducts*. Two terms s, t are *joinable* if $s \longrightarrow\!\!\!\!\!\rightarrow v \twoheadleftarrow\!\!\!\!\!\leftarrow t$ 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 [24]). Given a labeled relation $\longrightarrow$ on an abstract set, we denote by $DS(m, n)$ the set of *decreasing rewrite sequences* of the form $u \xrightarrow{\delta}\!\!\!\!\!\rightarrow v$ or $u \xrightarrow{\gamma}\!\!\!\!\!\rightarrow s \xrightarrow{n} t \xrightarrow{\delta}\!\!\!\!\!\rightarrow v$ 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 $v \xleftarrow{m} u \xrightarrow{n} w$, a *decreasing (rewrite) diagram* is a pair of derivations from v and w to some common term t, belonging to $DS(m, n)$ and $DS(n, m)$, respectively.

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

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

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

3 HIGHER-ORDER REWRITING

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

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

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

$\lambda\mathcal{F}$ is a mix of the pure lambda-calculus and Klop's combinatory reduction systems [15], that fits with Dedukti [8]. Terms are those of the pure lambda calculus enriched with $\mathcal{F}$*-headed* terms of the form $f(\overline{u})$ 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 $|f|$ or $|Z|$. Variables have arity zero. The grammar of terms is the following:

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

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 $u_m, \ldots u_n$. Commas in lists are sometimes omitted.

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

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

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*), $<_{\mathcal{P}}$ and $>_{\mathcal{P}}$ for their strict parts, and $p\#q$ for $(p \not\leq_{\mathcal{P}} q \wedge p \not\leq_{\mathcal{P}} q)$ (*parallel*). An order $>$ on positions is extended to sets of positions as follows: $P > Q$ iff $\forall p \in P\ \exists q \in Q$ such that $p > q$.

Given a term M, we use $\mathcal{P}os(M)$, $\mathcal{VP}os(M)$, $\mathcal{MP}os(M)$ for its sets of: all positions, positions of free variables, positions of meta-variables, respectively; and $\mathcal{V}ar(M)$, $\mathcal{BV}ar(M)$ and $\mathcal{MV}ar(M)$ for its sets of: free variables, bound variables and meta-variables, respectively. A term M is *ground* if $\mathcal{V}ar(M) = \emptyset$, *closed* if $\mathcal{MV}ar(M) = \emptyset$, 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 position p, $M|_p$ for the subterm of M at position p, $M[N]_p$ for the term obtained by replacing in M the subterm $M|_p$ by the term N. The latter notation extends to sets P of parallel positions in $M[\overline{N}]_P$ 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. We say that a variable $x \in \mathcal{BV}ar(M)$ is *bound above* p in M if $M|_q = \lambda x.N$ for some $q <_{\mathcal{P}} p$.

3.2 Substitutions

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

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

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

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

The result $t\sigma$ of substituting the term t is called an *instance* (of t) and the operation itself an *instantiation*. The substitution 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.

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

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

Our definition of substitution ensures the following important property:

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

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

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

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

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

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

Given a term u and a list $P = \{p_i\}_{i=1}^{i=n}$ of parallel positions in u, we define the term obtained by *splitting* u along P as $\underline{u}_P = u[Z_1[\overline{x_1}]]_{p_1} \dots [Z_n[\overline{x_n}]]_{p_n}$ (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. We do not claim for originality here, although we cannot tell who made use of it first.

3.3 Functional reductions

Arrow signs used for rewriting will often be decorated, below by a name, and above by a position p or set of positions P, as in $s \xrightarrow[R]{p} t$ or by a property that this position or set of positions satisfies, as in $u \xrightarrow[R]{\geq_{\mathcal{P}} p} v$ and in $v = u{\downarrow}_R^{\geq_{\mathcal{P}} P}$ (v is obtained from u by normalizing its subterms $v|_{p \in P}$ with $\mathcal{R}$).

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

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

The particular case for which v is a variable is denoted by β^0, a notation introduced by Miller. 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 another important property: $M\eta$-steps disappear when taking ground instantiations, a key property for us.

Lemma 3.3. *Let $\overline{u}$ be a sequence of closed terms, z a variable such that $z \notin \mathcal{V}ar(\overline{u})$, X a meta-variable such that $|X| > |\overline{u}|$ and σ a closed substitution. Then $\lambda z.X[\overline{u}, z]\sigma = X[\overline{u}]\sigma$.*

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

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

3.4 Higher-order reductions

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

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

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

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

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

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

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

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

We can now define higher-order rules and rewriting:

Definition 3.8 (Rule). A (higher-order) *rule* is a triple $i : L \to R$, 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 splittings belong to the meta-language.

Definition 3.9 (Higher-order untyped rewriting). Given a 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 $u \xrightarrow[i]{p} v$, or $u \xrightarrow[L\to R]{p} v$, iff $u|_p = L\gamma$ for some substitution γ, and $v = u[R\gamma]_p$. 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 $\overline{x}$ the variables bound above p in u and X a fresh meta-variable of arity $|\overline{x}|$.
- $u = \underline{u}_p\overline{u}^p = \underline{u}_p\{X \mapsto \lambda\overline{x}.u|_p\} = \underline{u}_p\{X \mapsto \lambda\overline{x}.L\gamma\}$
- $v = \underline{u}_p\{X \mapsto \lambda\overline{x}.R\gamma\}$, hence $\underline{v}_p = \underline{u}_p$, $\overline{v}^p = \{X \mapsto \lambda\overline{x}.R\gamma\}$ and $v|_p = R\gamma$.

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

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

Besides, we do *not* assume that u, or v, is β-normal, or even β-normal up to position p. Two reasons prevent it, β-normal forms may not exist, and we need monotonicity and stability properties:

Lemma 3.11 (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 3.12 (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 $u[s]_q|_{q\cdot p} = s|_p$. Monotonicity extends to several different rewrites under a same context $u[\ ]_Q$, where Q is a set of parallel positions.

Stability is just as easy.

Lemma 3.13 (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 3.14 (Substitution Lemma). *Let* $u \xrightarrow[\mathcal{R}]{} v$ *and* $\sigma \xrightarrow[\mathcal{R}]{} \tau$. *Then,* $u\sigma \xrightarrow[\mathcal{R}]{}\!\!\!\!\!\rightarrow\ v\tau$.

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

- $u = f(\overline{u})$ with $f \in \mathcal{F}$. By induction hypothesis, $\overline{u}\sigma \longrightarrow\!\!\!\!\!\rightarrow\ \overline{u}\tau$, by monotonicity, $f(\overline{u}\sigma) \longrightarrow\!\!\!\!\!\rightarrow\ f(\overline{u}\tau)$.
- $u = (v\ w)$ and $u = \lambda x.v$ are similar to the first case.
- $u = x$ is straightforward.
- $u = X[\overline{u}]$ with $X \notin \mathcal{D}om(\sigma)$. Similar to the first case.
- $u = X[\overline{u}]$, $\lambda\overline{xy}.u' \in \sigma$, $\lambda\overline{xy}.v' \in \tau$, $|\overline{x}| = |\overline{u}|$, and $u' \longrightarrow v'$.
  Then $X[\overline{u}]\sigma = \lambda\overline{y}.u'\{\overline{x} \mapsto \overline{u\sigma}\}$ and $X[\overline{u}]\tau = \lambda\overline{y}.v'\{\overline{x} \mapsto \overline{u\tau}\}$. By induction hypothesis, $\overline{u\sigma} \longrightarrow\!\!\!\!\!\rightarrow \overline{u\tau}$. By monotonicity, $u\sigma = \lambda\overline{y}.u'\{\overline{x} \mapsto \overline{u\sigma}\} \longrightarrow\!\!\!\!\!\rightarrow \lambda\overline{y}.u'\{\overline{x} \mapsto \overline{u\tau}\} = u\tau$.
- $u = X[\overline{vw}]$, $\lambda\overline{x}.Y[\overline{s}] \in \sigma$ and $\lambda\overline{x}.Y[\overline{t}] \in \tau$, with $|\overline{x}| = |\overline{v}|$.
  Then $u\sigma = (Y[\overline{s}\{\overline{x} \mapsto \overline{v\sigma}\}\ \overline{w\sigma})$ and $u\tau = (Y[\overline{s}\{\overline{x} \mapsto \overline{v\tau}\}\ \overline{w\tau})$. By induction hypothesis, $\overline{v\sigma} \longrightarrow\!\!\!\!\!\rightarrow \overline{v\tau}$ and $\overline{w\sigma} \longrightarrow\!\!\!\!\!\rightarrow \overline{w\tau}$. Hence $u\sigma \longrightarrow\!\!\!\!\!\rightarrow u\tau$ by monotonicity again.

Since $u\tau \longrightarrow\!\!\!\!\!\rightarrow v\tau$ by stability, we conclude that $u\sigma \longrightarrow\!\!\!\!\!\rightarrow v\tau$ by transitivity of rewriting. □

3.5 Rewrite theories

Definition 3.15. 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 $\xrightarrow[\mathcal{R}\cup\beta_\alpha]{}$. We say that $\lambda\mathcal{F}$ is a *left-linear* theory if $\mathcal{R}$ is a set of left-linear rewrite rules.

Rewrite theories are used in Dedukti [1] to define the conversion rule of the calculus, which, as is customary, is 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. Showing this meta-theoretical property of $\lambda\mathcal{F}$ will involve rewriting arbitrary terms, and possibly use $M\eta$-rewriting steps for its checking. In other words, confluence of $\mathcal{R} \cup \beta$ will not be satisfied on the whole set of terms, but only on the subset of closed terms. We don't need more, of course.

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

3.6 The rewrite theory of global states

Our running example here will be Plotkin's and Power's theory of global states for a single location [23]. It is described by two types, $\mathtt{Val}$ for values and $\mathtt{A}$ for states, a unary operation $\mathtt{lk}$ for looking up a state, a binary operation $\mathtt{ud}$ 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:

$$\mathtt{lk} \;:\; (\mathtt{Val} \to \mathtt{A}) \to \mathtt{A} \quad \Big| \quad \mathtt{ud} \;:\; \mathtt{Val}, \mathtt{A} \to \mathtt{A}$$

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

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

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

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

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

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

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

4.1 Matching and unification problems

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

Definition 4.1. A (matching or unification) *equational problem* 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 and v is either a pre-pattern (unification case), or an arbitrary term (matching case).

Given the problem $P = \bigwedge_i u_i = v_i$, we denote by $l\mathcal{MV}ar(P)$ it set of left meta-variables $\bigcup_i \mathcal{MV}ar(u_i)$, and by $r\mathcal{V}ar(P)$ and $r\mathcal{MV}ar(P)$ its respective sets of right variables $\bigcup_i \mathcal{V}ar(v_i)$ and meta-variables $\bigcup_i \mathcal{MV}ar(v_i)$.

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

Definition 4.2 (Solutions and unifiers). Let P be the equational problem $\bigwedge_i u_i = v_i$.
A substitution γ is a *matching solution* of P if $\mathcal{D}om(\gamma) = l\mathcal{MV}ar(P)$ and $\forall i\ (u_i\gamma =_\alpha v_i)$.
A ground substitution γ is a *unifier* (or *unifying substitution*) of P if $\forall i\ u_i\gamma =_\alpha v_i\gamma$.
A ground closed unifier of domain $\mathcal{MV}ar(P)$ is called a *unifying solution* of P.
The equational problems containing $\bot$ have no matching solution nor unifier.

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

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

Lemma 4.3. *If a unification problem P has a unifier γ, then it has a more general unifier σ such that $\mathcal{D}om(\sigma) \subseteq \mathcal{MV}ar(P)$.*

Proof. *By induction on the number n of meta-variables in $\mathcal{D}om(\gamma) \setminus \mathcal{MV}ar(P)$.*

- *Base case: $\mathcal{D}om(\gamma) \subseteq \mathcal{MV}ar(P)$, then $\sigma = \gamma$.*
- *Induction step: let X be a meta-variable such that $X \mapsto s \in \gamma$ and $X \notin \mathcal{MV}ar(P)$. Let τ be the substitution of domain $\mathcal{D}om(\gamma) \setminus X$ such that for all $Y \in \mathcal{D}om(\tau)$, $Y \mapsto \lambda\overline{y}.t\{X \mapsto X'\} \in \tau$ iff $Y \mapsto \lambda\overline{y}.t \in \gamma$. Then, $\gamma = \tau\{X' \mapsto X, X \mapsto s\}$, hence τ is more general than γ. Now, $u\tau = u\gamma\{X \mapsto X'\} = v\gamma\{X \mapsto X'\} = v\tau$, hence τ is a unifier of P. By induction hypothesis, there exists a unifier σ smaller than τ, hence smaller than γ by transitivity of subsumption, which satisfies our requirements.*

4.2 Matching and unification rules

Definition 4.4. A unification problem $\mathcal{P}$ is *linear* if no meta-variable occurs more than once in $\mathcal{P}$. A matching problem $\mathcal{P}$ is *linear* if 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. This is enough for our use of unification to solving equations between left-hand sides of rules, since we restrict ourselves to left-linear rewrite rules.

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

Definition 4.5. A free 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. We denote by $\mathcal{UV}ar(u)$ the set of unprotected variables of u.

Dec-Fun	$f(\overline{u}) = f(\overline{v}) \longrightarrow \wedge_{i=1}^{i=\lvert f\rvert} u_i = v_i$	if $f \in \mathcal{F} \cup \mathcal{X} \cup \{@\}$
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 $f, g \in \mathcal{F} \cup \mathcal{X} \cup \{@, \lambda\}$ and $f \neq g$
Meta-Abs	$X[\overline{x}] = \lambda y.v \longrightarrow X[\overline{x}y] = v$	if $X \in \mathcal{Z}$ and $\lvert X\rvert > \lvert\overline{x}\rvert$
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 $\exists z \in \mathcal{V}ar(u)$ s.t. $z \notin (\overline{x} \cup r\mathcal{V}ar(P_0))$
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 $\exists z \in \mathcal{UV}ar(u)$ st $z \notin \overline{x}$
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 $\begin{cases} \overline{y} \notin \overline{x} \cup \mathcal{BV}ar(u) \\ \lvert X\rvert = \lvert\overline{x}\rvert \text{ if } u(\wedge) \in \mathcal{F} \cup \{@, \lambda\} \\ \mathcal{UV}ar(u) \subseteq \overline{x} \\ \lvert Y\rvert - \lvert\overline{y}\rvert \geq \lvert X\rvert - \lvert\overline{x}\rvert \text{ if } q = \wedge \end{cases}$ 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

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

Pattern matching and unification are described by the rewrite rules given in Figure 1. Rules written in black apply to both matching and unification problems. Rules written in green apply to matching problems only, while rules in blue apply to unification problems only. Note that the constant $\bot$ 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-Abs*, the first subset of four black rules treats equations between expressions which are not pre-redexes. Those equations can be decomposed or fail, depending on the respective root symbol of the left-hand and right-hand sides. These rules are just the same as those for first-order unification, with the exception, of course, of *Dec-Abs*. Note that *Dec-Fun* removes equations of the form $x = x$, and that we abuse our notations in *Conflict* by assimilating λ with a symbol. The rôle of *Meta-Abs* is to ensure that the arity condition for meta-variables is met by the substitution that will be obtained if the algorithm succeeds.

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

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

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

In the particular case where $q = \wedge$, $|X|-|\overline{x}| = |Y|-|\overline{y}|$ 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*.

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

4.3 Examples

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

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

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

(where Z is a fresh variable of arity 1)

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

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

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

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

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

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

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

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

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

(where Z is a fresh variable of arity 1)

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

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

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

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

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

(where Z is a fresh variable of arity 0)

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

$$\mathtt{lk}(\lambda w.\mathtt{lk}(Y[w])) = \mathtt{lk}(\lambda w.U) \underset{Dec-Fun}{\longrightarrow} \lambda w.\mathtt{lk}(Y[w]) = \lambda w.U \underset{Dec-Abs}{\longrightarrow} \mathtt{lk}(Y[w]) = U \underset{Swap}{\longrightarrow} U = \mathtt{lk}(Y[w])$$
$$\underset{Drop}{\longrightarrow} U = \mathtt{lk}(Z) \wedge Y[w] = Z$$

(where Z is a fresh variable of arity 0)

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

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

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

4.4 Soundness

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

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

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

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

Definition 4.10. Given two problems P and P_0, a matching solution γ of P is *adequate* for P_0 if $\mathcal{D}om(\gamma) \subseteq \mathcal{MV}ar(P_0)$ and $\mathcal{R}an(\gamma) \subseteq r\mathcal{MV}ar(P_0) \cup r\mathcal{V}ar(P_0)$.

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

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

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

Lemma 4.12. Dec-Fun *and* Conflict *preserve all solutions.*

Lemma 4.13. Dec-Abs *is sound.*

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

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

Lemma 4.14. Meta-Abs *preserves all solutions.*

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

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

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

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

Proof. By non-decreasingness Lemma 3.2, $ar(X[\overline{x}]\sigma) \geq ar(X[\overline{x}]) > ar(u)$. □

Lemma 4.16. *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 two failure rules.

Lemma 4.17. Fail-Protect *is sound.*

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

Lemma 4.18. *The elementary unification problem $X[\overline{x}] = u$ has no solution if $z \in \mathcal{UV}ar(u) \setminus \overline{x}$.*

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

Lemma 4.19. *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. Assume first that γ is a unifier of P. Then, $X[\overline{x}]\gamma = u[Z[\overline{z}]_q\gamma = u\gamma[Z[\overline{z}]\gamma]_q$ by definition of a substitution and the fact that $u[Z[\overline{z}]_q$ is a pre-pattern. Since $Z[\overline{z}]\gamma = Y[\overline{y}]\gamma$, it follows that $X[\overline{x}]\gamma = u[Z[\overline{z}]_q\gamma = u[Y[\overline{y}]_q\gamma = u[Y[\overline{y}]_q\gamma$, hence $\gamma_{\setminus Z}$ unifies E.

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

We can now conclude:

Lemma 4.20. *The matching/unification rules are terminating and sound.*

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

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

4.5 Solved forms

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

Definition 4.21 (Solved forms).

(1) An equation $u = v$ is in *arity-solved form* if u is a pre-redex $X[\overline{x}|$ such that $ar(v) \geq |X| - |\overline{x}|$;
(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 4.22.

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

Proof. (1) The only if case being clear, let us assume that no rule other than *Fail-Protect*, *Fail-Protect*, *Drop* and *Meta-Abs* can apply to $u = v$. Necessarily $u = X[\overline{x}]$ 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* does no apply by assumption, we conclude that $|X| - |\overline{x}| \leq |Y| - |\overline{y}|$, hence $u = v$ is in arity-solved form.

(2) The case of *Meta-Abs* follows from the definition of arity. Consider now *Drop*. If $q = \wedge$, then $|X| - |\overline{x}| \leq |Y| - |\overline{y}| = |Z| - |\overline{z}|$ 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 4.23. *All rules preserve the following two properties of an unification problem P:*

(1) For all equations $u = v \in P$ not in arity-solved form, all meta-variables in $\mathcal{MV}ar(u) \cup \mathcal{MV}ar(v)$ are linear in P.

(2) For all equations $X[\overline{x}] = v \in P$ in arity-solved form, the meta-variable X is linear in P and for all pre-redexes $Z[\overline{z}]$ in v, either the meta-variable Z is linear in P or $\overline{z} \subseteq \mathcal{V}ar(v) \cup \overline{x}$.

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

Proof. Let Q be the normal form of P. By Lemma 4.22 (1), all equations in Q are in arity-solved form. By Lemma 4.23 (2), pre-redexes in their left-hand sides are linear in Q. We are left proving that for all $X[\overline{x}] = v \in Q$, $\mathcal{V}ar(v) \subseteq \overline{x}$. 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 4.25. *Solved forms of equational problems are computed in linear time.*

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

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

4.6 Matching solutions and unifiers

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

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

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

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

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

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

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

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

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

Proof. Let σ be $\mathrm{mgu}(P)$. We need to show that σ is a unifier, that it is most general, and that it is unique up to equivalence.

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

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

Finally, let τ be another most general unifier. We assume $\mathcal{D}om(\tau) \subseteq l\mathcal{MV}ar(P)$ and prove $\sigma =_{\alpha\cup M\eta} \tau$ from $X_i[\overline{x_i}]\sigma = X_i[\overline{x_i}]\tau$ as in the proof of Lemma 4.26. □

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

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

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

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

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

5 LOCAL PEAKS IN REWRITE THEORIES

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

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

5.1 Decreasing diagrams for free

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

Lemma 5.1. *Given terms s, t such that $s \underset{i:L\to R}{\overset{p}{\longleftarrow}} u \underset{j:G\to D}{\overset{q}{\longrightarrow}} t$, there are three possibilities: (i) $p\#q$ (disjoint peak case); (ii) $q \geq_{\mathcal{P}} p \cdot F_L$ or $p \geq_{\mathcal{P}} q \cdot F_G$ (ancestor peak case); and (iii) $p = q \cdot o$ with $o \in \mathcal{FP}os(G)$ or $q = p \cdot o$ with $o \in \mathcal{FP}os(L)$ (overlapping peak case).*

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

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

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 $s \underset{j}{\overset{Q}{\Longrightarrow}} \underset{\beta}{\overset{p}{\longleftarrow}} t$ for some set Q of parallel positions of s such that $Q \geq_{\mathcal{P}} p$.*

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

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

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

The case of a local peak $s \underset{i}{\overset{p}{\longleftarrow}} u \underset{j}{\overset{q}{\longrightarrow}} t$, 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 $u \underset{i:L\to R}{\overset{p}{\longrightarrow}} v$ with L linear and $q \in \mathcal{P}os(u)$ such that $q \geq_{\mathcal{P}} p \cdot F_L$. Then $\underline{u}_q = u[Z[\overline{z}]]_q \underset{i}{\overset{p}{\longrightarrow}} w$ 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 and we call $M = u|_{p\cdot o}$. We have $M = (L\gamma)|_o = L|_o\gamma = X[\overline{x}]\gamma$.

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 $X[\overline{x}]\theta\tau = M[u|_q]_{q'} = M[M|_{q'}]_{q'} = M = X[\overline{x}]\gamma$, since L is linear, the only occurrence of the meta-variable X in L is at position $p \cdot o$ and therefore $L\gamma = L\theta\tau$.

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, $s \xrightarrow[j]{\geq_{\mathcal{P}} p} \xleftarrow[i]{p} t$.*

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 3.12, $u = v\sigma \xrightarrow[j]{} v\tau = t$. By Lemma 5.3, $v \xrightarrow[i]{p} w$ for some w such that $s = w\sigma$. By Lemma 3.14, $u = v\sigma \xrightarrow[j]{}\!\!\!\!\!\rightarrow w\tau$. The result follows. □

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, this transformation is likely to force the use of β-steps to joined them which won't be supported by our criteria.

5.2 Critical peaks

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

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

Lemma 5.6. *We have the following properties*
(1) If L is a pattern or a pre-pattern or ground, then so is $L\uparrow^{\overline{x}}$.
(2) If $L\uparrow^{\overline{x}}\ \theta = t$ then there exists σ such that $L\sigma = t$ and $\mathcal{R}an(\sigma) \subseteq \mathcal{R}an(\theta) \cup \overline{x}$.
(3) If $L\sigma = t$ then there exists θ such that $L\uparrow^{\overline{x}}\ \theta = t$ and $\mathcal{R}an(\theta) = \mathcal{R}an(\sigma) \setminus \overline{x}$.

Proof. (1) By definition. (2) $\sigma = \sigma_{\overline{x}}^{L}\theta$. (3) If $\sigma = \{Y_i \mapsto v_i\}$ then take $\theta = \{Y_i' \mapsto \lambda\overline{x}.v_i\}$. □

We can now define untyped higher-order critical peaks:

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

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

Our 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 5.8 (Critical peak lemma). *Assume* $s \underset{i}{\overset{p}{\leftarrow}} u \underset{j}{\overset{q}{\rightarrow}} t$ *is an overlapping peak of* $j : G \to D$ *onto* $i : L \to R$ *at position* $o \in \mathcal{FP}os(L)$ *such that* $q = p \cdot o$. *Then there is a critical peak* $s' \underset{i}{\overset{\Lambda}{\leftarrow}} u' \underset{j}{\overset{o}{\rightarrow}} t'$ *and a closed substitution* θ *such that* $u'\theta = u|_p$, $s'\theta = s|_p$ *and* $t'\theta = t|_p$.

Proof. By definition of higher-order rewriting, there exists some substitutions γ and σ such that $L\gamma = u|_p$, $G\sigma = u|_q = (L\gamma)|_o$, $s = u[R\gamma]_p$ and $t = u[D\sigma]_q$. Let $\overline{x}$ be the set of variables bound above o in L. We cannot merge γ and σ into a substitution that can be applied to L as the $\overline{x}$ variables may be in $\mathcal{R}an(\sigma)$. However, by Lemma 5.6, there exists σ' such that $G\uparrow^{\overline{x}}\sigma' = (L\gamma)|_o$. Since σ' and γ have disjoint domains we can define $\gamma' = \gamma\sigma'$ and have both $L\gamma' = u|_p$ and $G\uparrow^{\overline{x}}\gamma' = (L\gamma')|_o = u|_q$. Besides, since $o \in \mathcal{FP}os(L)$ and $\mathcal{R}an(\gamma') \cap \overline{x} = \emptyset$ we have $(L\gamma')|_o = L|_o\gamma'$. Therefore, γ' is a solution of the equation $L|_o = G\uparrow^{\overline{x}}$. Let ω be its most general unifier, there exist a substitution θ such that $\omega\theta = \gamma'$. The critical peak is $s' = R\omega \underset{i}{\overset{\Lambda}{\leftarrow}} L\omega = u' = L\omega[G\uparrow^{\overline{x}}\omega]_o \underset{j}{\overset{o}{\rightarrow}} L\omega[D\uparrow^{\overline{x}}\omega]_o = t|_p$. We check that $s'\theta = R\omega\theta = R\gamma' = R\gamma = s|_p$, $u'\theta = L\gamma = u|_p$ and $t'\theta = L\gamma[D\uparrow^{\overline{x}}\gamma']_o = u|_p[D\sigma]_o = t|_p$. □

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 5.9. 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 5.10. *Given a set of positions* O *there exist a unique set of parallel positions* $P \subseteq O$ *and family* $Q > \Lambda$ *of sets of positions such that* $O = P \otimes Q$.

Proof. $P = \{p \in O \mid \forall q \in O, p \not> q\}$ and $Q_p = \{q > \Lambda \mid p \cdot q \in O\}$ □

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 $O \neq \emptyset$, $\underline{O} \neq \emptyset$ and $\overline{O} \subset O$.

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

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 formulation of Tait's parallel rewriting definition.

Note that orthogonal rewriting contains parallel rewriting. Furthermore it is easy to show that our definition of orthogonal reduction coincides with Tait's parallel rewriting. This follows from the property: $t \overset{P}{\underset{\beta}{\otimes\!\!\Longrightarrow}} u \overset{Q}{\underset{\beta}{\otimes\!\!\Longrightarrow}} v$ implies $t \overset{P\cup Q}{\underset{\beta}{\otimes\!\!\Longrightarrow}} v$ if $\forall p \in P, \forall q \in Q, q \not\geq_{\mathcal{P}} p$, 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 5.12. *Assume $q \in \mathcal{P}os(u)$ and $O = P \uplus Q \uplus R \subseteq \mathcal{P}os(u)$ with $P\#q$, $Q \geq_{\mathcal{P}} q$ and $R <_{\mathcal{P}} q$. Then $u \overset{O}{\underset{\beta}{\otimes\!\!\Longrightarrow}} v$ iff $u \overset{P}{\underset{\beta}{\otimes\!\!\Longrightarrow}} \overset{Q}{\underset{\beta}{\otimes\!\!\Longrightarrow}} \overset{R}{\underset{\beta}{\otimes\!\!\Longrightarrow}} v$.*

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

Lemma 5.13. *$(\lambda x.M\ N) \overset{O}{\underset{\beta}{\otimes\!\!\Longrightarrow}} t$ with $\Lambda \in O$ if and only if $O = \{\Lambda\} \uplus 11 \cdot P \uplus 2 \cdot Q$, $M \overset{P}{\underset{\beta}{\otimes\!\!\Longrightarrow}} M'$, $N \overset{Q}{\underset{\beta}{\otimes\!\!\Longrightarrow}} N'$ and $t = M'\{x \mapsto N'\}$.*

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

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 5.14 (LAPOb). *Let $s \overset{q}{\underset{\mathcal{R}ll}{\Longleftarrow}} u \overset{O}{\underset{\beta}{\otimes\!\!\Longrightarrow}} t$ for some set $O >_{\mathcal{P}} q$. Then, $s \overset{\geq_{\mathcal{P}} q}{\underset{\beta}{\otimes\!\!\Longrightarrow}} r \overset{q}{\underset{\mathcal{R}ll}{\Longleftarrow}} t$.*
Besides $u|_q = L\gamma$ and $s = u[R\gamma]_q$ such that $\gamma \overset{}{\underset{\beta}{\otimes\!\!\Longrightarrow}} \sigma$, $t = u[L\sigma]_q$ 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), $\underline{u}_{\underline{O}} \overset{q}{\underset{\mathcal{R}ll}{\longrightarrow}} s'$ for some s' such that $s = s'\overline{u}^{\underline{O}}$. By definition of orthogonal rewriting, $\overline{u}^{\underline{O}} \overset{}{\underset{\beta}{\otimes\!\!\Longrightarrow}} \gamma$ for some γ such that $t = \underline{u}_{\underline{O}}\gamma$.

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

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

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

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

Example 5.16. If $u = \lambda x.\mathtt{f}(\lambda y.y, \lambda z.x)$, then $\{11, 12\}$ and $\{11, 121\}$ are rigid in u while $\{111, 121\}$ is not since $\mathcal{V}ar(u|_{111}) = \{y\} \not\subseteq \mathcal{V}ar(u|_1) = \{x\}$.

Lemma 5.17 (LAPOa). *Let $s \overset{P}{\underset{\beta}{\Longleftarrow\!\!\otimes}} u \overset{q}{\underset{\mathcal{R}ll}{\longrightarrow}} t$, where $\forall p \in P, p \not\geq_{\mathcal{P}} q$. Then $s \overset{Q'}{\underset{\mathcal{R}ll}{\Longrightarrow}} v \overset{P}{\underset{\beta}{\Longleftarrow\!\!\otimes}} t$ for some set $Q' \geq_{\mathcal{P}} \underline{(P \cup \{q\})}$. Besides $s = u'\overline{u}^q$, $v = u'\sigma$ and $t = \underline{u}_q\sigma$ such that $\underline{u}_q \overset{P}{\underset{\beta}{\otimes\!\!\Longrightarrow}} u'$, $\overline{u}^q \underset{\mathcal{R}ll}{\longrightarrow} \sigma$.*

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 rigid set of 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* $\succ_{mul}$ *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 rigid set of positions in* u, *no variable bound above* Q_1 *in* u *can occur in* $M|_{Q_1}$. *It follows that* $x \notin \mathcal{V}ar(M|_{Q_1})$ *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 3.12, 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 $t \overset{P' \uplus \{\Lambda\}}{\underset{\beta}{\Longrightarrow\!\!\otimes}} v = M_v\{x \mapsto N_v\}$, *we can conclude in both cases.*

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

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

rewrites at $1^2 \cdot Q_1 \ \# \ 2 \cdot P_2$, $(\lambda x.M \ N_w) \underset{\mathcal{R}ll}{\overset{1^2 \cdot Q_1}{\Longrightarrow}} (\lambda x.M_t \ N_w) \underset{\beta}{\overset{2 \cdot P_2}{\Longleftarrow\!\!\otimes}} (\lambda x.M_t \ N) = t$. *By stability Lemma 3.13 used at all positions in* Q_1, $M\{x \mapsto N_w\} \underset{\mathcal{R}ll}{\overset{Q_1}{\Longrightarrow}} M_t\{x \mapsto N_w\} \underset{\beta}{\overset{\Lambda}{\Longleftarrow}} (\lambda x.M_t \ N_w)$. *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 do not 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 critical pairs are all joinable with* $\mathcal{R}ll \cup M\eta$. *Then* $\lambda\mathcal{F}$ *is confluent.*

Proof. *Let* $\longrightarrow = \underset{\mathcal{R}ll}{\longrightarrow} \cup \underset{\beta}{\otimes\!\!\Longrightarrow}$. *The relations* $\underset{\lambda\mathcal{F}}{\longrightarrow}$ *and* $\longrightarrow$ *verify the assumptions of the following basic property: assume* $\overset{1}{\longleftarrow\!\!\!\!\rightarrow} \subseteq \overset{2}{\longleftarrow\!\!\!\!\twoheadrightarrow}$ *and* $\overset{2}{\longrightarrow} \subseteq \overset{1}{\longrightarrow\!\!\!\!\twoheadrightarrow}$, *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* $\underset{\mathcal{R}ll}{\longrightarrow\!\!\!\!\twoheadrightarrow}$, $\bot$ *being chosen minimal.*

We now show that all local peaks have decreasing diagrams.

Let $s \underset{}{\overset{P}{\longleftarrow}} u \overset{Q}{\longrightarrow} t$ *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) $s \underset{}{\overset{P}{\Longleftarrow\!\!\otimes}} u \underset{}{\overset{Q}{\otimes\!\!\Longrightarrow}} t$. *Orthogonal* β*-reductions are known to be joinable in at most one step from each side, hence* $s \underset{}{\overset{P'}{\otimes\!\!\Longrightarrow}} v \underset{}{\overset{Q'}{\Longleftarrow\!\!\otimes}} t$ *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 (LAPR*$a$*) 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* $i : L \to R \in \mathcal{R}\mathrm{ll}$, $j : G \to D \in \mathcal{R}\mathrm{ll}$, $q \in p \cdot o$ *and* $o \in \mathcal{FP}os(F_L)$. *By Lemma 5.8, there is a critical peak* $s' \underset{i}{\overset{\Lambda}{\longleftarrow}} u' \underset{j}{\overset{o}{\longrightarrow}} t'$ *obtained by overlapping* G *onto* L *at position* o *and such that* $u|_p = u'\theta$, $s|_p = s'\theta$ *and* $t|_p = t'\theta$. *By assumption, this peak is joinable with rules of* $\mathcal{R}\mathrm{ll} \cup M\eta$. *By the stability Lemma 3.13 and Corollary 3.4, since* θ *is ground, the pair* $s'\theta, t'\theta$ *is joinable with rules of* $\mathcal{R}\mathrm{ll}$. *By monotonicity lemma 3.12, so is the pair* s, t. *Note that there are no facing steps here, since labels decrease strictly along* $\mathcal{R}\mathrm{ll}$*-reductions.*

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

From Lemma 5.12 we have $s \overset{R}{\Longleftarrow\!\!\otimes} v \overset{Q}{\Longleftarrow\!\!\otimes} u' \overset{P}{\Longleftarrow\!\!\otimes} u$ *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 5.14 we have* $v \underset{i}{\overset{q}{\longrightarrow}} r \underset{\beta}{\overset{\geq_{\mathcal{P}} q}{\Longleftarrow\!\!\otimes}} t'$ *and by Lemma 5.17 we have* $s \underset{i}{\Longrightarrow} w \underset{\beta}{\overset{R}{\Longleftarrow\!\!\otimes}} r$. *From Lemma 5.12 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 $\mathcal{R}\mathrm{ll}$-rewrites, which we could have singled out.

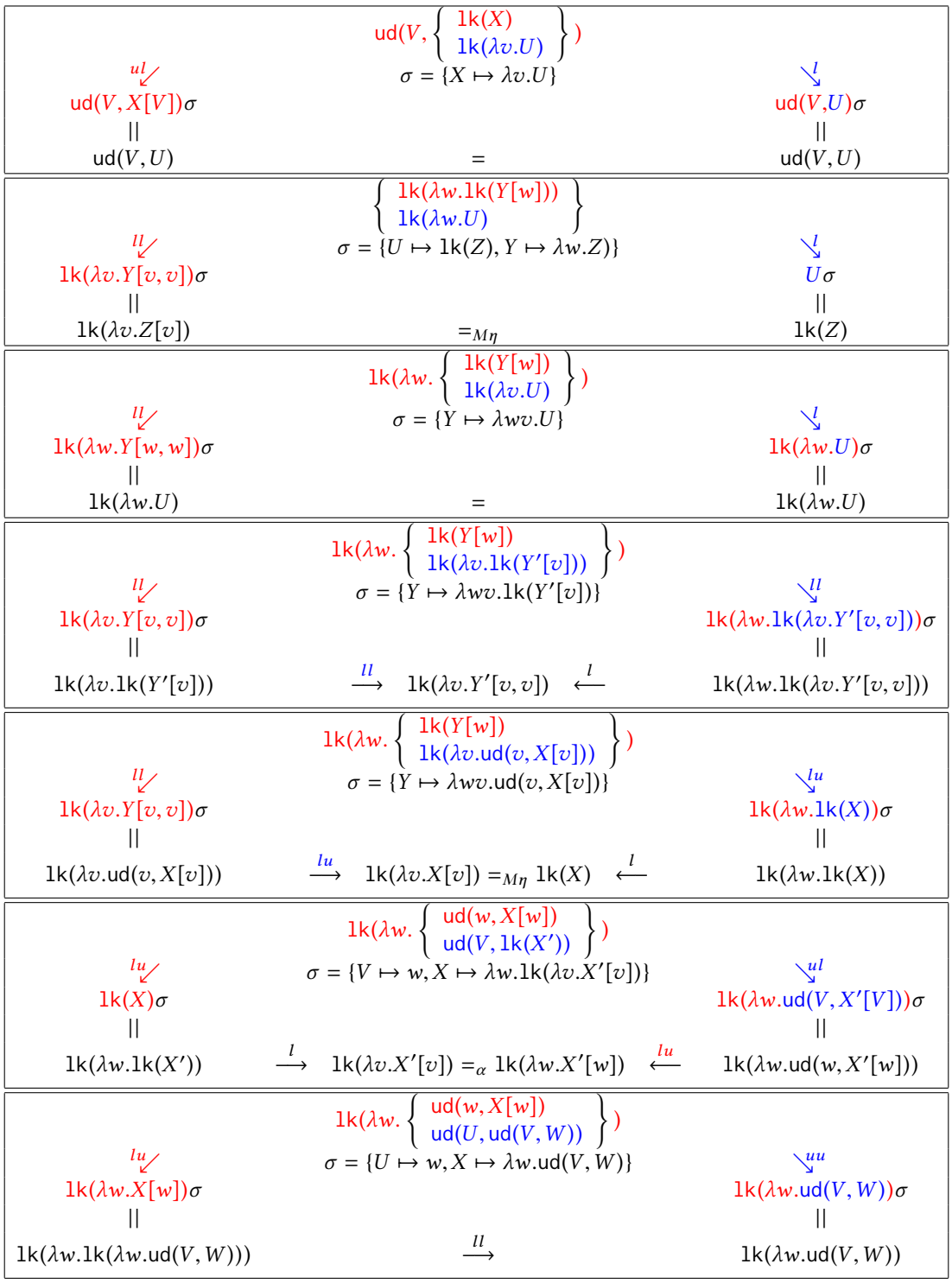
Example 6.2. SOL [11] shows the confluence of the theory of global states for the case of simple types with prenex polymorphism. We show below that it is confluent for any type discipline. To this end, we need to show first that its untyped version is terminating, and then, that the critical pairs are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

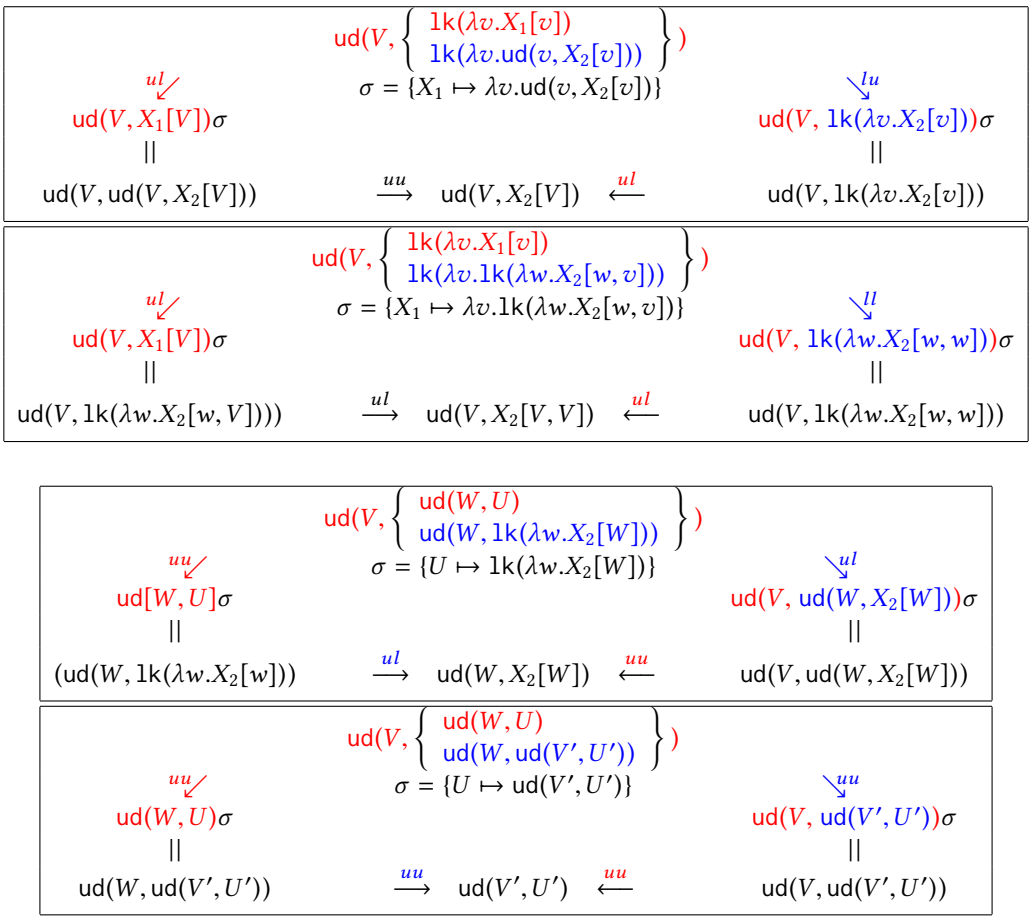
We are left with verifying the joinability of critical pairs of the obtained set of rules. Because the first argument of ud, a value, is now a constant, overlapping two ud-symbols is only possible if these constants are equal. This is what happens when overlapping the original rules, in which the first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

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

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

Our choice for making the calculations is the second set.


ud(V, { lk(X) ; lk(λv.U) })
σ = {X ↦ λv.U}
ul
ud(V, X[V])σ
||
ud(V, U)
=
l
ud(V,U)σ
||
ud(V, U)
{ lk(λw.lk(Y[w])) ; lk(λw.U) }
σ = {U ↦ lk(Z), Y ↦ λw.Z)}
ll
lk(λv.Y[v, v])σ
||
lk(λv.Z[v])
=Mη
l
Uσ
||
lk(Z)
lk(λw. { lk(Y[w]) ; lk(λv.U) })
σ = {Y ↦ λwv.U}
ll
lk(λw.Y[w, w])σ
||
lk(λw.U)
=
l
lk(λw.U)σ
||
lk(λw.U)
lk(λw. { lk(Y[w]) ; lk(λv.lk(Y′[v])) })
σ = {Y ↦ λwv.lk(Y′[v])}
ll
lk(λv.Y[v, v])σ
||
lk(λv.lk(Y′[v]))
ll
⟶
lk(λv.Y′[v, v])
l
⟵
ll
lk(λw.lk(λv.Y′[v, v]))σ
||
lk(λw.lk(λv.Y′[v, v]))
lk(λw. { lk(Y[w]) ; lk(λv.ud(v, X[v])) })
σ = {Y ↦ λwv.ud(v, X[v])}
ll
lk(λv.Y[v, v])σ
||
lk(λv.ud(v, X[v]))
lu
⟶
lk(λv.X[v]) =Mη lk(X)
l
⟵
lu
lk(λw.lk(X))σ
||
lk(λw.lk(X))
lk(λw. { ud(w, X[w]) ; ud(V, lk(X′)) })
σ = {V ↦ w, X ↦ λw.lk(λv.X′[v])}
lu
lk(X)σ
||
lk(λw.lk(X′))
l
⟶
lk(λv.X′[v]) =α lk(λw.X′[w])
lu
⟵
ul
lk(λw.ud(V, X′[V]))σ
||
lk(λw.ud(w, X′[w]))
lk(λw. { ud(w, X[w]) ; ud(U, ud(V, W)) })
σ = {U ↦ w, X ↦ λw.ud(V, W)}
lu
lk(λw.X[w])σ
||
lk(λw.lk(λw.ud(V, W)))
ll
⟶
uu
lk(λw.ud(V, W))σ
||
lk(λw.ud(V, W))



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 to be joined from both sides.

7 IMPACT AND RELATED WORK

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

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

7.1 Relationship to Nipkow's higher-order rewriting

Nipkow's rewriting assumes terms to be simply typed, but it can be easily extended to other typing disciplines. The major requirement is indeed that 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 $|X| = |\overline{x}|$. 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 on whether the rule $L \to R$ belongs to $\mathcal{R}_{nw}$ or $\mathcal{R}_{kp}$: the equality $u = L\sigma$ when L is a Klop left-hand side of rule becomes $u =_{\beta^0} L'\sigma'$ for the corresponding Nipkow left-hand side of rule L'. The same applies to unification of left-hand sides.

We now make these remarks formal:

Definition 7.1. $u \xrightarrow[L\to R]{p} v$ for $L \to R \in \mathcal{R}_{nw}$ 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 $L \to R \in \mathcal{R}_{nw}$. As it is known that η-expanded forms are closed under β-reduction and substitution, we have:

Lemma 7.2. *If* $u \xrightarrow[\mathcal{R}_{nw}]{p} v$ *then* $u = u{\updownarrow}$ *and* $u \xrightarrow[\mathcal{R}_{kp}]{p} \xrightarrow[\beta]{} v = v{\updownarrow}$.

Lemma 7.3. *For all* $u \in \mathcal{T}$ *such that* $u \xrightarrow[\mathcal{R}_{kp}]{p} v$ *and* $v \in \mathcal{T}$, $u{\updownarrow} \xrightarrow[\mathcal{R}_{nw}]{} v{\updownarrow}$.

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

Corollary 7.4. *If* $u \xrightarrow[\beta\eta\mathcal{R}_{kp}]{} v$ *then* $u{\updownarrow} \xrightarrow[\mathcal{R}_{nw}]{} v{\updownarrow}$.

Lemma 7.5. *Assume a rewrite system* $\mathcal{R}$ *such that* $\xrightarrow[\beta]{} \cup \xrightarrow[\mathcal{R}_{kp}]{}$ *is Church-Rosser. Then* $\xrightarrow[\mathcal{R}_{nw}]{}$ *is Church-Rosser on* β*-normal* η*-expanded terms of* $\mathcal{T}$.

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

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

Proof. $\xrightarrow[\mathcal{R}]{}$ satisfies the hypotheses of Theorem 6.1. We conclude that $\xrightarrow[\beta]{} \cup \xrightarrow[\mathcal{R}]{}$ is Church-Rosser and by Lemma 7.5, $\xrightarrow[\mathcal{R}_{nw}]{}$ is too on β-normal η-expanded terms of $\mathcal{T}$. □

This result is of course not really surprising, 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.

7.2 Modularity of higher-order confluence

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

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

7.3 Applicability to theories with annotated abstractions

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 several dependent type theories without difficulty, as well as for other, related rewrite relations, as we have shown with Nipkow's higher-order rewriting.

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

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

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

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

8 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 [17]. 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 [7].

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

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 forthcoming paper. 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 [**?**], for which four inifite families of critical pairs are generated whose joinability uses a proof by (a simple) induction.

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 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, at the price of imposing that meta-variables do not occur nested in one another in the right-hand sides of the rules. This restriction looks very strong. However, any expression such as $X[Y]$ can be transformed into $(X\ Y)$, hence eliminating the nesting. There is of course a general transformation that will eliminate all nestings, 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. It is certainly possible to exhibit examples for which this transformation would work. We have not encountered such a natural example so far.

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 losing confluence, as shown by Klop [15]. 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.

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

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] Bruno Barras, Samuel Boutin, Cristina Cornes, JudicaÃńl Courant, Jean-Christophe FilliÃćtre, Eduardo GimÃĺnez, Hugo Herbelin, GÃĺrard Huet, CÃĺsar MuÃśoz, Chetan Murthy, Catherine Parent, Christine Paulin-Mohring, Amokrane SaÃŕbi, and Benjamin Werner. The coq proof assistant reference manual : Version 6.1. 06 1997.

[4] Bruno Barras and Benjamin Grégoire. On the role of type decorations in the calculus of inductive constructions. In C.-H. Luke Ong, editor, *Computer Science Logic, 19th International Workshop, CSL 2005, 14th Annual Conference of the EACSL, Oxford, UK, August 22-25, 2005, Proceedings*, volume 3634 of *Lecture Notes in Computer Science*, pages 151–166. Springer, 2005.

[5] Gilles Barthe and Morten Heine Sørensen. Domain-free pure type systems. *J. Funct. Program.*, 10(5):417–452, 2000.

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

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

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

[9] 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.

[10] 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.

[11] 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.

[12] J. R. Hindley. An abstract form of the Church-Rosser theorem. i. *J. Symb. Log.*, 34(4):545–560, 1969.

[13] Jean-Pierre Jouannaud and Jiaxiang Liu. From diagrammatic confluence to modularity. *Theor. Comput. Sci.*, 464:20–34, 2012.

[14] Jean-Pierre Jouannaud and Vincent van Oostrom. Diagrammatic confluence and completion. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *ICALP (2)*, volume 5556 of *Lecture Notes in Computer Science*, pages 212–222. Springer, 2009.

[15] Jan Willem Klop. *Combinatory reduction systems*. PhD thesis, CWI tracts, 1980.

[16] Jiaxiang Liu and Jean-Pierre Jouannaud. Confluence: The unifying, expressive power of locality. In Shusaku Iida, José Meseguer, and Kazuhiro Ogata, editors, *Specification, Algebra, and Software - Essays Dedicated to Kokichi Futatsugi*, volume 8373 of *Lecture Notes in Computer Science*, pages 337–358. Springer, 2014.

[17] Richard Mayr and Tobias Nipkow. Higher-order rewrite systems and their confluence. *Theoretical Computer Science*, 192:3–29, 1998.

[18] Dale Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. *Journal of Logic and Computation*, 1(4):497–536, 1991.

[19] Maxwell H. A. Newman. On theories with a combinatorial definition of 'equivalence'. *Ann. Math.*, 43(2):223–243, 1942.

[20] Tobias Nipkow. Higher-order critical pairs. In *Proceedings of the 6th annual IEEE Symposium on Logic in Computer Science (LICS '91)*, pages 342–349, Amsterdam, The Netherlands, July 1991.

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

[22] 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.

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

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

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

[26] 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.

[27] 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.

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

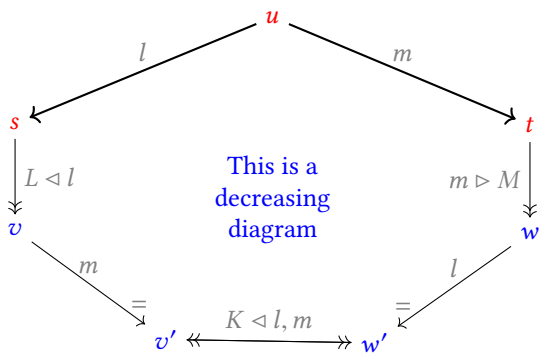


Fig. 2. Decreasing diagram

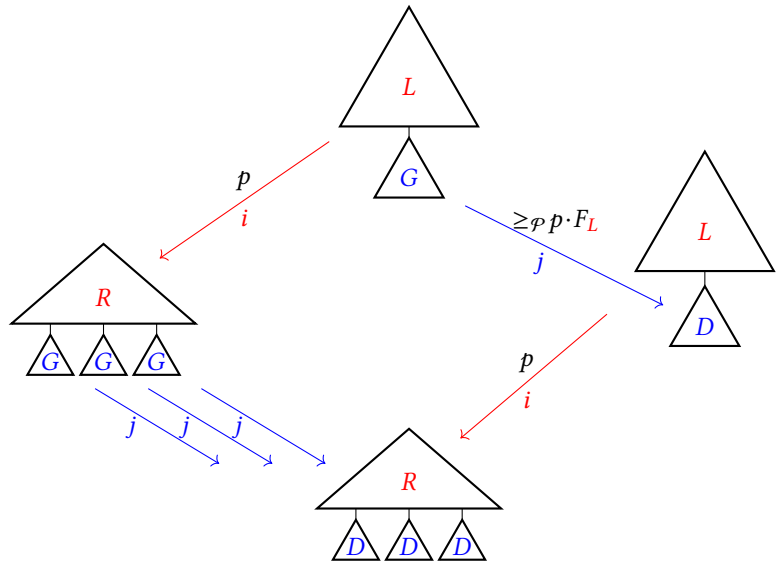


Fig. 3. Ancestor peaks in rewrite theories. L, G stand for terms rewriting to R, D, using a red rule in $\mathcal{R}$ll and a blue rule in $\mathcal{R}$ll $\cup \beta$.

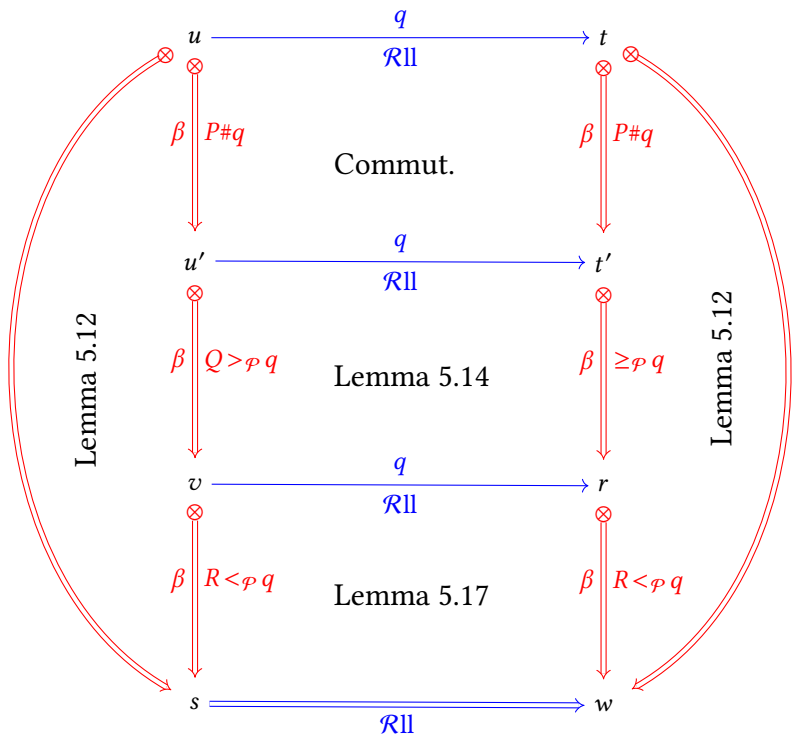


Fig. 4. Construction of a decreasing diagram for heterogeneous local peaks.

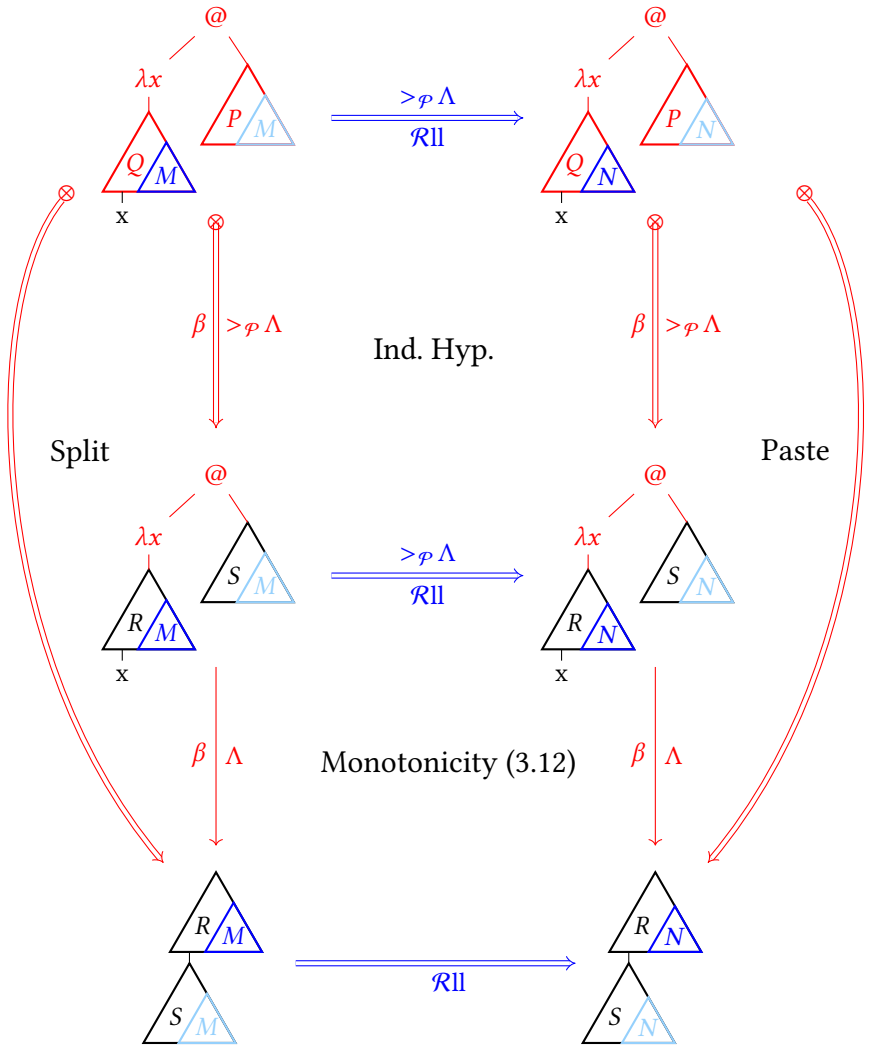


Fig. 5. Construction of a decreasing diagram for peak: Lemma 5.17

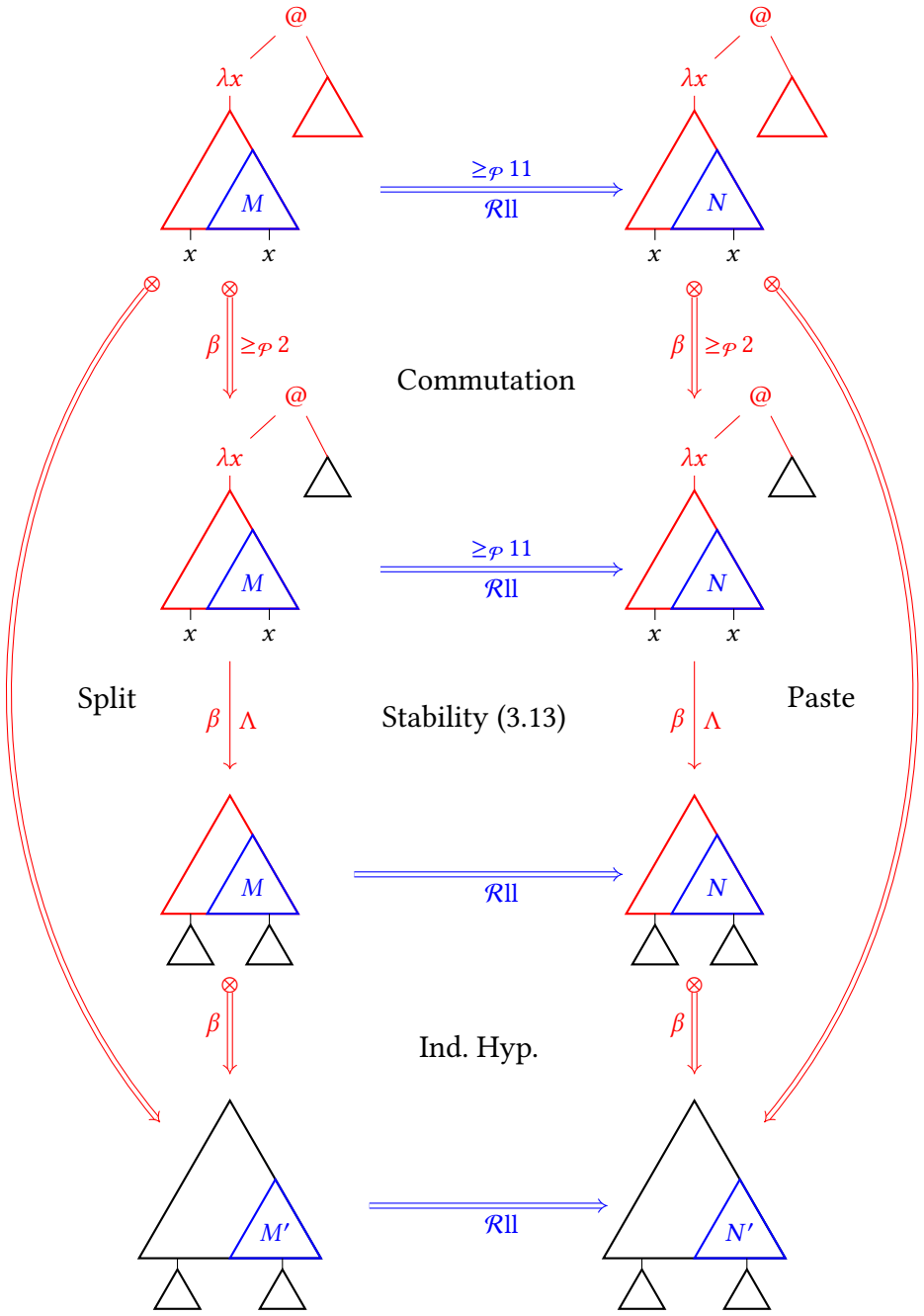


Fig. 6. Construction of a decreasing diagram for peak: Lemma 5.17