Confluence in Non-Left-Linear Untyped Higher-Order Rewrite Theories

Gaspard Férey, Jean-Pierre Jouannaud

Confluence in Non-Left-Linear Untyped Higher-Order Rewrite Theories

Gaspard Férey
Deducteam, LSV, ENS de Paris-Saclay, France
gaspard.ferey@gmail.com

Jean-Pierre Jouannaud
Deducteam, LSV, Université de Paris-Saclay, France
jeanpierre.jouannaud@gmail.com

ABSTRACT

We develop techniques based on van Oostrom's decreasing diagrams that reduce confluence proofs to the checking of critical pairs for higher-order rewrite rules extending beta-reduction on pure lambda-terms. We show that confluence is preserved for a large subset of terms that contains all pure lambda terms. Our results are applied to famous Klop's examples of non-confluent behaviours in presence of convergent rewrite rules and to fragments of various encodings, in a dependent type theory with rewrite rules, of the Calculus of Constructions with polymorphic universes.

KEYWORDS

Lambda calculus, Church-Roser property, confluence, higher-order rewrite rules, non-linear patterns, sub-rewriting, higher-order critical pairs, term layers, decreasing diagrams.

ACM Reference Format:
Gaspard Férey and Jean-Pierre Jouannaud. 2021. Confluence in Non-Left-Linear Untyped Higher-Order Rewrite Theories. In *Proceedings of the 23rd Symposium on Principles and Practice of Declarative Programming, PPDP 2021, Tallinn, Estonia, September 6–8, 2021.* ACM, New York, NY, USA, 14 pages.
https://doi.org/10.1145/NNNNNNN.NNNNNNN

1 INTRODUCTION

Agda, Coq and Dedukti are implementations of dependent type theory that all allow for user-defined rewrite rules. Adding rewrite rules makes it difficult to prove consistency, unless the rewrite rules satisfy some very specific format that includes inductive types, such as the general schema[4], which imposes critical-pair free left-linear rewrite rules. More permissive attempts are non-conclusive yet [5].

The two essential properties of a type theory, consistency and decidability of type checking, follow from three simpler ones: type preservation, termination and confluence. In dependent type theories, confluence is needed to prove type preservation and termination, making all three properties interdependent if termination is used in the confluence proof. This circularity can be possibly broken in two ways: by proving all properties together within a single induction [9]; 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.

PPDP '21, September 6–8, 2021, Tallinn, Estonia

ACM ISBN 978-1-4503-8689-0…$15.00
https://doi.org/10.1145/NNNNNNN.NNNNNNN

Techniques for showing confluence of higher-order dependent theories in presence of critical pairs are restricted to those for which the left-hand sides of rules are linear [7, 8]. We consider here the case of non-left-linear rewrite rules. We have a specific interest in the following examples of non-left-linear rules that pop-up when encoding type-theoretic universes:

$$\max(X,X) \to X$$
$$\mathbf{F}(X,X,Y) \to Y \qquad \mathrm{u}(X,\mathrm{c}(X,Y)) \to Y$$

But non left-linear rules have a bad rewriting behaviour in presence of β-reductions. A counter-example to termination in the simply typed λ-calculus is given in [20]. Numerous counter-examples to confluence in the pure λ-calculus are given in [12], including surjective pairing which came up as a surprise at that time. A striking one is provided by the rule $f(X,X) \to a$, whose interaction with a fixpoint combinator generates diverging reductions that cannot be joined: confluence of the pure λ-calculus cannot be preserved when adding rewrite rules whose left-hand sides are non-linear.

Indeed, non-left-linear rules do not behave as left-linear ones. Consider the first-order rules $f(X,Y) \to a$ and $c \to g(c)$. Then, $f(c,c)$ rewrites to a using the first rule. It also rewrites to $f(c,g(c))$ using the second rule. Since the first rule still applies to $f(c,g(c))$, we get the same result a, although a bit more painfully. Assuming now that the first rule is $f(X,X) \to a$, $f(c,c)$ still rewrites to a, but $f(c,g(c))$ does not anymore. An additional rewrite step $f(c,g(c)) \to f(g(c),g(c))$ is needed, which destroys the so-called strong confluence or diamond property, or, in modern terms, van Oostrom's decreasing diagram property. Adding then the rewrite rule $f(X,g(X)) \to b$ breaks confluence since $f(c,c)$ can rewrite to a and b, both in normal form.

One can also say that adding the rule $c \to g(c)$ to the confluent system $\{f(X,X) \to a, f(X,g(X)) \to b\}$ breaks its confluence. Since this first-order rule mimics a fixpoint, adding the untyped lambda calculus to that systems destroys its confluence as well. The point is that these two rules have a *hidden* critical pair, since c and $g(c)$ are now equated by the fixpoint rule. In case all those critical pairs are joinable, as is not the case in this example, then confluence of first-order systems is preserved [14]. Restoring strong confluence of the peak is done by using a rewrite relation, called *sub-rewriting*, which has the effect of linearizing the appropriate instances of the non-left-linear rules. The confluence proof is then done by induction on a stratification of terms by *layers* used as labels for van Oostrom's decreasing diagrams technique. A key assumption in the proof is that rewriting is layer non-increasing.

Unfortunately, this technique falls apart for higher-order systems, since some beta rewrites steps are layer increasing. Indeed, adding non-left-linear rules does not preserve confluence of the

pure lambda calculus, as shown by Klop. The question then becomes: on which subset(s) of terms is confluence preserved? and what stratification can make beta be layer non-increasing?

A first answer was given in [1] by categorizing terms into two layers: a bottom *confined* layer made of *first-order* terms, and an upper layer for all others. With confinement, arbitrary first-order rules are allowed, but non-left-linear higher-order rules are not. Our first example passes this restriction, because the meta-variable X denotes here a universe, encoded in [1] as an integer equipped with first-order operations. The other examples do not.

Our new answer is a stratification of terms by infinitely many layers defined by a simple typing system based on integers as base types which ensures that non-linear variables of a left-hand side of rules are substituted by terms of a strictly smaller layer than the left-hand side instance itself. By also making beta be layer non-increasing, this integer type system defines the layers we need to show confluence for sub-rewriting first, and then for rewriting. Our main result is that the user's rules preserve confluence of beta on the set of integer-typable terms provided they are layer non-increasing and their critical pairs are joinable by decreasing diagrams.

Our confluence result is not only theoretical: our three examples pass the test. Furthermore, the integer type system can be easily implemented within a standard type checker for dependent types with little overhead.

Section 2 recalls our model of higher-order rewriting and Section 3 the axioms that term layers should satisfy. We show in Section 4 that confluence of the λ-calculus is preserved on layered terms by adding rules whose critical pairs are joinable by decreasing diagrams. The proof is done by induction on layers, using van Oostrom's decreasing diagram theorem. Section 5 describes our layering type system, and applies it to Klop's and our examples.

2 THE HIGHER-ORDER REWRITING CALCULUS

Given an *untyped* annotated 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 $\lambda\mathcal{F}$, a calculus whose reduction relation extends the β-rule of the underlying λ-calculus by a set R of user-defined rewrite rules built over that vocabulary [8].

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

$\lambda\mathcal{F}$ is a mix of the pure annotated lambda-calculus and Klop's combinatory reduction systems [12], that fits with incarnations of dependent type theory such as Agda, Coq and Dedukti. Terms are those of the annotated lambda calculus enriched with $\mathcal{F}$*-headed* terms of the form $f(\overline{u})$ with $f \in \mathcal{F}$, $\overline{u}$ denoting a list of terms separated by commas, 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 $|f|$. Variables have arity zero, meta-variables have a maximum arity. The set of terms, $\mathcal{T}_{\lambda\mathcal{F}}$, is defined by the following grammar rules:

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

The abstraction operator $''\lambda x : .''$ has arity 2: the syntax is slightly richer than usual in order to enable abstracting step by step derivations in dependently typed lambda calculi by derivations in $\lambda\mathcal{F}$. The calculus without annotation being of course itself an abstraction of $\lambda\mathcal{F}$, all results presented here adapt straightforwardly to that calculus by forgetting annotations. We will use this facility unannounced in examples.

Following usage, we write a for $a(\,)$, X for $X[\,]$ and $f(x\ y)$ for $f((x\ y))$. We use the small letters $f, g, h, \ldots$ for function symbols, $x, y, z, \ldots$ 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 $\mathcal{X} \cup \mathcal{Z}$. By function symbols, we sometimes mean those in $\mathcal{F}$, as well as application and abstraction.

We use the notation $|\_|$ for various quantities besides symbols arities, e.g., the length of a list, the size of an expression, the cardinality of a set. We use $[m..n]$ for the list of natural numbers from m to n, and $\overline{u}[m..n]$ for the sublist $u_m \ldots u_n$ of $\overline{u}$.

Unlike function symbols and Klop's meta-variables, meta-variables here have an arity which is not fixed, but bounded, a handy feature used in Dedukti that avoids writing complex applications of variable length as in $(X\ a)$ and $(((X\ a)b)c)$ which are replaced by $X[a]$ and $X[a, b, c]$ if $|X| \geq 3$.

Positions in terms are words over the natural numbers, using Λ for the empty word, $\cdot$ for concatenation, P/p for $\{q : q \cdot p \in P\}$, $\leq_{\mathcal{P}}$ for the prefix order (*above*), $\geq_{\mathcal{P}}$ for its inverse (*below*), $<_{\mathcal{P}}$ and $>_{\mathcal{P}}$ for their strict parts, $p\#q$ for incomparable positions (*parallel*), P_{min} for the set of minimal positions in P, and $p \geq_{\mathcal{P}} Q$ ($p \leq_{\mathcal{P}} Q$, resp.), Q a set of parallel positions, for $\exists q \in Q :\ p \geq_{\mathcal{P}} q$ ($p \leq_{\mathcal{P}} q$, resp.).

Given a term M, we use: $M(p)$ for its symbol at positions p; $ar(M)$ for the number of abstractions at the head of M; $\mathcal{P}os(M)$, $\mathcal{FP}os(M)$, $\mathcal{VP}os(M)$, $M\mathcal{P}os(M)$ for the following respective sets of positions of M: all positions, the positions of function symbols, of free variables, and of meta-variables; $\mathcal{V}ar(M)$ for its sets of free variables; $M\mathcal{V}ar(M)$, $\mathcal{MV}ar^{l}(L)$ and $\mathcal{MV}ar^{nl}(M)$ for its sets of arbitrary, linear and non-linear meta-variables; A term M is *pure* if it has no function symbol, *ground* if $\mathcal{V}ar(M) = \varnothing$, *closed* if $M\mathcal{V}ar(M) = \varnothing$ (let $\mathcal{T}$ be their set), and *linear* if $\mathcal{MV}ar^{nl}(M) = \varnothing$.

2.2 Substitutions

A *substitution* is a mapping from a finite set of variables and meta-variables to terms, written $\sigma = \{x_1 \mapsto M_1, \ldots, x_n \mapsto M_n\}$, or $\sigma = \{\overline{x} \mapsto \overline{M}\}$, where $ar(M_i) \geq |x_i|$, which extends to an *arity-preserving*, *capture-avoiding* homomorphism on terms. Let the *domain* of σ be $\mathcal{D}om(\sigma) = \{x_1, \ldots, x_n\} \subseteq \mathcal{X} \cup \mathcal{Z}$, while $\mathcal{R}an(\sigma) = \bigcup_{i=1}^{n} \mathcal{V}ar(M_i) \cup M\mathcal{V}ar(M_i)$ 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, the result $t\sigma$ of substituting the term t by the substitution σ, called an *instance* of t, is defined by induction on t as follows:

$x_i\sigma = \sigma(x_i)$; $y\sigma = y$ 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 if it satisfies that condition).

Assuming now $Z \mapsto \lambda\overline{x}.s \in \sigma$, where s is not an abstraction, the more complex rule for meta-variables inspired from Klop's definition of substitution in the case of fixed arities [12] is as follows:

(1) case $|\overline{u}| = m \leq n = |\overline{x}|$: then
$Z[\overline{u}]\sigma = \lambda\overline{x}[m{+}1..n].s\{\overline{x}[1..m] \mapsto \overline{u}\sigma\}$,
(replacement of missing arguments of Z is delayed)
(2) case $|\overline{u}| = m > n$: since $ar(s) \geq |Z| - |\overline{x}| \geq |\overline{u}| - |\overline{x}| > 0$, let $s = Y[\overline{t}]$ and $\overline{u} = \overline{v}\ \overline{w}$ with $|\overline{v}| = n$ and $|Y| - |\overline{t}| \geq |\overline{w}|$; then $Z[\overline{u}]\sigma = Y[\overline{t}\{\overline{x} \mapsto \overline{v}\sigma\}, \overline{w}\sigma]$.

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. We denote by $\mathcal{P}os(\sigma)$ the sequence $\{\mathcal{P}os(\sigma(x_i))\}_i$ of sets of positions of σ.

Let for example s be the term $f(X[(x\ \ y), y])$, where X has two arguments, $(x\ \ y)$ and y, and σ be the substitution $\{X \mapsto \lambda x'y'z'.g(x', y', z'), y \mapsto a\}$. We get $s\sigma = f(\lambda z'.g((x\ y)\sigma, y\sigma, z'\sigma)) = f(\lambda z'.g((x\ a), a, z'))$. Let us now compare with the instance by σ of the term $u = f((X\ (x\ y))\ y)$, in which X is applied successively to $(x\ y)$ and y. Then $u\sigma = f((\lambda x'y'z'.g(x', y', z')\ \ (x\ a))\ a)$. We can see that $u\sigma$ reduces to $s\sigma$ in two β-steps: substitutions of meta-variables hides those reductions, this will be the key to making higher-order rewriting behave like first-order rewriting.

Substitutions extend to sequences of terms and substitutions.

2.3 Splitting and linearization

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 $\overline{u}^P\underline{u}_P = u$.

Let, for example, u be the term $f(\lambda x.g(a\ \ x), f(a, a))$, and P the set $\{1^3, 2{\cdot}1\}$ of parallel positions. Then, $\underline{u}_P = f(\lambda x.g(X[x]), f(Y, a))$ and $\overline{u}^P = \{X \mapsto \lambda y.(a\ \ y), Y \mapsto a\}$.

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 takes place independently in a term and in its context, as it will often happen here.

Splitting a term u at the set of parallel positions P of its meta-variables yields a linear term u^{lin}, the *linearization* of t. A convention to be used throughout the paper is to denote by X^p the meta-variable at position p in u^{lin}, if X is the meta-variable at position p in u. X is called the *kernel* of X^p.

Let for example u be the term $f(\lambda x.g(X[x]), f(X, a))$, hence $P = \{1^3, 2 \cdot 1\}$. Then, $u^{lin} = \underline{u}_P = f(\lambda x.g(X^{1{\cdot}1{\cdot}1}[x]), f(X^{2{\cdot}1}, a))$ and $\overline{u}^P = \{X^{1{\cdot}1{\cdot}1} \mapsto \lambda y.X[y]), X^{2{\cdot}1} \mapsto X\}$.

2.4 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 $u \xrightarrow[R]{<_{\mathcal{P}} p} v$ or $u \xrightarrow[R]{\#P} v$. We use $\longrightarrow\!\!\!\!\!\rightarrow$ for the reflexive transitive closure of $\longrightarrow$, and $\longleftarrow$ for its inverse.

Two different kinds of reductions coexist in $\lambda\mathcal{F}$, functional and higher-order reductions, both operating 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.

2.5 Functional reductions

Functional reduction is the relation on terms generated by the rule $\beta_\alpha : (\lambda x : u.v\ \ w) \longrightarrow v\{x \mapsto w\}$. The usually omitted α-index stresses that renaming bound variables, called *α-conversion*, is built-in. As is customary [17], the particular case for which v is a variable is denoted by β^0. Instantiating a β^0-step may yield a full β-step. For example, $s = (\lambda x.(\lambda y.g(y)\ x)\ a) \xrightarrow[\beta^0]{1\cdot1} (\lambda x.g(x)\ a) \xrightarrow[\beta]{\Lambda} g(a)$ while $s \xrightarrow[\beta]{\Lambda} (\lambda y.g(y)\ a) \xrightarrow[\beta]{\Lambda} g(a)$.

2.6 Higher-order reductions

Higher-order reductions result from rules whose left-hand sides are higher-order patterns in Miller's or Nipkow's sense [15], although they need not be typed here:

Definition 1 (Pattern). *A* pre-redex *of arity n in a term L is an unapplied meta-term $Z[\overline{x}]$ whose arguments $\overline{x}$ are n pairwise distinct variables. A* pre-pattern *is a ground β-normal term all of whose meta-variables occur in pre-redexes. A* pattern *is a pre-pattern which is not a pre-redex nor an abstraction.*

It is important to assume, as does Nipkow, that pre-patterns are β-normal. Note that erasing types from a Nipkow's pattern yields a pattern in our sense, since his pre-redexes being of base type, they cannot be applied.

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

Definition 2 (Fringe). *The* fringe *F_L of a pre-pattern L is the set of parallel positions of its pre-redexes, defining $F_\beta = \{1{\cdot}1, 1{\cdot}2, 2\}$ for convenience. We denote by F_L^l and F_L^{nl} the subsets of F_L made of the positions of its linear and non-linear meta-variables, respectively. The set of* functional positions *of L is $\mathcal{FP}os(L) = \{p : p \not\geq_{\mathcal{P}} F_L\}$.*

Example 1. *The term $L = f(\lambda xyz.g(X, X[x, y]))$ is a pattern. Its pre-redexes are the terms X 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 $f(\lambda x.X[x, x])$, $f(X[a])$, $f(X[Y])$, $f(X\ Y)$, and $\lambda x.f(X[x])$ are not patterns.*

Note that the set of functional positions coincides with the usual notion for first-order terms.

We can now define higher-order rules and rewriting:

Definition 3 (Rule). *A (higher-order)* rule *is a triple $i : L \to R$, whose (possibly omitted)* index *i is a name, left-hand side L is a pattern, and right-hand side R is a ground β-normal term such that $M\mathcal{V}ar(R) \subseteq M\mathcal{V}ar(L)$. The rule is* left-linear *if L is linear,* right-linear *if R is linear, and* linear *if both left- and right-linear.*

Definition 4 (Higher-order rewriting). *Given open term u, position $p \in \mathcal{P}os(u)$, and rule $i : L \to R$, u rewrites with i at p, written $u \xrightarrow[i]{p} v$, iff $u|_p = L\gamma$ for some substitution γ, and $v = u[X[\overline{x}]]_p\{X \mapsto$*

$\lambda\overline{x}.R\gamma\} = u[R\gamma]_p$, *where* $\overline{x}$ *is the list of variables of* $u|_p$ *which are bound above the position* p *in* u. *We write* $u \xrightarrow[\mathcal{R}]{p} v$ *for* $\exists i \in \mathcal{R} : u \xrightarrow[i]{p} v$.

Let's now make our splitting notations fully explicit in $u \xrightarrow[i]{p} v$.

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

Example 2. *Let* $L = der(\lambda x.sin(F[x])) \to R = \lambda x.cos(F[x])$, *and take for* σ *the identity substitution* $\{F \mapsto \lambda x.x\}$. *Then,* $L\sigma = der(\lambda x.sin(x))$ *and* $R\sigma = cos(x)$, *hence* $der(\lambda x.sin(x)) \longrightarrow \lambda x.cos(x)$.

In sharp contrast with Nipkow [15], we observe that we do not need matching modulo β^0 explicitly, since the needed β^0-steps are now hidden in Klop's definition of substitution for meta-variables. Nor do we assume that u, or v, is β-normal, or even β-normal up to position p. We cannot for two reasons: β-normal forms may not exist, and we need monotonicity and stability properties:

Lemma 1 (Monotonicity). *Let* $s \xrightarrow[L\to R]{p} t$ *and* u *a term such that* $q \in \mathcal{P}os(u)$. *Then,* $u[s]_q \xrightarrow[L\to R]{q\cdot p} u[t]_q$.

Lemma 2 (Stability). *Let* $s \xrightarrow[L\to R]{p} t$ *and* σ *a substitution. Then* $s\sigma \xrightarrow[L\to R]{p} t\sigma$.

Both properties extend to rewriting at a set of parallel positions.

2.7 Rewrite theories

Definition 5. *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* $\mathcal{F}$, *defining the rewrite relation of* $\lambda\mathcal{F}$ *as* $\xrightarrow[\mathcal{R}\cup\beta_\alpha]{}$, *also written* $\xrightarrow[\mathcal{R}\beta]{}$.

Rewrite theories have been used successively in Isabelle [19], in Dedukti [2], and in Agda and Coq [5]. The rewrite relation implemented in Dedukti is the one we just described, Klop's notion of substitution for meta-variables being hard-wired.

In this paper, we are interested in rewrite theories whose left-hand sides of rules may be non-linear, and whose rewrite relation is therefore non-confluent.

The precise problem we address here is to approximate the subset of closed terms on which confluence of beta-reductions is preserved, assuming $\mathcal{R}$ satisfies some reasonable assumptions. Finding a meaningful such subset is the open problem we consider.

2.8 Local peaks and critical peaks

Our confluence proof is based on the analysis of *local peaks* of the relation $\xrightarrow[R\beta]{}$, which are triples $s, u, t \in \mathcal{T}$ such that $s \xleftarrow[L\to R]{p} u \xrightarrow[G\to D]{q} t$. There are 3 categories of them: *disjoint* if $p\#q$, *ancestor* if $p \geq_{\mathcal{P}} q \cdot \mathcal{FP}os(L)$, and *overlapping* if $q \in p \cdot \mathcal{FP}os(L)$. The analysis of overlapping peaks requires computing higher-order critical pairs, introduced in [18], [15]. 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. A critical peak is not local in our sense, since it may involve meta-variables. A further 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 resorts to lifting (of which splitting is a variant):

Definition 6 (Lifting). *Given a term* L *and a list* $\overline{x}$ *of pairwise different variables such that* $\mathcal{V}ar(L) \cap \overline{x} = \varnothing$, *we call* lifting *of* L *with respect to* $\overline{x}$, *denoted by* $L\uparrow^{\overline{x}}$, *the term* $L\sigma_L^{\overline{x}}$, *where* $\sigma_L^{\overline{x}} = \{Y \mapsto Y'[\overline{x}] : Y \in M\mathcal{V}ar(L), Y' \text{ fresh}, |Y'| = |Y| + |\overline{x}|\}$.

Definition 7 (Critical peak). *Let* $i : L \to R$ *and* $j : G \to D$ *be rules in* $\mathcal{R}$ *that are renamed apart,* $o \in \mathcal{FP}os(L)$ *and* $\mathcal{V}ar(L|_o) = \overline{x}$ *such that the equation* $\lambda\overline{x}.L|_o = \lambda\overline{x}.G\uparrow^{\overline{x}}$ *has a most general solution* σ. *Then,* $R\sigma \xleftarrow[i]{\Lambda} L\sigma \xrightarrow[j]{o} L\sigma[D\uparrow^{\overline{x}}\sigma]_o$ *is a* critical peak *of* j onto i *at position* o. *The associated* critical pair *is* $\langle R\sigma, L\sigma[D\uparrow^{\overline{x}}\sigma]_o\rangle$.

The rôle of lifting is to capture the variables of $\overline{x}$ in $G\sigma$ by making explicit the dependencies of the meta-variables of G upon the variables of L bound above o. By taking $Y' \mapsto \lambda\overline{x}.v \in \theta$ iff $Y \mapsto v \in \sigma$, we get $G\sigma = G\uparrow^{\overline{x}}\theta$ and $\mathcal{R}an(\theta) \cap \overline{x} = \varnothing$, hence $L\sigma = L\sigma[G\uparrow^{\overline{x}}\theta]_p$.

Here is the untyped analog of Nipkow's critical pair lemma developed for the case of simply typed higher-order rewrite rules:

Lemma 3 (Critical peak lemma). *Assume* $s \xleftarrow[i]{p} u \xrightarrow[j]{q} t$ *is an overlapping peak of rule* $j : G \to D$ *onto rule* $i : L \to R$ *at position* $o \in \mathcal{FP}os(L)$ *such that* $q = p \cdot o$. *Then there is a critical peak* $s' \xleftarrow[i]{\Lambda} u' \xrightarrow[j]{o} t'$ *and a closed substitution* θ *such that* $u'\theta = u|_p$, $s'\theta = s|_p$ *and* $t'\theta = t|_p$.

Finally, we shall also need the following easy property:

Lemma 4. *Let* L, G *be linear patterns sharing no meta-variable and a position* $p <_{\mathcal{P}} F_L$ *such that* $Var L|_p = \overline{x}$. *Assume* $L|_p$ *and* $G\uparrow^{\overline{x}}$ *have most general unifier* σ. *Then,* $H = L[G\uparrow^{\overline{x}}]_p\sigma$ *is a linear pattern.*

Higher-order unification of (untyped) patterns is described in [8] for linear patterns, where its extension to non-linear ones is also sketched, following the standard path. We take the existence and computation of a most general unifier for granted. Assuming that rewrite rules are typed checked, as they are in Agda, Coq, Dedukti, this assumption follows indeed from typed pattern unification.

In order to ease the reading, we assume from now on that critical peaks need no lifting, that is, $\mathcal{V}ar(L|_p) = \varnothing$ as soon as a left-hand side of rule G unifies with a left-hand side of rule L at position p. This assumption is only for convenience, and fits with our examples.

This ends up this introduction to $\lambda\mathcal{F}$, first defined in [8]. From now on, we may denote by $\longrightarrow$ the rewrite relation of $\lambda\mathcal{F}$.

3 LAYERED REWRITING

Stratification of terms is at the core of our technique for proving confluence of rewriting on the set $\mathcal{T}$ of closed terms. We recall the notion of *confined* terms and rewrites, the first level of our hierarchy of closed terms, before to give the requirements for the whole hierarchy. Rewriting can thus be classified by layers according to that hierarchy. We then introduce sub-rewriting, in which different

occurrences of a non-linear variable in a rule may be instantiated by different terms provided they are joinable in a strictly lower layer, and state its basic properties, including a reduction of its overlapping peaks to the higher-order critical peaks we just defined. We end up recalling Tait's notion of orthogonal β-reductions.

3.1 Confinement

Confinement was introduced in [1], the only work we are aware of, apart from Klop's [12], that addresses the problem of confluence of a set of higher-order rules containing possibly non-left-linear rules. Since these rules can only rewrite terms belonging to a subset of first-order terms, their confluence can be checked by standard methods. Blending them with left-linear higher-order rules is a non-trivial problem already, treated in [1].

To this end, the signature $\mathcal{F}$ is split into two disjoint subsets, $\mathcal{F}_c$, the set of *confined* function symbols, and $\mathcal{F}_u$, the set of *unconfined* function symbols, so that $\mathcal{F} = \mathcal{F}_c \uplus \mathcal{F}_u$. We denote by $\mathcal{T}_c \subseteq \mathcal{T}_{\lambda\mathcal{F}}$ the set of *confined* first-order terms built over $\mathcal{F}_c \cup \mathcal{X}$.

Accordingly, the rewrite system $\mathcal{R}$ is split into disjoint sets of rules, R_c and R_u, so that $\mathcal{R} = R_c \cup R_u$. Rules in R_c are *confined*, they operate on confined first-order expressions. We do not need to know what these rules are, we will not need them but the rewrite relation they generate on confined terms. Rules in R_u are higher-order rules, possibly non-left-linear, which operate on non-confined expressions, hence are headed by a symbol from $\mathcal{F}_u$ or an application.

3.2 Term layering

From now on, we assume a subset $\mathcal{L} \subset \mathcal{T}$ of *layered terms*, union of subsets $\mathcal{L}_n$, the set of terms belonging to *layer* $n \in \mathbb{N}$. Layers need not be disjoint. We write $\mathcal{L}_{\leq n} = \bigcup_{k \leq n} \mathcal{L}_k$. Variables need belong to a layer too, hence are pairs of the form $x : n$. Substitutions are written $\{x_i : n_i \mapsto u_i\}_i$, where $n_i \in \mathrm{Nat}$. A meta-variable in a rule has no a priori layer, its layer will be determined when matching a given closed term against that rule. More generally, open terms such as left-hand and right-hand sides of a rule, have no layer.

Definition 8. *A substitution σ is* well-layered *if $\forall x : n \mapsto u \in \sigma : u \in \mathcal{L}_{\leq n}$. We denote by WLS their set.*

We assume the following closure properties of layered terms:

- **(H0)** $\mathcal{L}_0 = \mathcal{T}_c$
- **(H1)** $t \in \mathcal{L}_n$ and u is a subterm of t implies $u \in \mathcal{L}_{\leq n}$
- **(H2)** $t \in \mathcal{L}_n$ and $t \longrightarrow u$ implies $u \in \mathcal{L}_{\leq n}$
- **(H3)** $t \in \mathcal{L}_n$ and $\sigma \in WLS$ implies $t\sigma \in \mathcal{L}_{\leq n}$
- **(H4)** $L \to R \in \mathcal{R}_u$, $X \in \mathcal{MV}ar^{nl}(L)$, $L(p) = X$ and $L^{lin}\sigma \in \mathcal{L}_n$ implies $X^p\sigma \in \mathcal{L}_{<n}$
- **(H4')** $L \to R \in \mathcal{R}_u$, $X \in \mathcal{MV}ar^{l}(L)$, $X \in \mathcal{MV}ar^{nl}(R)$ or X nested in R (i.e., $R(p) = Y$ and $R(p \cdot q) = X$ for some $q \neq \Lambda$), and $L\sigma \in \mathcal{L}_n$ implies $X\sigma \in \mathcal{L}_{<n}$

Consider for example the rule $F(X, X, Y) \to Y$, and let $u \in \mathcal{L}_m, v \in \mathcal{L}_{\leq m}, w \in \mathcal{L}_n$ and $F(u, v, w) \in \mathcal{L}_k$. Then, by **(H1,H3,H4)**, $k \geq m + 1$ if $m \geq n$ and $k \geq n$ if $n > m$.

As can be guessed, our goal is to describe techniques for showing that the derivation relation $\underset{\mathcal{R}\beta}{\longrightarrow}$, which cannot be confluent on $\mathcal{T}$ if $\mathcal{R}$ contains non-left-linear rules, is nevertheless confluent on $\mathcal{L}$. To make sense, $\mathcal{L}$ should be a large enough subset of closed terms. In particular, it should contain all instances of L and R by a well-layered substitution σ whenever $L \to R \in \mathcal{R}$ and $L\sigma$ and $R\sigma$ are closed terms. We shall see that $\mathcal{L}$ can be made indeed very large. In particular, it will contain the whole set of pure annotated λ-terms.

3.3 Layer Rewriting

Let $\Longrightarrow$ a rewriting relation on $\mathcal{L}$, possibly rewriting several redexes at once. Its steps can be labeled by layers as follows:

Definition 9 (Layer rewriting). *Let $u \underset{L \to R}{\overset{P}{\Longrightarrow}} v$ for some $u \in \mathcal{L}$, set of positions P and rule $L \to R$. We say that u rewrites to v*

- *at layer n, written $u \underset{L \to R, n}{\overset{P}{\Longrightarrow}} v$, if $\forall p \in P : u|_p \in \mathcal{L}_n$,*
- *below layer n, written $u \underset{L \to R, <n}{\overset{P}{\Longrightarrow}} v$, if $\forall p \in P : u|_p \in \mathcal{L}_{<n}$.*

The rule name $L \to R$ may be omitted, or replaced by $\mathcal{R}$. The wording layered reduction *is reserved to plain rewriting (P is a singleton).*

Confined rules rewrite at layer 0 only. Non-confined rules rewrite at layers $n \geq 1$. Note also that we do not care about the layer of the whole term u, only the redexes' layers play a rôle in layer rewriting.

Layer rewriting has been crafted to satisfy the two usual basic properties, monotonicity and stability:

Lemma 5. *Let $\Longrightarrow$ be a monotonic, stable relation on $\mathcal{L}$. Then $\underset{n}{\Longrightarrow}$ and $\underset{\leq n}{\Longrightarrow}$ are monotonic, and $\underset{\leq n}{\Longrightarrow}$ is stable.*

3.4 Sub-rewriting

A main idea of the confluence proof to come is to proceed by induction on the layer of terms in $\mathcal{L}$. In order to prove confluence in layer $n + 1$, the induction hypothesis will guarantee confluence in all layers up to n. This remark is at the heart of the sub-rewriting relation. Sub-rewriting was introduced in [13] and developed in [14], both in a first-order setting. We go a step further here with an axiomatic setting that captures $\lambda\mathcal{F}$. If an instance of the left-hand side of a higher-order rule belongs to some layer, its non-linear variables must be instantiated by terms of a strictly lower layer by **(H4)**. This allows to *equalize* different values of these variables by rewriting recursively in a strictly lower layer. Though their use is the same, layers defined in [14] are completely different from the abstract layers defined here and the concrete layers defined in Section 5.

Definition 10 (Sub-rewriting). *A term $u \in \mathcal{L}$* sub-rewrites *to a term v at position $p \in \mathcal{P}os(u)$ for some rule $i : L \to R$, written $u \underset{\mathcal{R}}{\overset{p}{\rhd\!\!\longrightarrow}} iv$, if there exists a substitution θ such that:*

(i) $u \xrightarrow{\geq_{\mathcal{P}} p \cdot F^{nl}_L} u[L\theta]_p$ if $i \in \mathcal{R}_u$ and $u = u[L\theta]_p$ if $i \in \mathcal{R}_c$,

(ii) $v = u[R\theta]_p$.

The term $u|_p$ is called a sub-rewriting redex *of u at p. The substitution θ is an* equalizer *of L, and the rewrite steps from $u|_p$ to $L\theta$ constitute the corresponding* equalization *steps if $L \to R \in \mathcal{R}_u$.*

Sub-rewriting can be categorized by layers, with the expected notation. Sub-rewriting at layer 0 coincides with plain rewriting.

An important property of a sub-rewriting redex $L\theta$ that illustrates the informal introduction of its definition, is that it is an

instance of L^{lin} by two substitutions, one, σ, for its variables originating from the linear variables of L, and one, τ, for those originating from the non-linear variables of L:

Lemma 6. *Let $u \xrightarrow[\mathcal{R}_u]{p} v$. Then there exist a rewrite rule $L \to R \in \mathcal{R}$, substitutions σ, τ, δ, and an integer n such that:*

(1) $u \xrightarrow[L \to R, n]{p} v$
(2) $\mathcal{D}om(\sigma) = \mathcal{MV}ar^l(L)$, $\mathcal{D}om(\delta) = \mathcal{MV}ar^{nl}(L)$, *and* $\mathcal{D}om(\tau) = M\mathcal{V}ar(L^{lin}) \setminus \mathcal{MV}ar^l(L)$
(3) $u|_p = L^{lin}(\sigma \cup \tau)$ *and* $v|_p = R(\sigma \cup \delta)$
(4) $\forall X^p \in \mathcal{D}om(\tau) : \ \tau(X^p) \underset{<n}{\twoheadrightarrow} \delta(X)$

Proof. 1 follows from **(H1)**. 2 and 3 are clear, 4 is trivial if $L \to R \in R_c$, and follows from **(H4)** and **(H2)** otherwise. □

Lemma 7 (Monotonicity). *Let $s[t]_p$ and $s[u]_p$ be layered terms such that $t \xrightarrow[\mathcal{R},n]{q} u$. Then $s[t] \xrightarrow[\mathcal{R},n]{p \cdot q} s[u]$.*

Proof. Monotonicity of higher-order rewriting and **(H1)**. □

Lemma 8 (Stability). *Let $t \xrightarrow[\mathcal{R},n]{p} u$ and assume that σ is well-layered. Then $t\sigma \xrightarrow[\mathcal{R},\leq n]{p} u\sigma$.*

Proof. By stability of higher-order rewriting and **(H3)**. □

3.5 Sub-rewriting peaks

A sub-rewriting equalization step uses rules from R_c, R_u, as well as the beta rule. The characterization of a sub-rewriting overlapping peak $s \xleftarrow[L \to R]{p} u \xrightarrow[G \to D]{p \cdot q} t$ such that $q <_{\mathcal{P}} F_L$ involves therefore unifying $L|_q$ with G modulo $\mathcal{R} \cup \beta$. In general, there would be no set of most general unifiers and confluence would not be checkable. Our goal is thus to find useful conditions that reduce sub-rewriting overlapping peaks to higher-order critical peaks.

A first condition allowing us to reduce unification of two patterns modulo $\mathcal{R} \cup \beta$ to higher-order unification of those patterns is that the equalization steps from u to s and u to t preserve L^{lin} and G^{lin}. This is the rôle of:

Definition 11 (Linear independence). *A set of rules $\mathcal{R}$ is* linearly independent *if:*
for any two rules $i : L \to R, j : G \to D \in \mathcal{R}$ and $\forall p \in \mathcal{P}os(L)$ such that L^{lin} and G^{lin} are renamed apart and unify at p, then:

(1) *$\forall q \in \mathcal{FP}os(G)$ such that $p \cdot q \geq_{\mathcal{P}} F^{nl}_L$, $G|_q$ is not unifiable with any linearized left-hand side of a rule in $\mathcal{R}$;*
(2) *$\forall p \cdot q \in \mathcal{FP}os(L)$ such that $q \geq_{\mathcal{P}} F^{nl}_G$, $L|_{p \cdot q}$ is not unifiable with any linearized left-hand side of rule in $\mathcal{R}$.*

Any pattern L unifies with a renaming of itself at the root, but in that case, it is easy to see that no check is to be performed. It is important to notice that, since patterns are β-normal and the β-rule is left-linear, $\mathcal{R} \cup \beta$ is linearly independent if so is $\mathcal{R}$.

Example 3. *Let $\mathcal{R} = \{f(X, f(X, Y)) \to a, f(Z, Z) \to b\}$. Checking linear independence requires considering all three possibilities of unifying one (linearized) left-hand side with a subterm of the (linearized) other:*

(1) *$f(X, f(X, Y))$ with itself at position 2, giving the solvable equation $f(X^1, f(X^{2\cdot 1}, Y)) = f(X'^{2\cdot 1}, Y')$. No check is needed here, the condition is satisfied;*
(2) *$f(Z, Z)$ with $f(X, f(X, Y))$ at position 2, giving the solvable equation $f(Z^1, Z^2) = f(X^{2\cdot 1}, Y))$. Again, no check is to be performed.*
(3) *$f(Z, Z)$ with $f(X, f(X, Y))$ at position $\wedge$, giving the solvable equation $f(Z^1, Z^2) = f(x^1, f(X^{2\cdot 1}, Y))$. Then, we need to check whether $f(X, Y)$ is unifiable with any one of the two (linearized) left-hand sides, and indeed, it is with both $f(Z^1, Z^2)$ and $f(X'^1, f(X'^{2\cdot 1}, Y'))$.*

$\mathcal{R}$ is thus not linearly independent, but replacing $f(X, f(X, Y))$ by $f(X, g(X, Y)$ yields a linearly independent set.

Our new assumption is therefore:

- **(H5)** $\mathcal{R}$ is linearly independent.

Another property of linear unification problems is needed. We refer to [8] for the patterns linear unification rules and the associated notion of solved form. These rules differ slightly from usual since our meta-variables have a bounded arity.

Definition 12. *The* linear solved form *$LSF(L, G, p)$ of two patterns L, G at position p in $\mathcal{FP}os(L)$ is the least set of equations containing the solved form of $L^{lin}|_p = G^{lin}$ and closed under* **Merge:** *if $X^0 = u, X^p = v \in LSF(L, G, p)$, then $LSF(u, v, \wedge) \in LSF(L, G, p)$.*

If $L|p$ and G are linear, their linear solved form is of course identical to their solved form. Otherwise, the recursion terminates since the size of (u, v) is strictly smaller than that of (L, G), of which they are strict subterms. Note that $LSF(L, G, p)$ may fail in cases where the solved form of $L^{lin}|_p = G^{lin}$ does not. This failure may originate either from $LSF(u, v, \wedge)$, or from the existence of an occur-check. Lemma 9 shows that the former cannot occur if the equation originates from an overlapping peak. We need ruling out the latter.

Definition 13. *Given two, possibly non-linear patterns L, G and a position $p \in \mathcal{FP}os(L)$ such that $LSF(L, G, p)$ is a solved form $\bigwedge_{i,o} X^o_i = u^o_i$, we define the* occur-check *quasi-order $\geq_{oc}$ between meta-variables of L, G as the least quasi-order generated by the pairs (X, Y) such that $X^o = u[Y^p]_q \in LSF(L, G, p)$. We say that L, G are in* linear occur-check *if there exists a variable $X >_{oc} X$.*

The existence of a linear occur-check implies of course that $L|_p$ and G are not unifiable, the solutions of such equations being infinite rational terms. In that case, our results could probably be extended, following [14], which tackles the very same problem in the simpler contect of first-order rewriting. We therefore add:

- **(H6)** No two rules of R_u are in linear occur-check.

Note that if one of $L|p$ and G is linear, then $L \to R$ and $G \to D$ cannot be in linear occur-check.

A last assumption is common for proofs based on DDs:

- **(H7)** $L \to R$ satisfies VC at $p \in \mathcal{FP}os(L)$ if the context and the subterm of L at p do not share any meta-variable.

Under these assumptions, we can show that overlapping sub-rewriting peaks reduce to instances of critical peaks:

Lemma 9 (Overlapping peak lemma). *Let $\mathcal{R}$ be a rewriting system satisfying assumptions **(H1)** to **(H6)**, and confluent on $\mathcal{L}_{<n}$. Let $s \xleftarrow[L\to R,n]{q} u \xrightarrow[G\to D,m]{q\cdot p} t$ be a closed overlapping sub-rewriting local peak. Then, the equation $L|_p = G$ has most general unifier ρ s.t.*

(1) *$v \xleftarrow[L\to R,n']{\Lambda} L\rho = L\rho[G\rho]_p \xrightarrow[G\to D,m']{p} w$ is a critical peak of $G \to D$ onto $L \to R$ at p*
(2) *$s \underset{<n}{\longrightarrow\!\!\!\!\!\rightarrow} v\delta$ and $t \underset{<n}{\longrightarrow\!\!\!\!\!\rightarrow} w\delta$ for some closed substitution δ ($t \underset{<m}{\longrightarrow\!\!\!\!\!\rightarrow} w\delta$ if $L \to R$ satisfies **(H7)** at p)*

Proof. By monotonicity, we can assume that $q = \Lambda$ wlog. The proof is illustrated at Figure 2.

By Lemma 6, $u = L^{lin}\sigma\tau$ with $\mathcal{D}om(\sigma) \subseteq \mathcal{MV}ar^{nl}(L)$, $\mathcal{D}om(\tau) \subseteq \mathcal{MV}ar^{l}(L)$, $\sigma \underset{<n}{\longrightarrow\!\!\!\!\!\rightarrow} \sigma'$ (by **(H3)**, **(H1)** and **(H2)**), and $u \longrightarrow\!\!\!\!\!\rightarrow L^{lin}(\sigma' \cup \tau) = s' \xrightarrow[L\to R]{\Lambda} R(\sigma' \cup \tau) = s$. By the same token, $u|_p = G^{lin}\gamma\theta$ with $\mathcal{D}om(\gamma) \subseteq \mathcal{MV}ar^{nl}(G)$, $\mathcal{D}om(\theta) \subseteq \mathcal{MV}ar^{l}(G)$, $\gamma \underset{<m}{\longrightarrow\!\!\!\!\!\rightarrow} \gamma'$ and $u|_p \longrightarrow\!\!\!\!\!\rightarrow G^{lin}(\gamma' \cup \theta) = t'|_p \xrightarrow[G\to D]{\Lambda} D(\gamma' \cup \theta) = t|_p$. It follows that $L^{lin}|_p$ and G^{lin} unify with mgu σ. Let H be the term $L^{lin}\sigma = L^{lin}\sigma[G^{lin}\sigma]_p$, a linear pattern by Lemma 4, and F_H its fringe.

All steps in the derivation from u to $L(\sigma' \cup \tau)$ must occur below F_H, since otherwise, the first one falsifying this property would falsify linear independence of $\mathcal{R} \cup \beta$, which follows from **(H5)**. This is true too of the derivation from u to $L(\sigma \cup \tau)[G(\gamma' \cup \theta)]_p$. Hence, s' and t' coincide with H at all positions above F_H, and since H is linear, $s' = H\delta$ and $t' = H\varphi$ for some substitutions δ and φ.

We construct now derivations from s' and from t' to a common reduct u' which is at the same time an instance of H (since H is linear), of L, and of $L[G]_p$, hence unifying $L|_p$ and G. To this end, we need to equalize, for each meta-variable $X \in M\mathcal{V}ar(L) \cup M\mathcal{V}ar(G)$, and for each of its occurrences $X^o \in M\mathcal{V}ar(H)$, the values of $X^o\delta$ and $X^o\varphi$ when they differ, hence defining a term $H\rho = L\rho = L\rho[G\rho]_p$ (we use here the fact that H, L, G have no meta-variable name in common). Note that, despite the fact that we have equalized the non-linear variables only, a linear variable Y may have different values by δ and φ in case it is a variable of, say G, sitting above a non-linear variable X of L in the overlap.

The construction is by induction on the number of meta-variables of H that have a different value by δ and φ. If there is none, we are done. Otherwise, by **(H6)**, we select a maximal subset S of meta-variables of H whose kernels are minimal in the occur-check order. S cannot be reduced to linear meta-variables, since the only way for such a Y to have different values by δ and τ is to depend upon a non-linear meta-variable X, and since Y must be minimal in the occur-check order, then $Y = X$ and X must belong to S by maximality. Therefore, by **(H4)** and **(H3)**, for all variables $X \in S$, $X\delta \in \mathcal{T}_{<n}$ and $X\varphi \in \mathcal{T}_{<m}$. Since all these terms occur at parallel positions, and $m \leq n$ by **(H1)**, we can use our confluence assumption for all those terms at once, hence defining ρ for all variables in S. Note that the derivation from s' (resp., t') operates in layers strictly below n. So does of course the derivation from t' to u', but in case **(H7)** is satisfied, all meta-variables of H that possibly have different values by δ and φ are meta-variables from G^{lin}, hence belong to $\mathcal{T}_{<m}$, and the derivation operates in layers strictly smaller than m. We then conclude the construction by induction hypothesis, yielding $U' = H\rho = L\rho = L\rho[G\rho]_p$, the derivations from s' and t' to u' satisfying the announced layer requirements.

Since u' is an instance of both L and $L[G]_p$ then G unifies with L at p with most general unifier ρ, hence giving rise to the announced critical pair, so that $u' = L\sigma\rho$. We are left with routine calculations for commuting the two derivations originating from s' (resp., t'), and conclude the proof. □

3.6 Orthogonal functional reductions

Tait's proof of confluence for the lambda calculus is based on using what he calls parallel β-reductions, that are called orthogonal β-reductions in [8], a terminology we prefer. Writing a term u as $\underline{u}_K\overline{u}^K$ for some set K of parallel positions of u, the idea is that an orthogonal rewrite step rewrites simultaneously in $\underline{u}_K$ and $\overline{u}^K$. Tait chooses an arbitrary splitting set K, we choose a canonical one. This is a matter of convenience, parallel rewriting with left-linear non-overlapping rules being strongly confluent for all choices.

Definition 14. *Let P be a set of m parallel positions, $K = \{k_i\}_i$ be a set of n positions such that $\forall k \in K\ \exists p \in P :\ k >_{\mathcal{P}} p$, and Q be a set of n sets of positions $\{Q_i\}_i$. Then we define $P \otimes_K Q = P \cup \{k_i \cdot q : i \in [1..n]$ and $q \in Q_i\}$.*

Consider a set of positions P. We can always write $P = O \otimes_K Q$, where $O = P_{min}$, the set of minimal positions in P, and $K = (P \setminus O)_{min}$, Q being then uniquely defined. The triple (O, K, P) is called *canonical splitting* of P.

Definition 15. *Orthogonal β-rewriting the term u to the term v at a set of positions $P \subseteq \mathcal{P}os(u)$, denoted $u \overset{P}{\otimes\!\!\Longrightarrow} v$, is the smallest relation that contains parallel β-rewriting at P when P is a set of parallel positions and that otherwise satisfies the following property: let (O, K, Q) be a canonical splitting of P, then $v = w\theta$, where $\underline{u}_K \overset{O}{\otimes\!\!\Longrightarrow} w$ and $\overline{u}^K \overset{Q}{\otimes\!\!\Longrightarrow} \theta$.*

Orthogonal functional reductions can of course be categorized by levels with the expected notation. We have:

Lemma 10. *Let $P = O \otimes_K Q \subseteq \mathcal{P}os(u)$, $\underline{u}_K \underset{L\to R,n}{\overset{O}{\otimes\!\!\Longrightarrow}} v$ and $\overline{u}^K \underset{L\to R,n}{\overset{Q}{\otimes\!\!\Longrightarrow}} \tau$. Then $u \underset{L\to R,n}{\overset{P}{\otimes\!\!\Longrightarrow}} v\tau$.*

Proof. By induction on $|O|$. □

4 CONFLUENCE

Preparation being now done, we can develop the confluence proof. We first recall the basics of van Oostrom's decreasing diagrams technique, before carrying out that proof by induction on layers. Various properties of local sub-rewriting peaks are considered first, that is, disjoint peaks, ancestor peaks, and critical peaks, before investigating local functional peaks and local heterogeneous peaks.

Our objective will systematically be the same: show that the considered local peak has a decreasing diagram. We will then be ready for the confluence proof itself.

The very first step is to clarify which rewrite relation is used in the confluence proof. Since,

$$\xrightarrow[(\mathcal{R}\beta),n]{} \subseteq \overset{}{\underset{\leq n}{\otimes\!\!\Longrightarrow}} \cup \underset{\mathcal{R},n}{\rhd\!\!\longrightarrow} \subseteq \xrightarrow[(\mathcal{R}\beta),\leq n]{}$$

the rewrite relation in $\lambda\mathcal{F}$ is confluent on $\mathcal{L}$ iff $\bigcup_{n\geq 0} \underset{\leq n}{\otimes\!\!\Longrightarrow} \cup \underset{\mathcal{R},n}{\rhd\!\!\longrightarrow}$ is confluent on $\mathcal{L}$. We will actually use an improved version of the latter relation.

Lemma 11. *$\overset{O}{\underset{\leq n}{\otimes\!\!\Longrightarrow}} \subseteq \overset{P}{\underset{n}{\otimes\!\!\Longrightarrow}} \underset{\beta,<n}{\longrightarrow\!\!\!\!\rightarrow}$ with $P \subseteq O$.*

Proof. Let $u \overset{O}{\underset{\leq n}{\otimes\!\!\Longrightarrow}} v$. By induction on $|O|$. Case $O = \varnothing$ is clear. Otherwise, $O = O_{min} \otimes_K Q$. By **(H1)**, $P = \{o \in O : u|_o \in \mathcal{L}_n\} \cap O_{min} \neq \varnothing$. Hence $u \overset{P}{\underset{n}{\Longrightarrow}} u' \overset{(O_{min}\setminus P)}{\underset{\leq n}{\Longrightarrow}} v$, using **(H3)** for all redexes in u' instantiated by the rewrites at P. By Lemma 10, $u' \overset{P\otimes_K Q}{\underset{\leq n}{\otimes\!\!\Longrightarrow}} v$ We conclude by induction hypothesis. □

So, the rewrite relation we are interested in is actually:

$$\bigcup_{n\geq 0} \underset{\beta,<n}{\longrightarrow\!\!\!\!\rightarrow} \cup \underset{\beta,n}{\otimes\!\!\Longrightarrow} \cup \underset{\mathcal{R},n}{\rhd\!\!\longrightarrow}$$

4.1 Decreasing diagrams

In the following, we consider rewrite relations all of whose steps are equipped with a label in some well-founded set.

Definition 16 (Decreasing diagram [22]). *Given a labeled relation $\longrightarrow$ on an abstract set equipped with a well-founded order $\rhd$, 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 in the order 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, resp.*

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.*

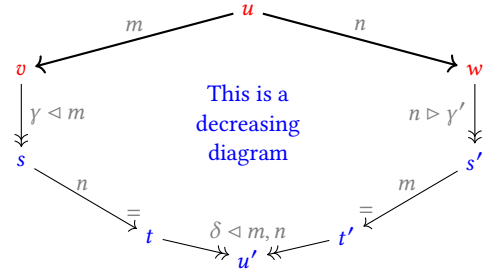

Figure 1: Decreasing diagram

Decreasing diagrams (DDs) are represented in Figure 1. In case a facing step is missing, its preceding side steps are absorbed by the middle ones. A DD is *simple* if it reduces to its middle steps.

Theorem 1 ([22]). *A labeled relation is Church-Rosser if all its local peaks have DDs.*

The careful reader could object that our rewriting relations work modulo $=_\alpha$ on terms, rather than on $=_\alpha$-equivalence classes of terms, and therefore we should have explicit $=_\alpha$-steps, with a minimal label as required in the corresponding extension of van Oostrom's theorem to rewriting modulo [10]. Renaming of bound variables being so specific an equivalence, we allow ourselves this little glitch, as is customary.

Our coming confluence proof is by induction on layers, assuming the relation $\underset{\mathcal{R}\beta,<n}{\longrightarrow}$ is confluent, where $n \neq 0$. Local peaks to be considered are between those three relations mentioned before, and will always involve (at least) one step in layer exactly n.

The label of a rewrite step will be a pair whose first argument is the layer of the redex, and the second, the name of the rule that is used (a technique called rule labeling). Arguments will be compared lexicographically, hence DDs when considering the layer only will again be DDs with pairs as labels. Rule names will be compared in a well-founded order $>_{\mathcal{R}}$, such that the β-rule's name 0 is maximal in $>_{\mathcal{R}}$. Rule names being usually non-zero natural numbers, we feel sorry that this might be felt counter-intuitive.

4.2 Decreasing diagrams for disjoint peaks

Disjoint peaks commute for all monotonic relations, as is the case here of $\bigcup_{n\geq 0} \underset{\beta,<n}{\longrightarrow\!\!\!\!\rightarrow} \cup \underset{n}{\otimes\!\!\Longrightarrow} \cup \underset{\mathcal{R},n}{\rhd\!\!\longrightarrow}$, yielding a diagram reduced to its facing steps, which is therefore decreasing.

4.3 Decreasing diagrams for functional peaks

They are of three kinds that we consider in turn.

Two orthogonal rewrite steps originating from the same term commute [21]: $s \underset{n}{\Longleftarrow\!\!\otimes} u \underset{m}{\otimes\!\!\Longrightarrow} t$ implies $s \underset{\leq m'}{\otimes\!\!\Longrightarrow} u \underset{\leq n'}{\Longleftarrow\!\!\otimes} t$ with $n' \leq n$ and $m' \leq m$ by **(H1)**, **(H3)** and Lemma 11, which is a DD.

The same argument applies to peaks $s \underset{\beta,<n}{\longleftarrow\!\!\!\!\leftarrow} u \underset{n}{\otimes\!\!\Longrightarrow} t$.

The induction hypothesis takes care of $\underset{\beta,<n}{\longrightarrow\!\!\!\!\rightarrow}$-local peaks.

4.4 DDs for ancestor sub-rewriting peaks

We consider now the case where a sub-rewriting rewrite step takes place below some meta-variable X of a rule $L \to R$ used by the other rewrite. These peaks are therefore of two kinds, depending whether the meta-variable X occurs linearly or non-linearly in L.

Lemma 12. *Assume $\underset{\mathcal{R}\beta,<n}{\longrightarrow}$ is confluent, and let $s \underset{\mathcal{R},n}{\lhd\!\!\longleftarrow} u \underset{\mathcal{R},m}{\overset{p\cdot o\cdot q}{\rhd\!\!\longrightarrow}} t$ with $L(o) = X$ being a linear meta-variable of L. Then s and t are joinable by a DD.*

Proof. In that case, $m \leq n$ and the two steps join as any ancestor peak, yielding a DD thanks to **(H4')**. □

Lemma 13. *Assume $\underset{\mathcal{R}\beta,<n}{\longrightarrow}$ is confluent, and let $s \underset{\mathcal{R},n}{\lhd\!\!\longleftarrow} u \underset{\mathcal{R},m}{\overset{p\cdot o\cdot q}{\rhd\!\!\longrightarrow}} t$ with $L(o) = X$ being a non-linear meta-variable of L. Then s and t are joinable by a DD.*

Proof. By monotonicity, we can assume wlog that $p = \Lambda$. By lemma 6, $u = L^{lin}\sigma \xrightarrow[<n]{\geq_{\mathcal{P}} F_L} s' = L\tau \xrightarrow[n]{\Lambda} s = R\tau$ and $L\tau \in \mathcal{L}_n$. By assumption, $X\sigma \xrightarrow[m]{q} X\gamma$. Since X is non-linear, $m < n$ by **(H4)**. By confluence assumption, $X\tau \xrightarrow[<n]{} v \xleftarrow[<n]{} X\gamma$. Let θ be identical to τ except for X for which its value is v. Then, $s = R\tau \xrightarrow[<n]{} R\theta \xleftarrow[k]{\Lambda} L\theta \xleftarrow[<n]{} t$. Since $\tau \longrightarrow \theta$, $k \leq n$ by **(H2)**. We have got a DD. □

4.5 DDs for heterogeneous peaks

We now consider local peaks made of a rewrite step with $\mathcal{R}$ on one side, and functional steps on the other side. Many cases must be considered. We start with $\xrightarrow[\beta,<n]{}$ which is easy:

Lemma 14. *Assume $\xrightarrow[\mathcal{R}\beta,<n]{}$ is confluent, and let $s \xleftarrow[L\to R,n]{p} u \xrightarrow[\beta,<n]{} t$. Then s and t are joinable by a DD.*

Proof. We prove that $s \xrightarrow[\beta,<n]{} v \xleftarrow[L\to R]{p} t$ for some v, a DD, by induction on the number of β-steps. Since they operate in $\mathcal{L}_{<n}$, they cannot occur at positions above p. The induction step has then three cases, the β-step occurring at a position $q\#p$, below a linear variable of L (same proof as for Lemma 12), or below a non-linear variable of L (same proof as for Lemma 13). □

We are left with local peaks involving an orthogonal β-step.

Lemma 15. *Assume $\xrightarrow[\mathcal{R}\beta,<n]{}$ is confluent and let $s \xleftarrow[L\to R,m]{p} u \overset{O}{\underset{k}{\otimes\!\!\Longrightarrow}} t$, with $max(m,k) = n$. There are two different cases that both give a DD:*

- $s \overset{O}{\underset{k}{\otimes\!\!\Longrightarrow}} \xrightarrow[<k]{} \xleftarrow[\mathcal{R},n]{p} t$ *if* $k < n$;
- $s \overset{O}{\underset{n}{\otimes\!\!\Longrightarrow}} \xrightarrow[<n]{} \xleftarrow[\mathcal{R},m]{p} t$ *if* $k = n$.

Proof. Since the pattern L is β-normal and the β-rule is left-linear, the positions in O can only be disjoint from p, below a variable of L, or strictly above p, so that $O = O_\# \uplus O_l \uplus O_{nl} \uplus O_p$. Accordingly, we split the orthogonal step into four, so that $u \overset{O_\#}{\underset{k}{\otimes\!\!\Longrightarrow}} u' \overset{O_l}{\underset{k}{\otimes\!\!\Longrightarrow}} s' \overset{O_{nl}}{\underset{k}{\otimes\!\!\Longrightarrow}} w \overset{O_p}{\underset{k}{\otimes\!\!\Longrightarrow}} t$, and by **(H1)**, $k \geq m$ hence $n = k$ if $O_p \neq \varnothing$.

Disjoint steps commute, hence $s \overset{O_\#}{\underset{k}{\otimes\!\!\Longrightarrow}} s' \xleftarrow[\mathcal{R},m]{p} u'$. The step at p commutes as well with the steps below linear variables of L, hence $s' \overset{O'_l}{\underset{k}{\otimes\!\!\Longrightarrow}} v' \xleftarrow[\mathcal{R},m]{p} v$. O'_l may of course be different from O_l, but $O'_l \# O_\#$ since the former are below p and the latter incomparable to p. As for the next step, we get $v' \overset{O'_{nl}}{\underset{k}{\otimes\!\!\Longrightarrow}} w' \xleftarrow[\mathcal{R},m]{p} w$. Again, O'_{nl} may differ from O_l, but again, $O'_{nl} \# (O_\# \uplus O_l)$, since different variables occur at unrelated positions in a term. The last step is slightly different, since positions in O_p stand above p, hence above $O_l \uplus O_{nl}$ as well as above $O'_l \uplus O'_{nl}$. Since the β-rewrite rule is left-linear but not right-linear in general, we get $w' \overset{O_p}{\underset{k}{\otimes\!\!\Longrightarrow}} t' \xleftarrow[\mathcal{R},m]{p} t$. Formally, we should use splitting here.

Let now $O' = O_\# \uplus O'_l \uplus O'_{nl} \uplus O_p$, Then, $s \overset{O'}{\underset{k}{\otimes\!\!\Longrightarrow}} t'$. Hence $w' \overset{O'}{\underset{n}{\otimes\!\!\Longrightarrow}} t' \xleftarrow[\mathcal{R},m]{p} t$ if $k = n$ and $w' \overset{O'}{\underset{k}{\otimes\!\!\Longrightarrow}} w' \xleftarrow[\mathcal{R},m]{p} t$ if $n > k$. In the former case, the diagram is decreasing because β is maximal in $\triangleright$. In the latter, layers suffice to show decreasingness. □

4.6 DDs for overlapping sub-rewriting peaks

Lemma 9 tells us that any overlapping peak reduces to a closed instance of a critical higher-order peak. To get a decreasing diagram for an overlapping peak, we therefore need to replace the instance of the critical higher-order peak by its corresponding decreasing diagram, assuming it has one. What we can compute, on the other hand, is a joinability diagram for the critical peak, we therefore need to instantiate that diagram before sticking it in, and get a decreasing diagram for the overlapping peak. Instantiating any joinability diagram by a well-layered substitution, however, may not result in a decreasing diagram, even if this joinability diagram *looks* decreasing, since instantiating a rewrite step may change its layer. And, of course, the joinability diagram rewrites open terms, which do not belong to $\mathcal{L}$, its rewrite steps do not have layers. This explains the need for the coming definition:

Definition 17. *Assume that $s \xleftarrow[L\to R]{\Lambda} u \xrightarrow[G\to D]{p} t$ is a critical peak. The joinability diagram $s \xrightarrow[\mathcal{R}\beta]{} \xleftarrow[\mathcal{R}\beta]{} t$ is said to be* stable decreasing *if, for all closed substitutions σ such that $u\sigma$ is layered, $s\sigma \xrightarrow[\mathcal{R}]{} \xleftarrow[\mathcal{R}]{} t\sigma$ is a DD for $s\sigma \xleftarrow[L\to R]{\Lambda} u\sigma \xrightarrow[G\to D]{p} t\sigma$.*

Lemma 16 (Critical peak lemma). *Let $\mathcal{R}$ a rewrite system satisfying **(H1)** to **(H6)** such that $\xrightarrow[\mathcal{R}\beta,<n]{}$ is confluent, all of whose critical peaks are joinable by a stable DD if **(H7)** is not satisfied, otherwise a stable simple DD.*

Then, overlapping sub-rewriting peaks $s \xleftarrow[\mathcal{R},n]{p} u \xrightarrow[\triangleright\mathcal{R},m]{q} t$ are joinable by a DD.

Proof. We can assume wlog that $q \in p \cdot \mathcal{FP}os(L)$. By Lemma 9, there are terms u', v, w such that $s \xrightarrow[<n]{} v$, $t \xrightarrow[<n]{} w$ ($t \xrightarrow[<m]{} w$ if **(H7)** is not satisfied), and $v \xleftarrow[n]{\Lambda} u' \xrightarrow[m]{p} w$ is an instance of the critical pair of $G \to D$ onto $L \to R$ at position p. By assumption and stability, this instance has a DD.

By **(H2)**, $m \leq n$. We assume first that $m < n$.

If **(H7)** is satisfied, then the critical pair's DD has no facing step, and the whole joinability diagram $s \xrightarrow[<n]{} v \xrightarrow[<n]{} \xleftarrow[<n]{} w \xleftarrow[<n]{} t$ is therefore decreasing.

If **(H7)** is satisfied, the DD has a facing step, hence the whole joinability diagram $s \xrightarrow[<n]{} v \xrightarrow[<n]{} \xleftarrow[<n]{} \xleftarrow[n]{} \xleftarrow[<n]{} w \xleftarrow[<m]{} t$ is decreasing again since $m < n$.

The slightly different case $m = n$ is left to the reader. □

4.7 Confluence proof

We are now ready for carrying out the confluence proof. All kinds of local peaks $\bigcup_n \xrightarrow[\beta,<n]{} \cup \overset{}{\underset{\beta,n}{\otimes\!\!\Longrightarrow}} \cup \xrightarrow[\triangleright\mathcal{R},n]{}$ have been considered, and

all have a decreasing diagram with respect to the labeling of elementary proof steps. By van Oostrom's theorem, this relation is confluent on $\mathcal{L}$ if it is confluent on $\mathcal{L}_0$, and therefore, $\underset{\mathcal{R}\beta}{\longrightarrow}$ is confluent on $\mathcal{L}$ iff it is confluent on $\mathcal{L}_0$. This shows the main result of this paper:

Theorem 2. *Let $\mathcal{R} = R_c \cup R_u$ be a set of rewrite rules satisfying our assumptions **(H1)** to **(H6)**. Assume further that all critical peaks $s \underset{L\to R}{\overset{\Lambda}{\longleftarrow}} u \underset{G\to D}{\overset{p}{\longrightarrow}}$ of $\mathcal{R}$ involving at least one rule of R_u have a stable DD if **(H7)** is satisfied or a simple stable DD otherwise, with respect to our order on proof steps. Then, $\lambda\mathcal{F}$ has a confluent rewrite relation $\underset{\mathcal{R}\beta}{\longrightarrow}$ iff R_c defines a confluent rewrite relation on confined terms.*

This result generalizes several other recent results.

Assume that R_u is a set of left-linear rules. Then, two layers of terms are enough, **(H1)**, **(H2)** and **(H3)** are trivially satisfied since all non-confined terms inhabit $\mathcal{L}_1$ and **(H4)** is void. Further, all decreasing diagrams wrt rule-labelling are stable decreasing, since the level of a rewrite step is then determined by the head of the redex, hence invariant by instantiation. Hence [1] is subsumed.

Assume further that R_c is empty. Then the set of confined terms, $\mathcal{L}_0$, is the empty set, and the second layer $\mathcal{L}_1$ is the set $\mathcal{T}$ of all closed terms. This shows that confluence of the pure λ-calculus is preserved when adding left-linear higher-order rewrite rules whose critical pairs have decreasing joinability diagrams with respect to rule labeling, since linear independence **(H5)**, absence of linear occur-checks **(H6)** and Variable Condition **(H7)** are always satisfied by a set of left-linear rewrite rules, as pointed out in Section 3.5, and **(H7)**. Compare this result with a similar one obtained in [8], whose proof requires R_u to be terminating on $\mathcal{T}$, the order on rule names being exactly the same. It generalizes also the confluence result of [15], which is restricted to simply typed λ-terms and a set R_u of terminating rules, apart from the fact that Nipkow's joinability diagrams decrease wrt termination instead of rule labelling.

This shows the strength of Theorem 2, but also the need for getting rid of **(H4')** which may slightly restrict the set $\mathcal{T}$ of confluent terms. Throwing in termination arguments could do.

4.8 Checking for stable decreasingness

In addition to the assumptions **(H1)** to **(H7)**, we have assumed that each critical peak has a joinability diagram that is stable decreasing, that is, all of whose ground instances are decreasing diagrams. This does not come for free. Computing a joinability diagram is easy. Showing it is stable decreasing is a different matter, and will have to be checked one by one. In practice, this is not that hard, and we suggest here sufficient conditions that can be easily implemented.

Let $s \underset{i:L\to R}{\overset{\Lambda}{\longleftarrow}} u \underset{j:G\to D}{\overset{p}{\longrightarrow}} t$ be a critical peak. We assume now that the meta-variables, and therefore the preredexes, in the critical peak have themselves layers, let Γ be a layering assignment for them so that u is layered. Then, by **(H2)**, s and t and all their reducts are layered as well, hence, $\Gamma \vdash s \underset{i,n}{\overset{\Lambda}{\longleftarrow}} u \underset{j,m}{\overset{p}{\longrightarrow}} t$, with $n \geq m$ by **(H1)**.

Consider the case where the joinability diagram has the form $s \underset{k}{\overset{o}{\longrightarrow}} \underset{l}{\overset{q}{\longleftarrow}} t$. Then, $s \underset{k,m'}{\overset{o}{\longrightarrow}} \underset{l,n'}{\overset{q}{\longleftarrow}} t$, and since $u \in \mathcal{L}_n$, $m \geq m'$ and $n \geq n'$ by **(H1)**. For this diagram to be decreasing, either one of the three following conditions is needed: (i)$n > m', n \geq n'$, or (ii) $n = m, n > n', n \geq m'$, or (iii) $n = m = n' = m'$ and $\{i, j\} \rhd_{mul} \{k, l\}$, where $\trianglerighteq_{mul}$ denotes the multiset extension of the order on rule names.

Consider (i) first. By **(H2)**, $n \geq n'$, but $n > m'$ needs to be checked. In practice, this should follow from the layer assignment system assigning layers to the signature as defined in Section 5. Consider now (ii). Again, $n \geq m'$ follows by **(H2)**, and $n > n'$ should be checked. For (iii), it is enough to check the condition on rule names. So, this joinability diagram is decreasing if for every level assignments, either one of the three following conditions is satisfied: (i) $n > m'$ (ii) $n > n'$ (iii) $\{i, j\} \rhd_{mul} \{k, l\}$. Of course, (iii) is appealing since it does not refer to the layers. But it may not always work, so the other two are useful too.

This reasoning can of course be extended to more complex joinability diagrams. In particular, condition (iii) extends easily: if $i_1, \ldots, i_n$ are the rule names occurring in the joinability diagram, $\{i, j\} \rhd \{i_1, \ldots, i_n\}$ ensures stable decreasingness.

5 LAYERING

Layering aims at eliminating terms that break confluence while retaining as many as possible of those that do not. We should further keep in mind that this is meant to be implemented, and should therefore cooperate with our favourite dependent type system, whose aim is also to eliminate (non-typable) terms. We will therefore define layers by means of a simple typing system in order to be easily blended with the dependent type system.

5.1 Term layering

A first step towards a stratification of terms satisfying our assumptions is to have an annotation of variables with the natural number layer they belong to. Since we need to distinguish free from bound variables, we shall write an abstraction as in $\lambda x^n : t.u$, where n is non-zero, and on the other hand declare a free variable x as in $x : n$, as already said. In the latter case, $n = 0$ is possible in order to have variables in confined terms. When taking a subterm of $\lambda x^n : t.u$, we need of course to declare the level n of the variable x which may now occur free in t.

Symbols need be assigned a more complex layering expression so as to allow mixing layers together in a controlled manner. Consider for instance the case of the non-linear rewrite rule $\mathbf{F}(X, X) \to X$. Well-layered instances of the left-hand side of this rule are expected to replace X by terms of a strictly lower level. A layering restriction ensuring this property is $\forall t, u \in \mathcal{L}_{\leq n}, \mathbf{F}(t, u) \in \mathcal{L}_{n+1}$. In the general case, we need annotating symbols $\mathbf{G}$ of arity p with a tuple of $p + 1$ layers $(k_0, \ldots, k_p)$. In order to satisfy **(H1)**, it is of course critical to ensure that the codomain's layer is larger than the domains'.

We define a *number expression* as an algebraic expression built from parameters in a set P, the constant 0, and the operators S, $+$, and $\times$. We use n for $S^n(0)$, and $\leq$ for comparing number expressions, assuming parameters are non-negative.

Definition 18. *A* layer declaration $\mathbf{G} : \forall P.k_0 \to \cdots \to k_n$ *of arity $n = |\mathbf{G}|$, is a $n + 1$-tuple of number expressions such that $\forall i, k_i \leq k_n$, the* output layer. *P is omitted when empty.*

An instance *of the layer declaration $\mathbf{G} : \forall P.k_0 \to \cdots \to k_n$ is a layer declaration $l_0 \to \cdots \to l_n$ such that the number expressions*

$k_1, \ldots, k_n$ evaluate to the natural numbers $l_1, \ldots, l_n$ for some natural number values of the parameters in P. We write $l_0 \to \cdots \to l_n \in \mathbf{G}$.

Definition 19. *A* layering assignment *is made of*

- $\forall x \in \mathcal{X}$*: the layer declaration* $x : n$*;*
- $\forall \mathbf{G} \in \mathcal{F}_c$*: the layer declaration* $\mathbf{G} : 0 \to \cdots \to 0$ *of arity* $|G|$*;*
- $\forall \mathbf{G} \in \mathcal{F}_u$*: a layer declaration of arity* $|G|$*. whose output layer must be strictly positive.*

By definitions of a layer declaration and a layering assignment:

Lemma 17. *Let* $\mathbf{G} \in \mathcal{F}_u$ *such that* $|G| = n$ *and* $l_0 \to \cdots \to l_n \in \mathbf{G}$*. Then,* $l_n > 0$ *and* $\forall i \leq n : l_i \leq l_n$*.*

Definition 20. *Assuming a layering assignment, we define the following layering system assigning terms to layers:*

$$\frac{x : n}{x \in \mathcal{L}_n} \qquad \frac{t \in \mathcal{L}_n \qquad u \in \mathcal{L}_{\leq n}}{(t\ u) \in \mathcal{L}_{\max(n,1)}} \qquad \frac{t \in \mathcal{L}_{\leq n} \qquad u \in \mathcal{L}_{\leq n}}{\lambda x^n : t.u \in \mathcal{L}_n}$$

$$\frac{l_1 \to \cdots \to l_p \to n \in \mathbf{G} \qquad \forall i : u_i \in \mathcal{L}_{\leq l_i}}{\mathbf{G}(u_1, \ldots, u_p) \in \mathcal{L}_n}$$

Layering an abstraction requires of course to add the assignment $x : n$ and remove any previous assignment for x.

Example 4. *Terms of the pure λ-calculus are well-layered, and all belong to layer $\mathcal{L}_n$ if all their variables, whether free or bound, are assigned layer n. λ-terms can also mix layers: assuming A, B are terms inhabiting layer k, the annotated λ-term $\Delta_n = \lambda x^n : A.(x\ x)$ inhabits layer n provided $k \leq n$, and $(\Delta_n\ \Delta_n)$ then inhabits layer n.*

Assuming the layering declaration $\mathbf{F} : \forall n.n \to n+1$, the term $t = \lambda y^{p+1} : A.\ (y\ \mathbf{F}(\lambda x^p : B.x))$ belongs to layer $\mathcal{L}_{p+1}$ for all $A \in \mathcal{L}_{\leq p+1}$ and $B \in \mathcal{L}_{\leq p}$. Consider now the term $\lambda y^0.\mathbf{F}(\mathbf{F}(y))$. Since the variable y belongs to layer 0 by assumption, $\mathbf{F}(y)$ belongs to layer 1, hence $\mathbf{F}(\mathbf{F}(y))$ belongs to layer 2, and therefore, the whole term belongs to layer 2 as well. Let now $\mathbf{G} : \forall n.n \to n+1 \to n+1$. Then, the term $\lambda y^m : \mathbf{G}(\mathbf{F}(y), \mathbf{F}(y))$ belongs to layer $m+2$, since y belongs to layer m, hence $\mathbf{F}(y)$ belong to layer $m+1$ and $m+2$, hence the claim.

The following properties hold for all layering systems:

Lemma 18. (1) *(Confinement)* $\mathcal{L}_0 = \mathcal{T}_c$*;*
(2) *(Subterm) Let* $t \in \mathcal{L}_n$ *and* $t \rhd u$*. Then* $u \in \mathcal{L}_{\leq n}$*;*
(3) *(Stability) Let* $t \in \mathcal{L}_n$ *and* $\sigma \in WLS$*. Then* $t\sigma \in \mathcal{L}_{\leq n}$*;*
(4) *(Beta) Let* $t \in \mathcal{L}_n$ *and* $t \xrightarrow[\beta]{} u$*. Then* $u \in \mathcal{L}_{\leq n}$*.*

Proof. (1) $\mathcal{L}_o = \mathcal{T}_c$ since the only rules that apply are the first one with $n = 0$ and the last one with a layering declaration of the form $\mathbf{G} : 0 \to \ldots \to 0$, which implies that $\mathbf{G}$ is a confined symbol.
(2) By induction on $|t|$ and definition of assignments.
(3) By induction on $|t|$ and definition of a substitution.
(4) By induction on $|t|$. If the β-step occurs in a strict subterm we conclude by induction hypothesis. Otherwise, $t = (\lambda x^n : a.v\ w)$ and necessarily $u, v, w \in \mathcal{L}_{\leq n}$. Since $u = v\{x \mapsto w\}$, and we conclude by (Stability). □

5.2 Properties involving $\mathcal{R}$

We show them first for the example of a single rule rewrite system $\mathcal{R} = \{\mathbf{F}(X, X) \to X\}$ and a layering assignment $\mathbf{F} : 0 \to 0 \to 1$.

Lemma 19. *The overloaded typing system defines layers $\{L_n\}$ that satisfy all properties **(H1)** to **(H7)**.*

Proof.
- **(H0)**, **(H1)**, **(H3)**: by Lemma 18(1,2,3).
- **(H2)**: Holds for β by Lemma 18(4). By monotonicity, it holds for the rule $\mathbf{F}(X, X) \to X$, since any instance $X\sigma$, assuming $\mathbf{F}^k(X, X)\sigma$ is well-layered, inhabits layer $k-1$ thanks to the layering declaration and Lemma 18(3).
- **(H4)**: Assuming $\mathbf{F}^n(X, X')\sigma \in \mathcal{L}_{n+1}$, then $X\sigma \in \mathcal{L}_n$ thanks to the layering declaration. **(H4')** is trivial here.
- **(H5)**, **(H6)** and **(H7)**: All easily checked since the only rule has no non-trivial subterm. □

Theorem 3. *Rewriting with the rule $\mathbf{F}(X, X) \to X$ preserves the confluence of β on well-layered terms generated by the layering declaration $\mathbf{F} : \forall k :\ k \to k \to k+1$.*

Proof. R_c is empty therefore confluent. There is no critical pair in R_u. By Lemma 19, all assumptions **(H1)** to **(H7)** are satisfied. We conclude by Theorem 2. □

To our knowledge, this is the first characterization of a set of terms containing all pure λ-terms for which the rule $\mathbf{F}(X, X) \to X$ preserves confluence of the β-rewrite rule. The only set of terms for which that rule was known to preserve confluence of beta is the set of simply typed lambda terms.

We could of course consider all three rules of the introduction, even together, the result would still hold. The layer declaration for the last rule should be: $\mathrm{c}, \mathrm{u} : \forall k : k \to k+1 \to k+1$

In general, if $\mathbf{G} : x_1^G \to \ldots x_n^G$ is the layer declaration for the function symbol $\mathbf{G}$, where the x_i^G's are variables ranging over natural numbers, finding values for those variables that ensure **(H0,1,2,3,4)** can be easily achieved by solving the constraints that ensure them for all rules in $\mathcal{R}$. In the previous example, the constraint system is just $x_2^F \geq x_0^F \wedge x_2^F \geq x_1^F$ (for **(H1)**), $x_0^F \geq x_0^X \wedge x_1^F \geq x_0^X$ (for **(H1)**), $x_2^F > x_0^X$ (for **(H4)**) and $x_2^F \geq x_0^X$ (for **(H2)**). Note that no constraint is generated for **(H4')** since the left-hand side of rule has no linear variable, but the same technique applies as well when there are some. The minimal solution of this system is $F : 0 \to 0 \to 1$, the one we announced before. When a minimal solution exists as above, **(H5,6,7)** remain to be checked.

5.3 Layering more terms

The set of well-layered terms contains many terms that are not even typable: in particular, all pure λ-terms are well-layered. This includes non-terminating terms such as the ill-typed $(\Delta\ \Delta)$ or fixpoint operators. On the other hand, some terms are excluded that are well-typed. Consider for instance the typing assignment $\mathbf{F} : A \to A \to A$. Then the term $\lambda x : A.\mathbf{F}(x, x)$ is easily typed with $A \to A$, yet no layer assignment for x allows it to be well-layered.

An easy way to extend the results of this paper to a wider set of terms would be to relax the layering inference rules. For instance, variables and function symbols could be assigned arbitrary simple types whose base types are natural numbers. The layering rules

would then be the typing rules of the simply typed λ-calculus which allow us to derive $\lambda x^n{:}A.\mathbf{F}(x,x) : n \to n+1$. Layers could then be defined as the largest base type occurring in a simple type.

In order to keep all terms from the pure λ-calculus, such as Δ, this simple type system should allow arbitrary applications *within* a layer. This can be simply achieved by using the following *conversion rule* between simple types: $n \equiv n \to n$ which allows typing $\Delta : n$.

In this extended setting, the term $s = \lambda z{:}A.\mathbf{F}(\mathbf{F}(z,z),\mathbf{F}(z,z))$ is well-layered (at layer 2) using two differently layered versions for $\mathbf{F}$. However, the term $t = \lambda z{:}A.\ (\lambda y{:}(A \to A).(y\ (y\ z))\ \lambda x{:}A.\mathbf{F}(x,x))$ which β-reduces to s, remains ill-layered.

We have not proved yet that this layering type system contains strictly more terms than the previous one, but we believe it does.

6 CONCLUSION

Our confluence result is the first which goes beyond Klop's unexpected observation that non-left-linear rules break the confluence property of the pure λ-calculus. As a result, most confluence results for untyped lambda calculi assume orthogonality, that is left-linear, critical pair free rewrite rules [16].

Confluence results for typed lambda calculi, such as Nipkow's and Mayr's [15], rely on termination, an approach that does not fit with dependent types. In their presence, existing results target specific sets of rules, either originating from inductive types [3] or satisfying the general schema [4, 11].

Our result answers indeed a new question: on what subset of pure terms is confluence of the pure lambda calculus preserved by rewrite rules whose left-hand sides are non-linear. We have explained why it is a relevant question, provided a theoretical answer, and shown that it can be used in practice.

Higher-order non-left-linear rewrite rules are hard to avoid when embedding rich logical systems into logical frameworks such as Isabelle or Dedukti. The case of polymorphic universes is a paradigmatic example in this respect. A preliminary, partial attempt to encode polymorphic universes was first considered in [1] using a confined level of first-order algebraic expressions encoding the universes which had non-left-linear rules, kept separated from the rest of the embedding system which used left-linear higher-order rules. This paper extends this work by considering arbitrarily many successive layers of higher-order terms on which higher-order rules can operate. It can therefore be used to show confluence of much more complex systems, such as the ones developed in [6].

Besides being new, we believe that our work opens a research path sketched in Section 5, whose goal is to accept more and more layered terms, until, possibly, all closed instances of the left-hand sides of a given non-left-linear system and their reducts pass the test. This is presumably not always possible, and it would then be important to have examples so as to understand why.

It remains to be seen whether confluence on a restricted set of layered terms is enough to guarantee that a type system relying on rewrite rules is well-behaved, that is, large enough to represent all expected objects, such as the image by a translation of a system we want to encode, for example, Coq with polymorphic universes.

Another research path would be to adapt to non-left-linear higher-order rules, confluence criteria developed for left-linear higher-order rules, allowing for example β-rewrite steps to be smaller than the other steps at some (not necessarily all) levels [7]. Although it could be technically delicate, we believe it should work.

Our confluence result assumes two very different sets of assumptions, one for layering, and another for relating sub-rewriting local peaks to rewriting local peaks in Lemma 9, namely **(H5)**, **(H6)** and **(H7)**. Both **(H5)** and **(H7)** are technical properties that one could believe non-essential. **(H7)** occurs in many other confluence result when first-order rewriting may be non-terminating, there is therefore no wonder that it pops up again here. **(H5)** plays a crucial role in Lemma 9, we do not see any way to get around it, but have no strong argument to assert its necessity. **(H6)** ensures that critical pairs are finite terms, instead of infinite rational terms. The same problem occurs already with first-order rewriting, and is successfully solved in [14]. We believe that the same techniques should work with higher-order rules whose left-hand sides are patterns, at the price of developing a unification algorithm for higher-order rational patterns, which should now be easy. **(H6)** would then disappear. Finally, we also believe that **(H4')** could be dropped by using orthogonal rewriting for higher-order rules. Then, β-reduction would just be yet another higher-order rule.

REFERENCES

[1] A. Assaf, G. Dowek, J.-P. Jouannaud and J. Liu. Untyped confluence in dependent type theories. draft hal-01515505, INRIA, january 2018. presented at HOR 2016.

[2] A. Assaf, G. Burel, R. Cauderlier, G. Dowek, C. Dubois, F. Gilbert, P. Halmagrand, O. Hermant, and R. Saillard. Dedukti: a Logical Framework based on the lambda-pi-Calculus Modulo Theory. draft, INRIA, 2019.

[3] B. Barras, P. Corbineau, B. Grégoire, H. Herbelin, and J. L. Sacchini. A new elimination rule for the calculus of inductive constructions. In S. Berardi, F. Damiani, and U. de'Liguoro, editors, *Types for Proofs and Programs, International Conference, TYPES 2008, Torino, Italy, March 26-29, 2008, Revised Selected Papers*, volume 5497 of *Lecture Notes in Computer Science*, pages 32–48. Springer, 2008.

[4] F. Blanqui. Definitions by rewriting in the calculus of constructions. *Mathematical Structures in Computer Science*, 15(1):37–92, 2005.

[5] J. Cockx, N. Tabareau, and T. Winterhalter. The taming of the rew: a type theory with computational assumptions. *Proc. ACM Program. Lang.*, 5(POPL):1–29, 2021.

[6] G. Férey. *Higher-Order Confluence and Universe Embedding in the Logical Framework*. PhD thesis, Université Paris-Saclay, Orsay, France, 2021. submitted.

[7] G. Dowek, G. Férey, J.-P. Jouannaud and J. Liu. Confluence of Left-Linear Higher-Order Rewrite Theories by Checking their Nested Critical Pairs. draft hal-03126111, INRIA, january 2021. Presented at HOR 2016. submitted to MSCS.

[8] G. Férey and J.-P. Jouannaud. Confluence in UnTyped Higher-Order Theories by Means of Critical Pairs. draft. hal-03126102, INRIA, march 2021. Under revision for TOCL, available from http://dedukti.gforge.inria.fr/.

[9] H. Goguen. The metatheory of UTT. In P. Dybjer, B. Nordström, and J. 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.

[10] J.-P. Jouannaud and J. Liu. From diagrammatic confluence to modularity. *Theor. Comput. Sci.*, 464:20–34, 2012.

[11] J.-P. Jouannaud and M. Okada. A computation model for executable higher-order algebraic specification languages. In *Proceedings of the 6th annual IEEE Symposium on Logic in Computer Science (LICS '91)*, pages 350–361, Amsterdam, The Netherlands, July 1991.

[12] J. W. Klop. *Combinatory Reduction Systems*. Number 127 in Mathematical Centre Tracts. CWI, Amsterdam, The Netherlands, 1980. PhD Thesis.

[13] J. Liu, N. Dershowitz, and J.-P. Jouannaud. Confluence by critical pair analysis. In G. Dowek, editor, *Rewriting and Typed Lambda Calculi - Joint International Conference, RTA-TLCA 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, volume 8560 of *Lecture Notes in Computer Science*, pages 287–302. Springer, 2014.

[14] J. Liu, J.-P. Jouannaud, and M. Ogawa. Confluence of layered rewrite systems. In S. Kreutzer, editor, *24th EACSL Annual Conference on Computer Science Logic, CSL 2015, September 7-10, 2015, Berlin, Germany*, volume 41 of *LIPIcs*, pages 423–440. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

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

[16] A. Middeldorp, V. van Oostrom, F. van Raamsdonk, and R. C. de Vrijer, editors. *Processes, Terms and Cycles: Steps on the Road to Infinity, Essays Dedicated to Jan*

Willem Klop, on the Occasion of His 60th Birthday, volume 3838 of *Lecture Notes in Computer Science*. Springer, 2005.

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

[18] T. Nipkow. Higher-order critical pairs. In *Proceedings of the 6th annual IEEE LICS Symposium*, pages 342–349, Amsterdam, The Netherlands, July 1991.

[19] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. 2002.

[20] M. Okada. Strong normalizability for the combined system of the typed lambda calculus and an arbitrary convergent term rewrite system. In G. H. Gonnet, editor, *Proceedings of the ACM-SIGSAM 1989 Symposium on Symbolic and Algebraic Computation, ISSAC '89, Portland, Oregon, USA, 1989*, pages 357–363. ACM, 1989.

[21] Terese. Term rewriting systems. In *Cambridge Tracts in Theoretical Computer Science, M. Bezem, J. W. Klop & R. de Vrijer eds.* Cambridge University Press, 2003.

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

7 APPENDIX

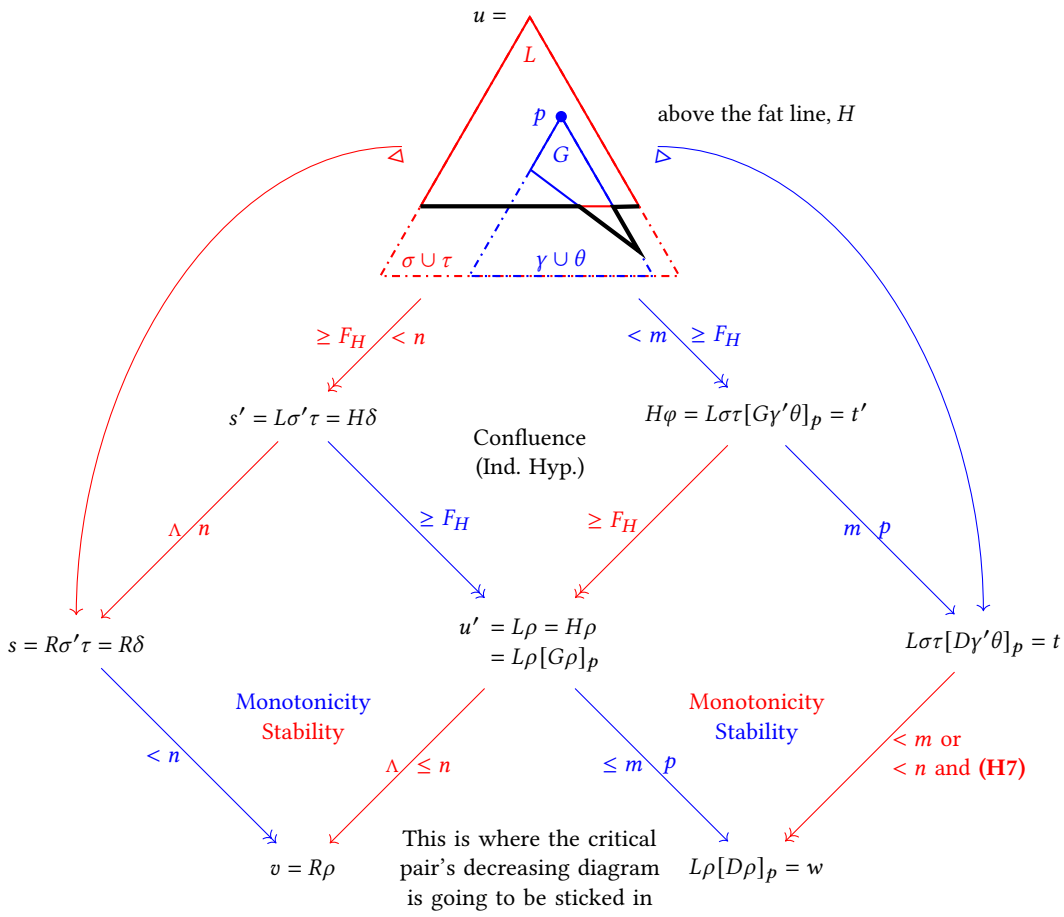

Figure 2: Proof structure of overlapping peak Lemma 9