Technical report corresponding to paper accepted for the International Conference on Rewriting Techniques and Applications
http://rewriting.loria.fr/rta/

HIGHER-ORDER (NON-)MODULARITY

CLAUS APPEL [1] AND VINCENT VAN OOSTROM [2] AND JAKOB GRUE SIMONSEN [1]

[1] Department of Computer Science, University of Copenhagen (DIKU)
Universitetsparken 1, 2100 Copenhagen Ø
Denmark
E-mail address, C. Appel: `spectrum@diku.dk`
E-mail address, J. G. Simonsen: `simonsen@diku.dk`

[2] ZENO Research Institute, Department of Philosophy, Utrecht University
Heidelberglaan 8, 3584 CS Utrecht
The Netherlands
E-mail address: `Vincent.vanOostrom@phil.uu.nl`

Abstract. We show that, contrary to the situation in first-order term rewriting, almost none of the usual properties of rewriting are modular for higher-order rewriting, irrespective of the higher-order rewriting format. For the particular format of simply typed applicative term rewriting systems we prove that modularity of confluence, normalization, and termination can be recovered by imposing suitable linearity constraints.

1. Introduction and summary of results

The *disjoint union* of two rewrite systems is the rewrite system obtained by taking the disjoint union of their signatures and taking the union of their respective sets of rules. *Modularity* is the study of properties preserved and reflected when taking the disjoint union of two rewrite systems and has been studied intensely for first-order term rewriting systems.

Higher-order term rewriting adds two features to first-order term rewriting: meta-variables and binding. Meta-variables are variables for functions, *i.e.* they can be *applied*. Binding allows to construct functions by means of *abstraction*. There are a plethora of formats of higher-order rewriting, spanning the gap from very specific to very general systems. In this report we consider the following common formats: *Applicative* TRSs [25, Section 3.3.5], contain meta-variables, but no bound variables. The prototypical example of an applicative TRS is Curry's combinatory logic. Equipping applicative TRSs with a simple type discipline results in Yamada's *simply typed term rewriting systems* (STTRSs) [29]. Klop's (functional) *combinatory reduction systems* (CRSs) [11], contain both meta-variables and bound variables. The prototypical example of a CRS is Church's λ-calculus. Equipping CRSs with a simple type discipline (and generalizing the notion of substitution), results in Nipkow's *pattern rewrite systems* (PRSs) [14]. We have chosen these formats because

1998 ACM Subject Classification: F.4.1, F.4.2.
Key words and phrases: Higher-order rewriting, modularity, termination, normalization.

they appear to be the most relatively well-known, and because they allow for the 'free' construction of rules, without a priori restrictions ensuring termination or confluence.

Modularity in higher-order systems has hitherto only been investigated in isolated cases; Klop proved that confluence is not a modular property in systems that can embed both TRSs and λ-calculus [9], and Klop, van Oostrom and van Raamsdonk showed that acyclicity (a term cannot be reduced to itself) of orthogonal systems, while modular for TRSs, is not a modular property of higher-order systems [10].

In this report we perform the first systematic study of modularity for higher-order rewriting, see the overview in Table 1. The only non-standard notion employed in the table, is the notion of pattern.

Table 1: Modular properties of first- and higher-order term rewriting system. The results marked (†) are new and proved in this report.

Property	**TRS**	**STTRS**	**CRS**	**PRS**
Confluence	Yes	No	No	No
Normalization	Yes	No (†)	No (†)	No (†)
Termination	No	No	No	No
Completeness	No	No	No	No
Confluence, for left-linear systems	Yes	Yes	Yes	Yes
Completeness, for left-linear systems	Yes	No (†)	No (†)	No (†)
Unique normal forms	Yes	No (†)	No (†)	No (†)
Normalization, non-duplicating pattern systems	Yes	Yes (†)	?	?
Termination, non-duplicating pattern systems	Yes	Yes (†)	?	No (†)

Definition 1.1. A left-hand side of a rule is a *pattern* if all meta-variables in it are only applied to sequences of pairwise distinct bound variables. A *pattern* rewrite system is one in which all left-hand sides of rules are patterns.

For applicative TRSs and STTRSs the restriction to patterns expresses that meta-variables do not occur *actively* in left-hand sides, *i.e.* left-hand sides do not have sub-terms of shape $Z\,t$, for Z a meta-variable. Combinatory Logic and all applicative TRSs obtained by Currying are pattern systems. CRSs and PRSs have the condition that left-hand sides of rules be patterns, built into their definition, making matching and unification of left-hand sides first-order like.

The structure of the report is as follows. We first recapitulate the main positive and negative modularity results from first-order term rewriting, as well as the techniques employed for obtaining them. Next we show by means of a slate of counterexamples, that none of the standard rewriting properties is modular, neither for applicative TRSs, nor for CRSs and PRSs. We end on a positive note, showing that imposing appropriate linearity restriction allows one to regain modularity of some properties, in particular confluence, termination and normalization of STTRSs.

We classify the properties discussed into existence (a normal form can/must/cannot be obtained) and uniqueness (at most one normal form can be obtained) properties. Termination, normalization and acyclicity are existence properties, and confluence and the unique normal forms property are uniqueness properties. Completeness combines both into a unique existence property.

As presenting the counterexamples requires much less technical machinery than the positive results, we postpone the introduction of that machinery to the section containing those positive results. For now, we assume the reader to be familiar with the basic notation for first-order term rewriting systems (TRSs), with simple types, and with the notion of bound variables [3, 25]. This should be sufficient to understand the underlying phenomena, although familiarity with the formats of higher-order term rewriting we treat is an advantage. We refer the reader to [25, Section 3.3.5] for the definition of applicative TRSs, to [29] for STTRSs, to [9, 11, 21] for CRSs, and to [21, 14] for PRSs. Furthermore, we assume the reader to be familiar with the concept of orthogonality in first-order rewriting and its (straightforward) extension to higher-order rewriting; on several occasions we shall use the fact that orthogonal first/higher-order term rewriting systems are confluent [9, 16, 22].

Throughout the report, we let Σ denote a (first- or higher-order) signature, and $\mathcal{T}$ denote a (first- or higher-order) rewriting system; we equip both of these with integer subscripts when more than one signature or system is needed. We denote by $A \uplus B$ the disjoint union of sets A and B, and we denote by $\mathcal{T}_0 \oplus \mathcal{T}_1 = (\Sigma_0 \uplus \Sigma_1, R_0 \uplus R_1)$ the disjoint union of the rewrite systems $\mathcal{T}_i = (\Sigma_i, R_i)$ for $i \in \{0,1\}$. A property P of a class $\mathcal{C}$ of rewrite systems is *modular* if $P(\mathcal{T}_0 \oplus \mathcal{T}_1) \Leftrightarrow P(\mathcal{T}_0) \,\&\, P(\mathcal{T}_1)$ for all $\mathcal{T}_0, \mathcal{T}_1 \in \mathcal{C}$.[1] We employ x, y, z to range over variables for terms of base type, and Z, W, X to range over meta-variables, *i.e.* variables which yield a term of base type when supplied with sufficiently many terms of the appropriate types. When appropriate, we underline the redex contracted in a rewrite step. Finally, we employ standard rewriting notation as given in [25].

Status. This is the technical report corresponding to a paper with the same title and authors to be published in the proceedings of the 21^{st} International Conference on Rewriting Techniques and Applications, Edinburgh, UK, July 11–13, 2010, and published by LIPIcs (Leibniz International Proceedings in Informatics). Except for correcting minor issues the difference is that the full proofs, which had to be omitted from the paper because of space–time limitations, are included in the present report.

1.1. Modularity in first-order rewriting

The study of modularity in term rewriting was essentially introduced by Toyama in two seminal papers showing, respectively, that confluence is modular for TRSs [27], but that termination is not [26]. Since then, modularity of various properties has been investigated, *e.g.*, normalization (easily seen to be modular, see also [12]), the unique normal forms property (modular [15]), unique normal forms with respect to reduction (not modular [15]), completeness (not modular [26]). For the non-modular properties, restrictions (*e.g.*, left- and/or right-linearity, non-collapsingness) have been put forth that ensure modularity, see for example [23, 28, 13, 24, 17]). Furthermore, modularity has been considered for several varieties of first-order rewriting, and new proofs have been given for modularity of confluence, *cf.* [20] and its references.

To set the stage for the rest of the report, we recapitulate Toyama's classical counterexample to modularity of termination for TRSs.

[1] An easy consequence of the fact that reduction steps in $\mathcal{T}_i$ can be embedded as reduction steps in $\mathcal{T}_1 \oplus \mathcal{T}_2$, is that each property P studied in this report holds for $\mathcal{T}_i$ if it holds for $\mathcal{T}_1 \oplus \mathcal{T}_2$. Hence our focus is exclusively on (dis)proving $P(\mathcal{T}_0 \oplus \mathcal{T}_1) \Leftarrow P(\mathcal{T}_0) \,\&\, P(\mathcal{T}_1)$.

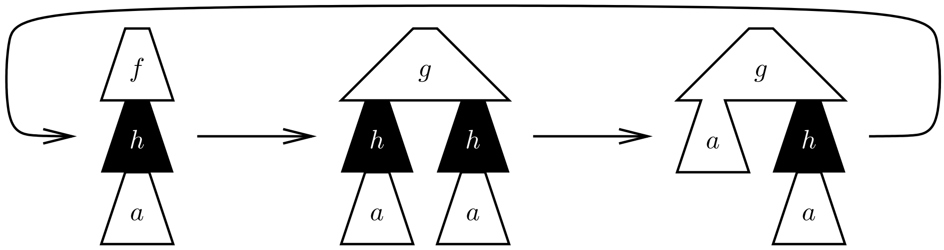

Figure 1: First-order systems: Redex creation can occur by duplicating or collapsing steps. The *rank*, the maximum number of signature changes on a path from the root to a leave in a term, *cannot* increase along reduction. Here, the "white" system is $R_0 = \{f(x) \to g(x,x), g(a,x) \to f(x)\}$ and the "black" system is $R_1 = \{h(y) \to y\}$ (cf. Counterexample 1.2).

Counterexample 1.2. The single rule TRS

$$f(a,b,x) \to f(x,x,x)$$

is easily proved to be terminating. However, when taking its disjoint union with the–also trivially terminating–two-rule TRS

$$\begin{array}{rcl} g(x,y) & \to & x \\ g(x,y) & \to & y \end{array}$$

we obtain a non-terminating system as witnessed by the cycle:

$$\underline{f(a,b,g(a,b))} \to f(\underline{g(a,b)}, g(a,b), g(a,b)) \to f(a, \underline{g(a,b)}, g(a,b)) \to f(a,b,g(a,b))$$

Intuitively, termination of the first TRS above relies on the *absence* of a term which reduces both to a and b; a property destroyed by the second TRS by its ability to encode non-deterministic choice.

Toyama's counterexample above holds true for any higher-order format embedding TRSs and their rewrite relation, in particular the formats considered in this report.

The main proof technique for establishing modularity results for first-order TRSs is based on terms in the disjoint union of TRSs being *stratified* in the sense that each term in the disjoint union has a unique decomposition into layers of components residing in either of the TRSs separately, and moreover that this stratification is *preserved* by rewriting in the sense that the *rank*, *i.e.* the number of layers, cannot increase along a reduction (Figure 1). In the higher-order case, preservation fails due to the presence of rules in which meta-variables can be applied to each other, which allow for *nesting* in the rhs of rules, whence the rank may *increase* along a reduction (Figure 2).

2. Counterexamples to modularity for applicative TRSs

In this section we set the stage for our positive modularity results for applicative TRSs in Sections 5 and 6. This we do by analysing known obstacles for obtaining such results, *cf.* [8], from the perspectives of simply typed STTRS and of pattern rules.

Applicative TRS can be embedded into ordinary (functional) TRSs by viewing the symbols from their signature as nullary function symbols and adjoining one binary function symbol for application. Therefore, one might naïvely expect the modularity results for

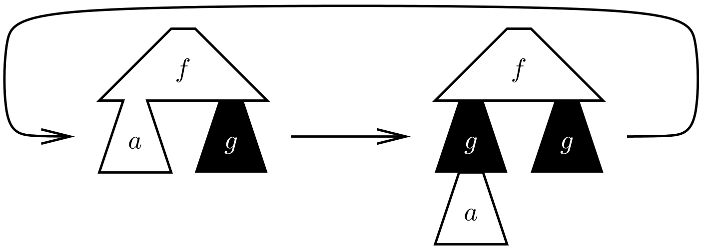

Figure 2: Higher-order systems: Redex creation can *also* occur by application. The rank (measured via the number of signature changes of head symbols of sub-terms) *may* increase along a reduction. Here, the "white" system is $R_0 = \{f\,a\,Z \to f\,(Z\,a)\,Z\}$ and the "black" system is $R_1 = \{g\,W \to W\}$ (cf. Counterexample 2.5).

TRSs to carry over to applicative TRSs. In fact, they do not, the reason being the change in status of the application symbol from being implicit in applicative TRSs to being an explicit element of the signature in their embedding; that is, the embedding of the disjoint union of two applicative TRSs is distinct from the disjoint union of their embeddings [8].

Remark 2.1. To prove, say, confluence of the disjoint union of the confluent applicative TRSs $\{f\,Z \to Z\}$ and $\{g\,W \to W\}$, one might also proceed as follows, *cf.* [25, Section 3.3.5]:
(1) *Uncurrying* yields the confluent functional TRSs $\{f(x) \to x\}$ and $\{g(y) \to y\}$;
(2) Modularity [27] yields confluence of the disjoint union $\{f(x) \to x, g(y) \to y\}$; and
(3) Preservation by *currying* [7] yields confluence of $\{f\,Z \to Z, g\,W \to W\}$, as desired.
The main obstacle following this route is that typically rules of applicative TRSs do contain active (higher-order) variables (this can be seen as the raison d'être of applicative TRSs) and such rules cannot be in the image of the currying transformation.

Uniqueness properties. Unlike what is the case for first-order TRSs, confluence is not a modular property of applicative TRSs as famously shown by Klop [9, Theorem III.1.2.12] who considered the disjoint union of combinatory logic and the applicative TRS $\{D\,Z\,Z \to Z\}$. In order to obtain our positive result of Section 5 we provide some further counterexamples and identify possible causes of non-modularity.

The confluence claims in the counterexamples below are readily verified by standard TRS theory (orthogonality respectively termination and critical pair criteria) applied to the embedding of the applicative TRSs. In our first counterexample the role of combinatory logic in Klop's example is taken over by the μ-rule, directly modeling recursion instead of encoding it via a fixed-point combinator in combinatory logic.

Counterexample 2.2. Taking the disjoint union of the confluent applicative TRSs $\{\mu\,Z \to Z\,(\mu\,Z)\}$ and $\{f\,W\,W \to a, f\,W\,(s\,W) \to b\}$ yields a non-confluent system as witnessed by $a \leftarrow f\,(\mu\,s)\,(\mu\,s) \to f\,(\mu\,s)\,(s\,(\mu\,s)) \to b$.

Counterexample 2.3. The disjoint union of the confluent applicative TRSs $\{g\,(Z\,W) \to g\,W\}$ and $\{h\,a \to b\}$ is non-confluent as witnessed by $g\,a \leftarrow g\,(h\,a) \to g\,b$.

Assigning types as $a, b, W : o,\ Z, g, h, s : o \to o,\ \mu : (o \to o) \to o$, and $f : o \to o \to o$ shows that the applicative TRSs in both counterexamples are in fact STTRSs, which entails that confluence is not modular for STTRSs. However, note that the first counterexample employs a non-left-linear rule (*e.g.*, the left-hand side $f\,W\,W$) and the second example a non-pattern rule (with left-hand side $g\,(Z\,W)$). In Section 5 we show that confluence *is* modular for left-linear pattern STTRSs.

The same counterexamples show that the unique normal forms property is not modular for applicative TRSs and STTRSs.

Remark 2.4. The problematic nature of non-pattern rules in applicative TRSs, *i.e.* rules which contain active variables as in the first rule in Counterexample 2.3, is well-known. For instance, adjoining to combinatory logic or the λ-calculus a combinator A defined by $A\,(Z\,W) = Z$, *i.e.* which extracts the function from a function application, immediately renders these calculi inconsistent in the sense that all terms become convertible [4].

Existence properties. Toyama's Counterexample 1.2 to modularity of termination for TRSs carries over immediately to applicative TRSs and even STTRSs, as the terminating TRSs involved can be viewed as terminating STTRSs by assigning appropriate types to the function symbols, *e.g.*, the type $o \to o \to o \to o$ to the ternary function symbol f. But unlike the TRS case also normalization fails for applicative TRSs, under various restrictions, caused by the possibility to apply meta-variables.

In Section 6 we will show that termination and normalization *are* modular for applicative non-duplicating, typable, pattern TRSs. Here we show that any two of them are not sufficient for modularity of termination. The termination claims in the counterexamples below are readily verified by current automated termination tools.

Counterexample 2.5. Taking the disjoint union of the applicative (duplicating) typable pattern TRSs $\{f\,a\,Z \to f\,(Z\,a)\,Z\}$ and $\{g\,W \to W\}$ enables the non-normalizable reduction $f\,a\,g \leftrightarrow f\,(g\,a)\,g$ despite both TRSs being normalizing (even terminating, note that no redex-creation is possible in either of them).

The same example provides a counterexample to modularity of (left-linear, orthogonal) termination, acyclicity, and completeness.

Counterexample 2.6. Taking the disjoint union of the applicative left-and-right-linear (non-typable) pattern TRSs $\{f\,Z\,W \to Z\,a\,f\}$ ('left rotation') and $\{g\,Z\,W \to W\,g\,b\}$ ('right rotation') enables the non-normalizable reduction $f\,g\,b \leftrightarrow g\,a\,f$ despite both TRSs being normalizing (even terminating, based on the insertion of a and b in their 'rotations').

The same example provides a counterexample to modularity of (left-and-right-linear, orthogonal) termination, acyclicity, and completeness.

Counterexample 2.7. Taking the disjoint union of the applicative left-and-right-linear typable (non-pattern) TRSs: $\{f\,(Z\,W) \to Z\,(a\,f)\}$ and $\{g\,(X\,Y) \to Y\,(g\,b)\}$ enables the non-normalizable reduction $f\,(g\,b) \leftrightarrow g\,(a\,f)$ despite both TRSs being normalizing (even terminating, based on the same idea as in the previous counterexample). The TRSs are seen to be typable by assigning types as: $W, b : o$, $Z, f, Y, g : o \to o$, and $a, X : (o \to o) \to o$.

The same example provides a counterexample to modularity of (left-and-right-linear, typable) termination and acyclicity.

3. Counterexamples to modularity for functional CRSs

For TRSs and PRSs, which will be treated in the next section, in general properties are preserved under signature extensions, *i.e.* are modular when one of the rewrite systems has no rules at all. The basic idea is to replace each fresh function symbol (from the other signature) by a variable of identical type: If a property does not hold with fresh function symbols, it does not hold with all fresh function symbols replaced by fresh variables, a contradiction. Obviously, this idea fails when the replacement of function symbols by variables is, for some reason, impossible. In particular, for functional CRSs the replacement is impossible in the absence of meta-variables in terms. More precisely, the rewrite relation of a functional CRS was defined in [11] as a relation on terms *without* meta-variables. As we show in this section, this causes that most properties are not even preserved under signature extensions.

Remark 3.1. The restriction of the rewrite relation to terms without meta-variables is analogous to the restriction of the rewrite relation of first-order TRSs to ground terms. From that perspective, the failure of modularity in case of signature extensions is unsurprising (*e.g.*, normalization is not modular w.r.t. the ground rewrite relation of TRSs; to wit $\{f(a)\to a, f(x) \to f(x)\}$ is ground normalizing but not so when the signature is extended with a constant b). We view our results below in a positive way, as suggesting to change the definition of the rewrite relation of a CRS to include meta-terms, having meta-variables, cf. [11, 5].

Uniqueness properties.

Counterexample 3.2. The CRS given by the rules

$$\begin{array}{rcl} f(f(W)) & \to & f(W) \\ f([x]Z(x)) & \to & f(Z(a)) \\ f([x]Z(x)) & \to & f([x]Z(Z(x))) \end{array}$$

can be shown to be confluent (an easy induction on terms, see the appendix), but is not so after extending the signature with a unary g:

$$f(g(a)) \leftarrow f([x]g(x)) \to f([x]g(g(x))) \to f(g(g(a)))$$

showing non-preservation of confluence.

In effect, the counterexample shows that a CRS can be confluent, *i.e.* confluent on terms, but not *meta*-confluent, *i.e.* not confluent on *meta*-terms. This is analogous to the fact that a TRS can be *ground* confluent, but not confluent.

By the same example it follows that the unique normal forms property is not preserved under signature extension either.

Existence properties. For TRSs, termination is preserved under signature extension, as follows by an easy induction on the rank of terms as fresh function symbols partition any term in the disjoint union into terminating components. For PRSs, termination is preserved under signature extension as explained above. For CRSs both of these methods fails, the former because of the lack of an appropriate notion of rank, and the latter because of the absence of fresh meta-variables.

Counterexample 3.3. The CRS having a single, unary function symbol f, and rule

$$f([x][y]Z(x,y)) \quad \to \quad Z([x][y]Z(y,x),[x]x)$$

is terminating, as can be shown by induction on terms (noting that the rewrite relation for CRSs is defined on *terms* not on *meta-terms*, termination of the CRS is shown by an easy induction on terms using that a term may contain *at most one* bound variable, in the absence of function symbols having arity greater than one, see the appendix). However, extending the signature with a binary symbol g allows to 'swap the roles of x and y'. For instance:

$$\begin{array}{rcl} f([x][y]g(f(x),f(y))) & \to & g(f([x][y]g(f(y),f(x))),f([x]x)) \\ & \to & g(g(f([x]x),f([x][y]g(f(x),f(y)))),f([x]x)) \end{array}$$

The reduction has shape $t \twoheadrightarrow g(g(f([x]x),t),f([x]x))$ for $t = g(g(f([x]x),t),f([x]x))$, giving rise to the *spiraling* reduction:

$$t \twoheadrightarrow g(g(f([x]x),t),f([x]x)) \twoheadrightarrow g(g(f([x]x),g(g(f([x]x),t),f([x]x))),f([x]x)) \twoheadrightarrow \ldots$$

showing non-preservation of termination. Note that it is essential for non-termination that $Z(x,y)$ is instantiated by a term containing both x and y, something impossible without function symbols of arity more than 1.

As the CRS is orthogonal and non-erasing, it is terminating iff it is normalizing, whence normalization is not preserved under signature extension either.

Both for TRSs and PRSs, left-linear completeness is preserved under signature extension. For TRSs this is just a special case of modularity of left-linear completeness [28, 24]. For PRSs, it follows by replacing fresh function symbols with fresh variables as explained above. For CRSs, left-linear completeness is not preserved under signature extension: The CRS in Counterexample 3.3 is orthogonal, hence left-linear and confluent, and is terminating, hence complete. However, it is not terminating after adding the fresh symbol g.

4. Counterexamples to modularity for PRSs

In this section we present counterexamples to modularity for Nipkow's pattern rewrite systems.

Since the terms of STTRSs can be embedded directly into PRSs, one might naïvely expect the counterexamples against modularity for STTRSs of Section 2 to carry over to PRSs. In fact, they do not, the reason being the possible presence of abstractions in PRS terms and the ensuing difference in substitution (of higher-order terms).

Example 4.1. As shown in Counterexample 2.5, the system $\{f\,a\,Z \to f\,(Z\,a)\,Z\}$ is terminating when viewed as an STTRS, but not so when viewed as a PRS. To wit, instantiating the meta-variable Z in the rule to $x.x$ yields the infinite looping reduction:

$$f\,a\,(x.x) \to f\,a\,(x.x)$$

Also, PRSs do, unlike CRSs, allow for function variables in terms, hence the counterexamples against modularity for CRSs of Section 3 based on signature extension, do not carry over to PRSs either.

Uniqueness properties. Klop showed in [9] that confluence is not a modular property for CRSs. In particular, his counterexample [9, Theorem III.1.2.10] involves (i) the non-left-linear first-order rule $\{D\,Z\,Z \to Z\}$, and (ii) the β-rule of the λ-calculus.

Similar to what we did in the case of applicative TRSs (Counterexample 2.2), we recast Klop's example as a PRS replacing λ-calculus by the μ-rule, directly modeling recursion instead of encoding it via the fixed-point combinator in the λ-calculus.

Counterexample 4.2. The first-order TRS consisting of the following two rules

$$\begin{array}{rcl} f(x,x) & \to & a \\ f(x,s(x)) & \to & b \end{array}$$

is terminating and has no critical pairs, hence is confluent by Huet's Critical Pair Lemma [6].

However, taking the disjoint union with the orthogonal—hence confluent—single-rule PRS $\mu(x.Z(x)) \to Z(\mu(x.Z(x)))$ yields a non-confluent system as witnessed by:

$$a \leftarrow f(\mu(x.s(x)),\mu(x.s(x))) \to f(\mu(x.s(x)),s(\mu(x.s(x)))) \to b$$

Intuitively, confluence of the TRS above relies both on termination and on the absence of a critical pair involving the two rules, which in turn relies on non-left-linearity and non-convertibility of t and $s(t)$ for any term t. Both of those features are destroyed by the PRS above due to its ability to encode recursion, as witnessed by taking $t = \mu(x.s(x))$.

The unique normal forms property is not modular for PRSs as shown by the same example employed above: As the rewrite systems are confluent they both have the unique normal forms property, but the terms a and b are distinct convertible normal forms in the disjoint union of the TRS and the PRS.

Existence properties. Left-linear completeness is modular for TRSs [28, 24], but fails to be so for PRSs.

Counterexample 4.3. Consider the PRS consisting of the single rule $f(x.x, xy.Z(x,y)) \to g(Z(a, f(x.Z(x,a), xy.Z(x,y))))$ where f and g are second-order symbols and a is a first-order symbol. The PRS is orthogonal, hence confluent. A straightforward analysis of the terms substitutable for Z shows that no redexes can be created (in particular, the sub-term headed by f in the right-hand side cannot give rise to a redex, as that would require $Z(x,y)$ to be instantiated by x which would cause the redex to be 'erased before it is created', so to speak, see the appendix), hence the system is terminating by the Finite Developments Theorem. However, taking the disjoint union with the left-linear and obviously complete TRS consisting of the single rule $h(x,y) \to x$ yields a non-terminating PRS as witnessed by:

$$\underline{f(x.x, xy.h(x,y))} \to g(h(a, f(x.\underline{h(x,a)}, xy.h(x,y)))) \to g(h(a, f(x.x, xy.h(x,y))))$$

Note the reduction sequence above is of the form $t \twoheadrightarrow g(h(a,t))$ for $t = f(x.x, xy.h(x,y))$, hence gives rise to the infinite spiraling reduction:

$$t \twoheadrightarrow g(h(a,t)) \twoheadrightarrow g(h(a, g(h(a,t)))) \twoheadrightarrow g(h(a, g(h(a, g(h(a,t)))))) \twoheadrightarrow \dots$$

Next we turn our attention to normalization. Normalization is modular in the first-order case as a simple bottom-up argument shows. The result does not extend to PRSs, to wit the following counterexample.

Counterexample 4.4. The PRS consisting of the two rules

$$\begin{array}{rcl} f(x.Z(x), y.y) & \to & f(x.Z(x), y.Z(Z(y))) \\ f(x.x, y.Z(y)) & \to & a \end{array}$$

is normalizing as can be shown by induction on terms substitutable for Z (consider an f-term: if it is not a redex, it cannot become one; if the second rule applies to it, then a is its normal form; if only the first rule applies to it, it can only be applied once, see the appendix). However, combining it with the trivially normalizing one-rule TRS $g(g(x)) \to x$ yields a system which is not normalizing (it 're-enables' application of the first rule), as witnessed by the cycle:

$$f(x.g(x), y.y) \leftrightarrow f(x.g(x), y.g(g(y)))$$

in which each term is the only possible reduct of the other.

Normalization of the PRS above relies on the left-hand side of its first rule to be non-embeddable into its right-hand side: if it *were* embeddable, the term substituted for $Z(Z(y))$ should be reducible to, and therefore identical, to y, but then the second rule would have been applicable to its lhs as well, ensuring normalization. By adding the projection rule of the TRS above, the left-hand side *can* be embedded, thus destroying normalization.

Counterexample 4.3 witnesses that left-linear completeness is not modular for PRSs.

5. Modularity of confluence in left-linear pattern systems

For first-order left-linear TRSs, modularity of confluence is a trivial consequence of modularity of confluence for arbitrary TRSs. However, since the latter fails in the higher-order case, one may wonder whether left-linearity would suffice to regain modularity of confluence. Indeed it does; the following is a direct corollary of the results of [22].

Theorem 5.1. *Confluence is modular for left-linear pattern systems (applicative TRSs, CRSs, and PRSs).*

The idea of the proof, as presented in the PhD thesis[2] of van Oostrom [18], is to use the Hindley–Rosen Lemma and confluence of each of the PRSs, to reduce confluence of the union to their commutation. The latter holds, because since the signatures are disjoint, and since the rules of the respective PRSs were assumed to be left-linear pattern rules, they are therefore orthogonal *to each other*. The results for applicative pattern TRSs and CRSs follow since these can be embedded faithfully into PRSs.

As a consequence confluence is modular for left-linear pattern STTRSs as well.

Remark 5.2. One may wonder whether confluence is modular for non-duplicating rewrite systems. In the case of CRSs and PRSs the answer is negative [9] (note that the β-rule of the λ-calculus is non-duplicating as a higher-order rule). We leave the question whether confluence is modular for non-duplicating applicative pattern (ST)TRSs to future research.

[2]In fact, the more general result is shown there that the (ordinary, non-disjoint) union of two left-linear confluent PRSs is confluent, if the rules are *weakly* orthogonal w.r.t. each other, *i.e.* all critical pairs are trivial.

6. Normalization and termination are modular for non-duplicating pattern STTRSs

In this section we show normalization and termination to be modular for non-duplicating pattern STTRSs. In order to overcome the problem that the classical notion of layer will not suffice as the *rank* of a term (as the rank could then increase across reduction)–illustrated in Figure 2–we introduce appropriate notions of *component* and *component-type size*, the idea of the latter being that even though components may become nested (rendering the classical notion of rank useless), this can only be done by means of applying one component to another leading to a decrease in the size of the component types. As this measure only takes creation of components by means of application into account, not duplication of existing components, the results are restricted to non-duplicating systems (they have to be in view of Section 2).

In order to stratify mixed applicative terms, we refine the standard notion of a multi-hole context (see *e.g.*, [25, Section 2.1.1]) based on classifying symbols into colors. A function symbol belonging to Σ_γ is said to have *color* γ. We will conventionally refer to color 0 as *white*, 1 as *black*, and employ both white ($\square$) and black ($\blacksquare$) typed holes, to be filled by top-white and top-black (see below) terms of the appropriate types, respectively. A hole which may be either white or black is denoted by $\boxtimes$. We will view colors both as booleans, applying negation ($\overline{\gamma}$) and exclusive-or ($\gamma_1 \otimes \gamma_2$) to them, and as numbers, multiplying by them modulo 2.

To illustrate our constructions we make use of the following running example.

Example 6.1. In $\mathcal{T}$ the disjoint union of the white rewrite system $\mathcal{T}_0 = \{f\,Z\,W \to Z\,W\}$ with $f : (o \to o) \to o \to o$ and $a : 0$, and the black rewrite system $\mathcal{T}_1 = \{g\,a \to a\}$ with $g : o \to o$ and $b : o$, we have the reduction:

$$g\,(f\,(f\,g)\,b) \to g\,(f\,g\,b) \to g\,(g\,b) \to g\,b \to b$$

One can think of components, to be defined next, as the applicative analogue of the notion of layer, well-known from the study of modular properties of first-order term rewriting systems, see *e.g.*, [25, Section 5.7.1].

Definition 6.2. For γ either black or white, a γ-*component* is a non-empty context built from γ-symbols and $\overline{\gamma}$-holes, which does not have active holes, *i.e.* holes are not applied.

Example 6.3. For the STTRSs of Example 6.1, $f\,\blacksquare\,\blacksquare$ and $f\,(f\,\blacksquare)$ are 0-components, and b, $g\,g$ and $g\,(g\,(g\,\square))$ are 1-components. Non-examples of components are $\blacksquare$ (empty), $f\,g$ (symbols of mixed colors), $\blacksquare\,\blacksquare$ (active hole), $f\,\square$ (same color symbol and hole), and $f\,\blacksquare\,\square$.

We employ C, D, E to range over components. In the following algebraic semantics, we will view every component C of type τ having holes of, from left to right, types $\sigma_1, \ldots, \sigma_n$ as an n-ary function symbol $\boldsymbol{C} : \sigma_1 \times \ldots \times \sigma_n \to \tau$ of the *component* signature $\boldsymbol{\Sigma}$. We employ $\boldsymbol{t}$, $\boldsymbol{s}$, $\boldsymbol{u}$ to range over $\boldsymbol{\Sigma}$-terms.

Definition 6.4. The *component* algebra has $\boldsymbol{\Sigma}$-terms as carrier.

$$\begin{array}{rcll} [\![f]\!] & = & f & \\ [\![@]\!](\boldsymbol{C}(\vec{\boldsymbol{t}}), \boldsymbol{D}(\vec{\boldsymbol{s}})) & = & (\boldsymbol{C}\,\boldsymbol{D})(\vec{\boldsymbol{t}}, \vec{\boldsymbol{s}}) & \text{if } C,\, D \text{ have the same color} \\ & = & (\boldsymbol{C}\,\boxtimes)(\vec{\boldsymbol{t}}, \boldsymbol{D}(\vec{\boldsymbol{s}})) & \text{if } C,\, D \text{ have distinct colors} \end{array}$$

The component algebra gives rise to an obvious bijection mapping each (closed) Σ-term t to its interpretation as $\boldsymbol{\Sigma}$-term, which we indicate by boldface $\boldsymbol{t}$, and vice versa. A term is said to be *top*-white/black if the root symbol of its interpretation is white/black.

Example 6.5. The interpretation of the top-black term $t = g\,(f\,(f\,g)\,b)$ of Example 6.1 is the component term $\boldsymbol{t} = \boldsymbol{C}(\boldsymbol{D}(\boldsymbol{E}_1, \boldsymbol{E}_2))$ with $\boldsymbol{C} = g\,\square$, $\boldsymbol{D} = f\,(f\,\blacksquare)\,\blacksquare$, $\boldsymbol{E}_1 = g$, $\boldsymbol{E}_2 = b$.

Remark 6.6. Our choice to model decomposition of terms by means of interpretation into the component algebra is at an abstraction level intermediate between traditional *ad hoc* approaches (involving notions such as *special subterms*, *cf.* [25, Section 5.7.1]), and more recent *categorical* approaches (involving notions such as *monads*, *cf.* [1]), to modularity. It should be interesting to investigate whether the latter approach, set up to deal with functional TRSs (and collapsing of components), can be adapted to the present setting of applicative TRSs (and application of components).

The idea of the following definition is to measure a term by the sum of the 'applicative power' of its components as expressed by their types. More precisely, the measure is defined by associating a pair to each term; the elements of this pair takes into account the color of the context in which the term is substituted as follows: If a top-white (top-black) term is put into a white (black) context, the type of the term itself, *i.e.* the top-component, does not contribute to its measure; it shouldn't since we are only interested in the types of *components*, *i.e.* in changes of color, and the top of a term will not be a component when put into a context of the same color.

Definition 6.7. The *component-type size*[3] $|t|$ of term t is defined to be the pair $|\boldsymbol{t}|$ defined by:

$$|\boldsymbol{C}(\vec{\boldsymbol{t}})| = (\gamma \cdot \#\tau + \#\vec{\boldsymbol{t}}, \overline{\gamma} \cdot \#\tau + \#\vec{\boldsymbol{t}}) \quad \text{if } C : \tau \text{ has color } \gamma$$

where $\#b = 1$, $\#(\sigma \to \tau) = \#\sigma + 1 + \#\tau$, and $\#\boldsymbol{C}(\vec{\boldsymbol{t}}) = \#\tau + \#\vec{\boldsymbol{t}}$ if $C : \tau$. We use $<$ to denote the order induced on $\boldsymbol{\Sigma}$-terms by comparing their component-type sizes by means of the product order of the less-than relation on such pairs of natural numbers (conjunction of component-wise less-than), and $\leq$ to denote its reflexive closure. We use $\cdot_1$ and $\cdot_2$ to denote the projection onto the first and second element of a pair, respectively.

That is, the first (second) element of the pair measures the size of the types of all components that the term contributes to when substituted into a white (black) context.

Example 6.8. As we have $g\,\square : o$, $f\,(f\,\blacksquare)\,\blacksquare : o$, $g : o \to o$, and $b : o$, for the components in Example 6.3, the component-type size $|t|$ of the top-black term $t = g\,(f\,(f\,g)\,b)$ is $(1 \cdot \#o + n, 0 \cdot \#o + n)$ with $n = \#o + \#(o \to o) + \#o$. Therefore, $n = 1 + 3 + 1 = 5$ and $|t| = (6, 5)$. That is, only if the (top-black) term t is put into a white context, the type o of the term itself also contributes (1) to the component-type size; otherwise, when put into a black context, only the sub-components contribute (5) to the component-type size.

Remark 6.9. Instead of pairs of 'white' and 'black' values, one could also work with triples comprising the color of the top-component, its size, and the sum of the sizes of the proper sub-components. Formally, a pair of natural numbers (n, m) with $n \neq m$, is mapped to the triple $(0, m - n, n)$ if $n < m$ and to $(1, n - m, m)$ otherwise. Conversely, a triple of natural numbers (γ, n, m) with $\gamma \leq 1$ and $0 < n$ is mapped to the pair $(\gamma \cdot n + m, \overline{\gamma} \cdot n + m)$.

[3] The component-type size is an adaptation of the rank introduced in [2, Definition 8.56].

Observe that if t and s have the same type but distinct top-colors, then their component-type sizes are distinct, which follows immediately from the above remark since a triple has the top-color as first component.

Lemma 6.10. *The component algebra equipped with $<$ constitutes a well-founded (weakly) monotone Σ-algebra* [25, Definitions 6.2.1,6.4.28].

Proof. Well-foundedness of $<$ is trivial. Application (@) being the only non-nullary symbol, it suffices to check its (weak) monotonicity. This follows by straightforward calculations, see the appendix. ∎

Example 6.11. Instead of computing directly $|g\,(g\,b)| = (1,0) < (6,5) = |g\,(f\,g\,b)|$, the lemma allows to conclude $|g\,(g\,b)| < |g\,(f\,g\,b)|$ from $|g\,b| = (1,0) < (4,5) = |f\,g\,b|$ by strict monotonicity of application in its second argument. The sub-term property [25, Definition 6.4.28] does *not* hold: $|f| = (0,3) \not< (0,1) = |f\,a|$ (it does for 'special' sub-terms).

In the traditional terminology of the theory of modularity [17, 25], the following key lemma bounds the component-type size of a term having a *monochrome top*, by the component-type sizes of its *principal/alien sub terms*.

Lemma 6.12. *Let all symbols in $t{:}\tau$ have color γ and ϕ be a substitution. Then for $b\in\{0,1\}$*

$$|t^{\phi}|_b \leq (b \otimes \gamma) \cdot \#\tau + \sum_{Z \in t} |\phi(Z)|_{\gamma}$$

where the summation quantifies over occurrences *of variables in t. Secondly, if t is non-empty and t^{ϕ} top-γ, then if t is a pattern, equality holds and t is top-γ. Thirdly, and conversely, if equality holds, t^{ϕ} is top-γ, and all $\phi(Z)$ are top-$\overline{\gamma}$, then t is a non-empty pattern.*

Proof. By induction on t and calculation, and in the case of the second claim noting that if t is non-empty and t^{ϕ} is top-$\overline{\gamma}$, then

$$|t^{\phi}|_b < \sum_{Z \in t} |\phi(Z)|_{\gamma}$$

That is: Strict inequality holds even after removing the first summand. See the appendix. ∎

Example 6.13. Let $\phi(Z) = g$ and $\phi(W) = b$. If $l = f\,Z\,W$ then all symbols in l are white and $|l^{\phi}| = |f\,g\,b| = (4,5)$. Computing the right-hand side of the inequality in the lemma for, respectively, $b = 0$ and $b = 1$ yields the same pair $(4,5)$. Since the range of ϕ consists of top-white terms, we must conversely have by the lemma that l is a non-empty pattern which indeed it is.

If $r = Z\,W$, then all symbols in r are trivially white and $|r^{\phi}| = |g\,b| = (1,0)$. Computing the right-hand side of the inequality in the lemma for, respectively, $b = 0$ and $b = 1$ yields the strictly greater pair $(4,5)$.

Definition 6.14. A rewrite rule is *non-duplicating* if no free (meta-)variable occurs more often in its right-hand side than in its left-hand side.

The following lemma is an analogue of the classical lemma in the theory of modularity of functional TRSs that the *rank* of a term cannot increase along a reduction *cf. e.g.*, [27, 25]. In the present applicative case, we have to require rules to be non-duplicating in the light of Counterexample 2.5. The condition entails that rule application can essentially only 'recombine' the components of a term, which will suffice for modularity of termination and normalization of STTRSs, as for these 'recombination' will entail a decrease in the component-type size.

Lemma 6.15. *If $t \to s$ in the disjoint union of non-duplicating pattern STTRSs, then $|t| \geq |s|$.*

Proof. We claim $|l^\phi| \geq |r^\phi|$, for any rule $l \to r$ with l, r of type τ, and any substitution ϕ. Assuming the claim holds, the result follows by weak monotonicity (Lemma 6.10). The claim itself holds since for $b \in \{0,1\}$

$$|l^\phi|_b = (b \otimes \gamma) \cdot \#\tau + \sum_{Z \in l} |\phi(Z)|_\gamma \geq (b \otimes \gamma) \cdot \#\tau + \sum_{Z \in r} |\phi(Z)|_\gamma \geq |r^\phi|_b \tag{6.1}$$

Here the equality holds by Lemma 6.12 using that l is a pattern, which holds by assumption, and is non-empty, which holds by the general assumption on applicative TRS; the first inequality holds by the assumption that rules are non-duplicating; and the second inequality holds by Lemma 6.12 again. ■

If the source and the target of a step have distinct top-colors then, by the observation after Remark 6.9, their component-type sizes must be distinct and the inequality in the lemma strict.

Example 6.16. For ϕ, l and r as in Example 6.13 we have $t = l^\phi \to r^\phi = s$ by an application of the rule $l = f\,Z\,W \to Z\,W = r$. As was computed there, indeed $|t| = (4,5) \geq (1,0) = |s|$; the component-size type strictly decreases because the rule 'recombines' the arguments substituted for Z and W in its right-hand side $Z\,W$.

We show now that if the component-type size does not decrease across a rewrite step, then the components are not 'recombined' and hence the step can be viewed as a step on component symbols. To that end, we define the rewrite relation $\Rightarrow$ on component terms as being generated by the, infinitely many, rewrite rules $\boldsymbol{C}(\vec{Z}) \to \boldsymbol{D}(\vec{W})$ for all components C,D of the same color, such that $C[\vec{Z}] \to D[\vec{W}]$.

Lemma 6.17. *If $t \to s$ and $|t| = |s|$ in the disjoint union of non-duplicating pattern STTRSs, then $\boldsymbol{t} \Rightarrow \boldsymbol{s}$.*

Proof. By induction on the derivation of $t \to s$, see the appendix. ■

Example 6.18. As seen in Examples 6.8 and 6.11 for the step $g\,(f\,(f\,g)\,b) \to g\,(f\,g\,b)$ it holds $|g\,(f\,(f\,g)\,b)| = (6,5) = |g\,(f\,g\,b)|$. From Example 6.5, the corresponding component terms are $\boldsymbol{t} = \boldsymbol{C}(\boldsymbol{D}(\boldsymbol{E}_1, \boldsymbol{E}_2))$ and $\boldsymbol{s} = \boldsymbol{C}(\boldsymbol{D'}(\boldsymbol{E}_1, \boldsymbol{E}_2))$, using the component symbols given there and $\boldsymbol{D'} = f \blacksquare \blacksquare$. We indeed have, as per the lemma, that $\boldsymbol{t} \Rightarrow \boldsymbol{s}$ by an application of the rule $\boldsymbol{D}(Z,W) \to \boldsymbol{D}(Z,W)$ obtained from the white step $D[Z,W] = f\,(f\,Z)\,W \to f\,Z\,W = D[Z,W]$.

Summarizing the above, steps either decrease the component-type size or respect components, in the sense that they can be lifted to the component algebra. This suffices for establishing modularity of termination and normalization for non-duplicating pattern STTRSs.

Theorem 6.19. *Termination is modular for non-duplicating pattern STTRSs.*

Proof. By Lemma 6.15 the terms along an hypothetical infinite $\rightarrow$-reduction must, after some finite number of stepsx all have the same component-type size. We conclude by Lemma 6.17 and the observation that if the rewrite relations $\rightarrow_\gamma$ are terminating, then $\Rightarrow$ is terminating (which can be seen by an application of recursive path orders induced by the precedences induced by $\rightarrow_\gamma$ on components). ∎

Theorem 6.20. *Normalization is modular for non-duplicating pattern STTRSs.*

Proof. By Lemma 6.15 the component-type size cannot increase along $\rightarrow$-reduction, hence any term can be reduced to a term of minimal component-type size, that is, such that any further reduction will leave the component-type size unchanged. We conclude by Lemma 6.17 and the observation that if the rewrite relations $\rightarrow_\gamma$ are normalising, then the corresponding $\Rightarrow$-strategy is terminating (as can be seen by an application of recursive path orders induced by the precedences induced by the normalising $\rightarrow_\gamma$-strategies on components). ∎

Example 6.21. Each applicative TRS in Example 6.1 is a non-duplicating terminating/normalizing pattern STTRS. Hence by Theorem 6.19/6.20 so is their disjoint union.

Remark 6.22. Since terms of base type are 'applicatively inert' it might be possible to lift the non-duplicatingness restriction on variables of base type in Theorem 6.20, by appropriately adapting the component-type size. We leave this to future research.

In view of the theorems and of the fact that termination and normalization are modular for non-duplicating functional TRSs [23, 15, 17], one may wonder whether normalization and termination are modular for non-duplicating (or left-and-right-linear) higher-order rewriting systems (CRSs or PRSs). Leaving the other cases to future research (but *cf.* [2, Chapter 9] for some initial and related results), we show that modularity of termination fails for left-and-right-linear PRSs because linear PRS rules (such as the β-rule) can still 'embed duplication'.

Counterexample 6.23. The following rules constitute a left-and-right-linear orthogonal PRS:

$$\begin{array}{rcl} f(a,b,Z) & \rightarrow & g(y.f(y,y,y),Z) \\ g(y.Z(y),W) & \rightarrow & Z(W) \end{array}$$

which can be shown terminating by adapting *e.g.* the computability-style termination proof of FD à la Tait of [19], using that by confluence there is no term which can be reduced to both a and b. We claim that for all terms t and for all terminating first-order substitutions ϕ, t^ϕ is terminating. For a proof, see the appendix. The result follows from the claim by taking the, terminating, identity substitution for ϕ.

Now note that this non-duplicating PRS can 'simulate' the duplicating first TRS of Counterexample 1.2:

$$f(a,b,x) \rightarrow g(y.f(y,y,y),x) \rightarrow f(x,x,x)$$

hence termination is not preserved when taking the disjoint union with the second TRS of that counterexample.

7. Conclusion

We have given higher-order counterexamples to the modularity of a number of properties known to be modular in the first-order rewriting. We think the surprise is more in that several properties *are* modular in the first-order case than in their failure to be modular in the higher-order case: Only because first-order variables cannot be nested, enabling the notion of rank, the absence of typing or linearity constraints is not immediately fatal for modularity. Imposing such constraints has enabled us to adapt the notion of rank of a term, to that of component-type size of the term, and to show that termination and normalization are modular for non-duplicating simply typed TRSs. It should be interesting to extend the latter result from rewrite systems with higher-order variables (STTRSs) to rewrite systems also having abstraction built in (CRSs,PRSs). One possibility would be to consider such systems having a *linear* substitution calculus, *i.e.* having terms modulo linear $\alpha\beta\eta$-calculus.

With respect to automation we note that the termination and confluence properties required of the first-order TRSs in this report have been established by available tools, but that tool support in the higher-order case is still lacking, making proof-by-hand necessary (*cf.* the proofs in the appendix, except for those corresponding to Section 6) for the moment. We would expect at least the last of these (corresponding to Counterexample 6.23) to be a consequence of an automatable general termination method.

A proof-theoretic contributation could reside in the notion of component algebra, lying betwixt ad hoc and categorical formalisations qua level of abstraction, enabling algebraic proofs, here of modularity of termination/normalization. It would be interesting to see whether the notion of component algebra can be profitably retrofitted onto classical modularity results.

Acknowledgments. We thank the organizers and participants of the Austria–Japan Summer Workshop on Term Rewriting, 8–13 August 2005, Obergurgl, for the opportunity to present an early version of this report, and them, Andrzej Filinski, Cynthia Kop, and the anonymous referees for feedback.

References

[1] M. Abbott, N. Ghani, and C. Lüth. Abstract modularity. In *Proceedings of the 16th International Conference on Rewriting Techniques and Applications (RTA 2005)*, volume 3467 of *Lecture Notes in Computer Science*, pages 46–60. Springer, 2005.

[2] C. Appel. Modularity in higher-order term rewriting. Master's thesis, DIKU, University of Copenhagen, June 2009.

[3] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

[4] H. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 1985.

[5] M. Hamana. Universal algebra for termination of higher-order rewriting. In *Proceedings of the 16th International Conference on Rewriting Techniques and Applications (RTA 2005)*, volume 3467 of *Lecture Notes in Computer Science*, pages 135–149. Springer, 2005.

[6] G. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27(4):797–821, Oct. 1980.

[7] S. Kahrs. Confluence of curried term-rewriting systems. *Journal of Symbolic Computation*, 19(6):601–623, June 1995.

[8] R. Kennaway, J. Klop, R. Sleep, and F.-J. d. Vries. Comparing curried and uncurried rewriting. *Journal of Symbolic Computation*, 21(1):15–39, Jan. 1996.

[9] J. Klop. *Combinatory Reduction Systems.* PhD thesis, Utrecht University, 1980.

[10] J. Klop, V. v. Oostrom, and F. v. Raamsdonk. Reduction strategies and acyclicity. In *Rewriting, Computation and Proof, Essays Dedicated to Jean-Pierre Jouannaud on the Occasion of His 60th Birthday*, volume 4600 of *Lecture Notes in Computer Science*, pages 89–112. Springer, 2007.

[11] J. Klop, V. van Oostrom, and F. van Raamsdonk. Combinatory reduction systems: introduction and survey. *Theoretical Computer Science*, 121(1-2):279–308, Dec. 1993.

[12] M. Kurihara and I. Kaji. Modular term rewriting systems and the termination. *Information Processing Letters*, 34(1):1–4, Feb. 1990.

[13] M. Kurihara and A. Ohuchi. Modularity in noncopying term rewriting. *Theoretical Computer Science*, 152(1):139–169, Dec. 1995.

[14] R. Mayr and T. Nipkow. Higher-order rewrite systems and their confluence. *Theoretical Computer Science*, 192(1):3–29, Feb. 1998.

[15] A. Middeldorp. Modular aspects of properties of term rewriting systems related to normal forms. In *Proceedings of the 3rd International Conference on Rewriting Techniques and Applications (RTA 1989)*, volume 355 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 1989.

[16] T. Nipkow. Higher-order critical pairs. In *Proceedings of the 6th annual IEEE Symposium on Logic in Computer Science (LICS 1991)*, pages 342–349, 1991.

[17] E. Ohlebusch. A simple proof of sufficient conditions for the termination of the disjoint union of term rewriting systems. *Bulletin of the European Association for Theoretical Computer Science*, 49:178–183, 1993.

[18] V. v. Oostrom. *Confluence for Abstract and Higher-Order Rewriting*. PhD thesis, VU Amsterdam, 1994.

[19] V. v. Oostrom. Take five. Technical Report IR 406, VU Amsterdam, June 1996.

[20] V. v. Oostrom. Modularity of confluence, constructed. In *Proceedings of the 4th International Joint Conference on Automated Reasoning (IJCAR 2008)*, volume 5195 of *Lecture Notes in Computer Science*, pages 348–363. Springer, 2008.

[21] V. v. Oostrom and F. v. Raamsdonk. Comparing combinatory reduction systems and higher-order rewrite systems. In *Proceedings of the 1st International Workshop on Higher-Order Algebra, Logic and Term Rewriting (HOA 1993)*, volume 814 of *Lecture Notes in Computer Science*, pages 276–304. Springer, 1994.

[22] V. v. Oostrom and F. v. Raamsdonk. Weak orthogonality implies confluence: the higher-order case. In *Proceedings of the 3rd International Symposium on Logical Foundations of Computer Science (LFCS 1994)*, volume 813 of *Lecture Notes in Computer Science*, pages 379–392. Springer, 1994.

[23] M. Rusinowitch. On termination of the direct sum of term-rewriting systems. *Information Processing Letters*, 26(2):65–70, Oct. 1987.

[24] M. Schmidt-Schauss, M. Marchiori, and S. Panitz. Modular termination of r-consistent and left-linear term rewriting systems. *Theoretical Computer Science*, 149(2):361–374, Oct. 1995.

[25] Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.

[26] Y. Toyama. Counterexamples to termination for the direct sum of term rewriting systems. *Information Processing Letters*, 25(3):141–143, May 1987.

[27] Y. Toyama. On the Church-Rosser property for the direct sum of term rewriting systems. *Journal of the ACM*, 34(1):128–143, Jan. 1987.

[28] Y. Toyama, J. Klop, and H. Barendregt. Termination for the direct sum of left-linear term rewriting systems. In *Proceedings of the 3rd International Conference on Rewriting Techniques and Applications (RTA 1989)*, volume 355 of *Lecture Notes in Computer Science*, pages 477–491. Springer, 1989.

[29] T. Yamada. Confluence and termination of simply typed term rewriting systems. In *Proceedings of the 12th International Conference on Rewriting Techniques and Applications (RTA 2001)*, volume 2051 of *Lecture Notes in Computer Science*, pages 338–352. Springer, 2001.

Appendix A. Proofs omitted from the main text

Proof of confluence of the CRS of Counterexample 3.2. To show confluence of the CRS, consider its first two rules which are length-decreasing, hence terminating. By Nipkow's higher-order Critical Pair Lemma—which also applies to CRSs—and Newman's Lemma it suffices to show its critical pair

$$f([x]Z(x)) \leftarrow f(f([x]Z(x))) \rightarrow f(f(Z(a)))$$

can be joined in a common reduct; it can by performing a single step from either term to the common reduct $f(Z(a))$. The third rule is shown redundant by proving that its lhs is convertible to its rhs. This is clear in case x does not occur in the term substituted for Z. Otherwise, from applying the second rule to either side one sees that it suffices to show that any term $f(C[a])$ with C a unary context, reduces to $f(a)$. This is shown by induction on the size of C, applying either the first or the second rule, depending on whether the head-symbol of C is either f or $[x]$, respectively. ∎

Proof of termination of the CRS of Counterexample 3.3. As the CRS is orthogonal and non-erasing, to prove termination it suffices to prove normalization.

This is proven by induction on the term t, the only interesting case being when it is of shape $f([x][y]t')$ with t' in normal form. Distinguish cases on whether x or y occurs free in t'.

In the case that neither x nor y occurs free in t', the term t simply reduces to the normal form t' by contracting the root redex.

If x occurs free in t', then y cannot occur free too, since the CRS does not contain function symbols of arity greater than one. Thus, contracting the root redex yields $t'[x := [x][y]t'[x := y]]$. This term can only be not in normal form, if t' is of shape $C[f(x)]$ for some context C in normal form, giving $C[f([x][y]C[f(y)])]$ which reduces to $C[C[f([x]x)]]$, which normalizes by the first case.

If y occurs free in t', then x cannot occur free too, since the CRS does not contain function symbols of arity greater than one. Thus, contracting the root redex yields $t'[y := [x]x]$ which is terminating by the induction hypothesis. ∎

Proof of termination of the PRS of Counterexample 4.3. To show termination of the PRS, we claim that it does not allow creation of redexes along any reduction, from which termination follows by the Finite Developments Theorem. To prove the claim, let $t \rightarrow s$ be a rewrite step, consider a redex in s, and assume, for contradiction, that the redex is created in the step $t \rightarrow s$. Split on cases as follows:

- If the position of f is a descendant of a position in t, then the origin must have been either in the context part or the substitution part of the step.

  In the former case, f can only not have been the head symbol of a redex in t, if part of the left-hand side is created by the step. This is impossible, since then the outermost symbol g of the right-hand side should be part of the pattern, which it cannot be.

  In the latter case, note that the only possible change the substitution part undergoes in the right-hand side compared to the left-hand side, is the replacement of its formal parameters x and y by actual terms. For the first occurrence of Z, these terms are a and some term having f as head-symbol respectively, neither of which cannot create a redex since a nor f occur in the left-hand side. For the second occurrence of Z, the actual terms are x and a respectively. For the latter we reason as before, while the former will be free in the substitution part, hence cannot be part of the pattern. For the third occurrence of Z, we reason in the same way.

- If the position of f is not a descendant of a position in t, then f is created as an "instance" of the occurrence of f in the right-hand side of the rewrite rule. But that cannot be as it would force f to be erased: The created redex occurs at the position of f in a sub-term

$$g(Z(a, f(x.Z(x,a), xy.Z(x,y))))$$

where Z has been replaced by some term and the entire sub-term reduced to $\beta\overline{\eta}$-normal form. Note that the term u substituted for Z must necessarily *erase* its second argument, as a redex could not have been created. But if u erases its second argument, then f is erased in the step $t \to s$, a contradiction. ∎

Proof of normalization of the PRS of Counterexample 4.4. Normalization of the PRS is shown by induction on terms. The only interesting cases are terms of the form $f(x.t, y.s)$ where t and s are normalizing by the induction hypothesis. Thus we may assume without loss of generality that t and s are in fact in normal form. To show normalization, distinguish cases on the shapes of t and s.

- If t is equal to x, then the term reduces to the normal form a.
- If t is distinct from x and s is distinct from y, then the term is in normal form itself already.
- If t is distinct from x and s is equal to y, the term reduces in one step to

$$f(x.t, y.t[x := t[x := y]])$$

By the assumption that t is distinct from x, the term in the first argument of the reduct is distinct from x and the term in the second argument is distinct from y, so neither of the rules is applicable at the head.

By the assumption that t is in normal form, both the terms in the first and second arguments are so as well, since looking at the left-hand sides of the rules one notes that the only way a redex could be created by a substitution requires that one substitutes the bound variable (x or y, respectively), which cannot be. ∎

Proof of (weak) monotonicity Lemma 6.10. We first show that application is monotonic in its first argument. Let $\boldsymbol{C}_1(\vec{\boldsymbol{t}_1})$, $\boldsymbol{C}_2(\vec{\boldsymbol{t}_2})$ and $\boldsymbol{D}(\vec{\boldsymbol{s}})$ be component terms with $C_i : \sigma \to \tau$ of color γ_i and $D : \sigma$ of color δ, and such that $|\boldsymbol{C}_1(\vec{\boldsymbol{t}_1})| < |\boldsymbol{C}_2(\vec{\boldsymbol{t}_2})|$. Defining $\gamma = \gamma_1$, distinguish cases on whether γ is γ_2 or not.

$(\gamma = \gamma_2)$ Then $|\boldsymbol{C}_1(\vec{\boldsymbol{t}_1})| < |\boldsymbol{C}_2(\vec{\boldsymbol{t}_2})|$ implies $\#\vec{\boldsymbol{t}_1} < \#\vec{\boldsymbol{t}_2}$, and we distinguish cases on whether γ is δ or not.

$(\gamma = \delta)$ Then $\gamma_2 = \delta$ and we calculate

$$\begin{array}{rcl} |(\boldsymbol{C_1}\,\boldsymbol{D})(\vec{\boldsymbol{t}_1}, \vec{\boldsymbol{s}})| & = & (\gamma \cdot \#\tau + \#\overrightarrow{\boldsymbol{t}_1, \boldsymbol{s}}, \overline{\gamma} \cdot \#\tau + \#\overrightarrow{\boldsymbol{t}_1, \boldsymbol{s}}) \\ & < & (\gamma \cdot \#\tau + \#\overrightarrow{\boldsymbol{t}_2, \boldsymbol{s}}, \overline{\gamma} \cdot \#\tau + \#\overrightarrow{\boldsymbol{t}_2, \boldsymbol{s}}) \\ & = & |(\boldsymbol{C_2}\,\boldsymbol{D})(\vec{\boldsymbol{t}_2}, \vec{\boldsymbol{s}})| \end{array}$$

$(\gamma \neq \delta)$ Then $\gamma_2 \neq \delta$ and we calculate

$$\begin{array}{rcl} |(\boldsymbol{C}\,\boxtimes)(\vec{\boldsymbol{t}_1}, \boldsymbol{D}(\vec{\boldsymbol{s}}))| & = & (\gamma \cdot \#\tau + \#\vec{\boldsymbol{t}_1}, \boldsymbol{D}(\vec{\boldsymbol{s}}), \overline{\gamma} \cdot \#\tau + \#\vec{\boldsymbol{t}_1}, \boldsymbol{D}(\vec{\boldsymbol{s}})) \\ & < & (\gamma \cdot \#\tau + \#\vec{\boldsymbol{t}_2}, \boldsymbol{D}(\vec{\boldsymbol{s}}), \overline{\gamma} \cdot \#\tau + \#\vec{\boldsymbol{t}_2}, \boldsymbol{D}(\vec{\boldsymbol{s}})) \\ & = & |(\boldsymbol{C}\,\boxtimes)(\vec{\boldsymbol{t}_2}, \boldsymbol{D}(\vec{\boldsymbol{s}}))| \end{array}$$

($\gamma \neq \gamma_2$) Distinguishing cases on whether γ is δ or not, we use that $|\boldsymbol{C}_1(\vec{\boldsymbol{t}_1})| < |\boldsymbol{C}_2(\vec{\boldsymbol{t}_2})|$ implies $\#(\sigma \to \tau) + \#\vec{\boldsymbol{t}_1} < \#\vec{\boldsymbol{t}_2}$, and that $\#\boldsymbol{D}(\vec{\boldsymbol{s}}) = \#\sigma + \#\vec{\boldsymbol{s}}$ in:

($\gamma = \delta$) Then $\gamma_2 \neq \delta$ and we calculate

$$\begin{aligned} |(\boldsymbol{C_1}\,\boldsymbol{D})(\vec{\boldsymbol{t}_1}, \vec{\boldsymbol{s}})| &= (\gamma \cdot \#\tau + \#\vec{\boldsymbol{t}_1, \boldsymbol{s}}, \overline{\gamma} \cdot \#\tau + \#\vec{\boldsymbol{t}_1, \boldsymbol{s}}) \\ &< (\gamma \cdot \#\tau + \#\vec{\boldsymbol{t}_2}, \boldsymbol{D}(\vec{\boldsymbol{s}}), \overline{\gamma} \cdot \#\tau + \#\vec{\boldsymbol{t}_2}, \boldsymbol{D}(\vec{\boldsymbol{s}})) \\ &= |(\boldsymbol{C}\,\boxtimes)(\vec{\boldsymbol{t}_2}, \boldsymbol{D}(\vec{\boldsymbol{s}}))| \end{aligned}$$

($\gamma \neq \delta$) Then $\gamma_2 = \delta$ and we calculate

$$\begin{aligned} |(\boldsymbol{C}\,\boxtimes)(\vec{\boldsymbol{t}_1}, \boldsymbol{D}(\vec{\boldsymbol{s}}))| &= (\gamma \cdot \#\tau + \#\vec{\boldsymbol{t}_1}, \boldsymbol{D}(\vec{\boldsymbol{s}}), \overline{\gamma} \cdot \#\tau + \#\vec{\boldsymbol{t}_1}, \boldsymbol{D}(\vec{\boldsymbol{s}})) \\ &< (\gamma \cdot \#\tau + \#\vec{\boldsymbol{t}_2, \boldsymbol{s}}, \overline{\gamma} \cdot \#\tau + \#\vec{\boldsymbol{t}_2, \boldsymbol{s}}) \\ &= |(\boldsymbol{C_2}\,\boldsymbol{D})(\vec{\boldsymbol{t}_2}, \vec{\boldsymbol{s}})| \end{aligned}$$

We now show that application is monotonic in its second argument. Let $\boldsymbol{C}(\vec{\boldsymbol{t}})$, $\boldsymbol{D}_1(\vec{\boldsymbol{s}_1})$ and $\boldsymbol{D}_2(\vec{\boldsymbol{s}_2})$ be component terms with $C : \sigma \to \tau$ of color τ and $D_i : \sigma$ of color δ_i, and such that $|\boldsymbol{D}_1(\vec{\boldsymbol{s}_1})| < |\boldsymbol{D}_2(\vec{\boldsymbol{s}_2})|$. Defining $\delta = \delta_1$, distinguish cases on whether δ_1 is δ_2 or not.

($\delta = \delta_2$) We distinguish cases on whether γ is δ or not.

($\gamma = \delta$) Then using $\gamma = \delta_2$ and $|\boldsymbol{D}_1(\vec{\boldsymbol{s}_1})| < |\boldsymbol{D}_2(\vec{\boldsymbol{s}_2})|$ implies $\#\vec{\boldsymbol{s}_1} < \#\vec{\boldsymbol{s}_2}$ we calculate

$$\begin{aligned} |(\boldsymbol{C}\,\boldsymbol{D_1})(\vec{\boldsymbol{t}}, \vec{\boldsymbol{s}_1})| &= (\gamma \cdot \#\tau + \#\vec{\boldsymbol{t}}, \vec{\boldsymbol{s}_1}, \overline{\gamma} \cdot \#\tau + \#\vec{\boldsymbol{t}}, \vec{\boldsymbol{s}_1}) \\ &< (\gamma \cdot \#\tau + \#\vec{\boldsymbol{t}}, \vec{\boldsymbol{s}_2}, \overline{\gamma} \cdot \#\tau + \#\vec{\boldsymbol{t}}, \vec{\boldsymbol{s}_2}) \\ &= |(\boldsymbol{C}\,\boldsymbol{D_2})(\vec{\boldsymbol{t}}, \vec{\boldsymbol{s}_2})| \end{aligned}$$

($\gamma \neq \delta$) Then $\gamma \neq \delta_2$ and using $\#\boldsymbol{D}_i(\vec{\boldsymbol{s}_i}) = \#\sigma + \#\vec{\boldsymbol{s}_i}$ we calculate

$$\begin{aligned} |(\boldsymbol{C}\,\boxtimes)(\vec{\boldsymbol{t}}, \boldsymbol{D}_1(\vec{\boldsymbol{s}_1}))| &= (\gamma \cdot \#\tau + \#\vec{\boldsymbol{t}}, \boldsymbol{D}_1(\vec{\boldsymbol{s}_1}), \overline{\gamma} \cdot \#\tau + \#\vec{\boldsymbol{t}}, \boldsymbol{D}_1(\vec{\boldsymbol{s}_1})) \\ &< (\gamma \cdot \#\tau + \#\vec{\boldsymbol{t}}, \boldsymbol{D}_2(\vec{\boldsymbol{s}_2}), \overline{\gamma} \cdot \#\tau + \#\vec{\boldsymbol{t}}, \boldsymbol{D}_2(\vec{\boldsymbol{s}_2})) \\ &= |(\boldsymbol{C}\,\boxtimes)(\vec{\boldsymbol{t}}, \boldsymbol{D}_2(\vec{\boldsymbol{s}_2}))| \end{aligned}$$

($\delta \neq \delta_2$) Distinguishing cases on whether γ is δ or not, we use that $|\boldsymbol{D}_1(\vec{\boldsymbol{s}_1})| < |\boldsymbol{D}_2(\vec{\boldsymbol{s}_2})|$ implies $\#\sigma + \#\vec{\boldsymbol{s}_1} < \#\vec{\boldsymbol{s}_2}$, and that $\#\boldsymbol{D}_i(\vec{\boldsymbol{s}_i}) = \#\sigma + \#\vec{\boldsymbol{s}_i}$ in:

($\gamma = \delta$) Then $\gamma \neq \delta_2$ and we calculate

$$\begin{aligned} |(\boldsymbol{C}\,\boldsymbol{D_1})(\vec{\boldsymbol{t}}, \vec{\boldsymbol{s}_1})| &= (\gamma \cdot \#\tau + \#\vec{\boldsymbol{t}}, \vec{\boldsymbol{s}_1}, \overline{\gamma} \cdot \#\tau + \#\vec{\boldsymbol{t}}, \vec{\boldsymbol{s}_1}) \\ &< (\gamma \cdot \#\tau + \#\vec{\boldsymbol{t}}, \boldsymbol{D}_2(\vec{\boldsymbol{s}_2}), \overline{\gamma} \cdot \#\tau + \#\vec{\boldsymbol{t}}, \boldsymbol{D}_2(\vec{\boldsymbol{s}_2})) \\ &= |(\boldsymbol{C}\,\boxtimes)(\vec{\boldsymbol{t}}, \boldsymbol{D}_2(\vec{\boldsymbol{s}_2}))| \end{aligned}$$

($\gamma \neq \delta$) Then $\gamma \neq \delta_2$ and we calculate

$$\begin{aligned} |(\boldsymbol{C}\,\boxtimes)(\vec{\boldsymbol{t}}, \boldsymbol{D}_1(\vec{\boldsymbol{s}_1}))| &= (\gamma \cdot \#\tau + \#\vec{\boldsymbol{t}}, \boldsymbol{D}_1(\vec{\boldsymbol{s}_1}), \overline{\gamma} \cdot \#\tau + \#\vec{\boldsymbol{t}}, \boldsymbol{D}_1(\vec{\boldsymbol{s}_1})) \\ &< (\gamma \cdot \#\tau + \#\vec{\boldsymbol{t}}, \vec{\boldsymbol{s}_2}, \overline{\gamma} \cdot \#\tau + \#\vec{\boldsymbol{t}}, \vec{\boldsymbol{s}_2}) \\ &= |(\boldsymbol{C}\,\boldsymbol{D_2})(\vec{\boldsymbol{t}}, \vec{\boldsymbol{s}_2})| \end{aligned}$$

Weak monotonicity of application follows. ∎

Proof of substitution Lemma 6.12. In case t is a single variable Z, then the inequality reads

$$|\phi(Z)|_b \leq (b \otimes \gamma) \cdot \#\tau + |\phi(Z)|_\gamma$$

which holds by monotonicity of addition in case $b = \gamma$, and in case $b \neq \gamma$, then $b \otimes \gamma = 1$ and supposing $|\phi(Z)| = |\boldsymbol{C}(\vec{\boldsymbol{t}})|$ with $C : \tau$ of color δ, $|\boldsymbol{C}(\vec{\boldsymbol{t}})|_b \leq \#\tau + |\boldsymbol{C}(\vec{\boldsymbol{t}})|_\gamma$ follows since the difference between the components of $|\boldsymbol{C}(\vec{\boldsymbol{t}})| = (\delta \cdot \#\tau + \#\vec{\boldsymbol{t}}, \overline{\delta} \cdot \#\tau + \#\vec{\boldsymbol{t}})$ is $\#\tau$. The second and third claims hold vacuously since t is empty and $\phi(Z)$ cannot be both top-γ and top-$\overline{\gamma}$.

In case t is a function symbol f, then equality holds as follows from the calculation

$$|f|_b = (\gamma \cdot \#\tau, \overline{\gamma} \cdot \#\tau)_b = (b \otimes \gamma) \cdot \#\tau$$

The second and third claims hold: f is top-γ and equality indeed holds, and f is a non-empty pattern.

In case $t = t_1\, t_2$, then, writing $\boldsymbol{t}_i^{\boldsymbol{\phi}} = \boldsymbol{C}_i(\vec{\boldsymbol{t}_i})$ with C_i of some color γ_i, and for some types τ, σ, $C_1 : \sigma \to \tau$ and $C_2 : \sigma$. The induction hypotheses for t_i, combined with the definition of $|t_i^{\phi}|_b$, yield:

$$(b \otimes \gamma_1) \cdot \#(\sigma \to \tau) + \#\vec{\boldsymbol{t}_1} \le (b \otimes \gamma) \cdot \#(\sigma \to \tau) + \sum_{Z \in t_1} |\phi(Z)|_\gamma$$

$$(b \otimes \gamma_2) \cdot \#\sigma + \#\vec{\boldsymbol{t}_2} \le (b \otimes \gamma) \cdot \#\sigma + \sum_{Z \in t_2} |\phi(Z)|_\gamma$$

whereas the right-hand side of the inequality to be established is

$$(b \otimes \gamma) \cdot \#\tau + \sum_{Z \in t_1} |\phi(Z)|_\gamma + \sum_{Z \in t_2} |\phi(Z)|_\gamma$$

Distinguish cases on whether γ_1 is γ_2 or not.

($\gamma_1 = \gamma_2$) Then the left-hand side of the inequality is

$$|t^{\phi}|_b = |(t_1\, t_2)^{\phi}|_b = |t_1^{\phi}\, t_2^{\phi}|_b = |(\boldsymbol{C}_1\, \boldsymbol{C}_2)(\vec{\boldsymbol{t}_1}, \vec{\boldsymbol{t}_2})|_b = (b \otimes \gamma_1) \cdot \#\tau + \#\vec{\boldsymbol{t}_1}, \vec{\boldsymbol{t}_2}$$

so we must establish

$$(b \otimes \gamma_1) \cdot \#\tau + \#\vec{\boldsymbol{t}_1}, \vec{\boldsymbol{t}_2} \le (b \otimes \gamma) \cdot \#\tau + \sum_{Z \in t_1} |\phi(Z)|_\gamma + \sum_{Z \in t_2} |\phi(Z)|_\gamma$$

and we make a further case-distinction on whether γ is γ_1 or not.

($\gamma = \gamma_1$) Then the induction hypotheses specialise to

$$\#\vec{\boldsymbol{t}_1} \le \sum_{Z \in t_1} |\phi(Z)|_\gamma$$

$$\#\vec{\boldsymbol{t}_2} \le \sum_{Z \in t_2} |\phi(Z)|_\gamma$$

from which we conclude.

($\gamma \neq \gamma_1$) Then the induction hypotheses specialise to

$$\#(\sigma \to \tau) + \#\vec{\boldsymbol{t}_1} \le \sum_{Z \in t_1} |\phi(Z)|_\gamma$$

$$\#\sigma + \#\vec{\boldsymbol{t}_2} \le \sum_{Z \in t_2} |\phi(Z)|_\gamma$$

from which we conclude using $(b \otimes \gamma_1) \cdot \#\tau \le \#\tau \le \#(\sigma \to \tau)$.

($\gamma_1 \neq \gamma_2$) Then the left-hand side of the inequality is

$$|t^{\phi}|_b = |(\boldsymbol{C}_1 \boxtimes)(\vec{\boldsymbol{t}_1}, \boldsymbol{C}_2(\vec{\boldsymbol{t}_2}))|_b = (b \otimes \gamma_1) \cdot \#\tau + \#\vec{\boldsymbol{t}_1}, \boldsymbol{C}_2(\vec{\boldsymbol{t}_2})$$

so we must establish

$$(b \otimes \gamma_1) \cdot \#\tau + \#\vec{\boldsymbol{t}_1}, \boldsymbol{C}_2(\vec{\boldsymbol{t}_2}) \le (b \otimes \gamma) \cdot \#\tau + \sum_{Z \in t_1} |\phi(Z)|_\gamma + \sum_{Z \in t_2} |\phi(Z)|_\gamma$$

and, noting $\#\boldsymbol{C}_2(\vec{\boldsymbol{t}_2}) = \#\sigma + \#\vec{\boldsymbol{t}_2}$, we make a further case-distinction on whether γ is γ_1 or γ_2.

$(\gamma = \gamma_1)$ Then the induction hypotheses specialise to

$$\#\vec{\boldsymbol{t}_1} \le \sum_{Z \in t_1} |\phi(Z)|_\gamma$$

$$\#\sigma + \#\vec{\boldsymbol{t}_2} \le \sum_{Z \in t_2} |\phi(Z)|_\gamma$$

from which we conclude.

$(\gamma = \gamma_2)$ Then the induction hypotheses specialise to

$$\#(\sigma \to \tau) + \#\vec{\boldsymbol{t}_1} \le \sum_{Z \in t_1} |\phi(Z)|_\gamma$$

$$\#\vec{\boldsymbol{t}_2} \le \sum_{Z \in t_2} |\phi(Z)|_\gamma$$

from which we conclude using $(b \otimes \gamma_1) \cdot \#\tau + \#\sigma \le \#\tau + \#\sigma \le \#(\sigma \to \tau)$.

This finishes the verification of the first claim in the case $t = t_1\, t_2$. To verify the second claim we distinguish cases on whether t^ϕ, and hence t_1^ϕ, is top-γ or top-$\overline{\gamma}$.

If t^ϕ is top-γ then the assumption that t is a pattern yields t_1 and t_2 are patterns again and t_1 is not empty. By the induction hypothesis equality holds for t_1 which is top-γ. Hence t is top-γ as well and, using the conventions of the first claim, $\gamma_1 = \gamma$. We distinguish cases on whether t_2 is empty or not.

If t_2 is not empty, then by the induction hypothesis equality holds for it, and t_2^ϕ has color γ. Replacing $\le$ by $=$ everywhere in the above verification of the case $\gamma_1 = \gamma = \gamma_2$ of the first claim, yields the desired result.

If t_2 is empty, say it is the single variable Z, then we distinguish cases on whether the color γ_2 of Z^ϕ is γ or not.

If $\gamma_2 = \gamma$, then we conclude from

$$|Z^\phi|_\gamma = |\boldsymbol{C}_2(\vec{\boldsymbol{t}_2})|_\gamma = (\gamma \cdot \#\sigma + \#\vec{\boldsymbol{t}_2}, \overline{\gamma} \cdot \#\sigma + \#\vec{\boldsymbol{t}_2})_\gamma = \#\vec{\boldsymbol{t}_2}$$

If $\gamma_2 \ne \gamma$ then we conclude from

$$|Z^\phi|_\gamma = (\overline{\gamma} \cdot \#\sigma + \#\vec{\boldsymbol{t}_2}, \gamma \cdot \#\sigma + \#\vec{\boldsymbol{t}_2})_\gamma = \#\sigma + \#\vec{\boldsymbol{t}_2} = \#\boldsymbol{C}_2(\vec{\boldsymbol{t}_2})$$

If t^ϕ is top-$\overline{\gamma}$ then the induction hypotheses for t_1 (strict) and t_2 yield:

$$\#(\sigma \to \tau) + \#\vec{\boldsymbol{t}_1} < \sum_{Z \in t_1} |\phi(Z)|_\gamma$$

$$(b \otimes \gamma_2) \cdot \#\sigma + \#\vec{\boldsymbol{t}_2} \le (b \otimes \gamma) \cdot \#\sigma + \sum_{Z \in t_2} |\phi(Z)|_\gamma$$

whereas the right-hand side of the strict inequality to be established is

$$\sum_{Z \in t_1} |\phi(Z)|_\gamma + \sum_{Z \in t_2} |\phi(Z)|_\gamma$$

Distinguish cases on whether γ_1 is γ_2 or not.

$(\gamma_1 = \gamma_2)$ Then we must establish

$$\#\tau + \#\vec{\boldsymbol{t}_1}, \vec{\boldsymbol{t}_2} < \sum_{Z \in t_1} |\phi(Z)|_\gamma + \sum_{Z \in t_2} |\phi(Z)|_\gamma$$

and we make a further case-distinction on whether γ is γ_1 or not.

$(\gamma = \gamma_1)$ Then the second induction hypothesis specialises to

$$\#\vec{\boldsymbol{t}_2} \leq \sum_{Z \in t_2} |\phi(Z)|_\gamma$$

from which we conclude.

$(\gamma \neq \gamma_1)$ Then the second induction hypothesis specialises to

$$\#\sigma + \#\vec{\boldsymbol{t}_2} \leq \sum_{Z \in t_2} |\phi(Z)|_\gamma$$

from which we conclude.

$(\gamma_1 \neq \gamma_2)$ Then we must establish

$$\#\tau + \#\vec{\boldsymbol{t}_1}, \boldsymbol{C}_2(\vec{\boldsymbol{t}_2}) \leq \sum_{Z \in t_1} |\phi(Z)|_\gamma + \sum_{Z \in t_2} |\phi(Z)|_\gamma$$

and, noting $\#\boldsymbol{C}_2(\vec{\boldsymbol{t}_2}) = \#\sigma + \#\vec{\boldsymbol{t}_2}$, we make a further case-distinction on whether γ is γ_1 or γ_2.

$(\gamma = \gamma_1)$ Then the second induction hypothesis specialises to

$$\#\sigma + \#\vec{\boldsymbol{t}_2} \leq \sum_{Z \in t_2} |\phi(Z)|_\gamma$$

from which we conclude.

$(\gamma = \gamma_2)$ Then the second induction hypotheses specialises to

$$\#\vec{\boldsymbol{t}_2} \leq \sum_{Z \in t_2} |\phi(Z)|_\gamma$$

from which we conclude using $\#\tau + \#\sigma \leq \#(\sigma \to \tau)$.

This finishes the verification of the second claim for the case $t = t_1\, t_2$. To verify the third claim for the case, first note that t_1^ϕ is top-γ since by assumption t^ϕ is. We distinguish cases on whether t_2^ϕ is top-γ or not.

If t_2^ϕ is top-γ, then we are in case $\gamma_1 = \gamma = \gamma_2$ of the first claim, and the equality claim entails

$$(b \otimes \gamma) \cdot \#\tau + \#\vec{\boldsymbol{t}_1}, \vec{\boldsymbol{t}_2} = (b \otimes \gamma) \cdot \#\tau + \sum_{Z \in t_1} |\phi(Z)|_\gamma + \sum_{Z \in t_2} |\phi(Z)|_\gamma$$

which, combined with the induction hypotheses of the first claim specialised to this case:

$$\#\vec{\boldsymbol{t}_1} \leq \sum_{Z \in t_1} |\phi(Z)|_\gamma$$

$$\#\vec{\boldsymbol{t}_2} \leq \sum_{Z \in t_2} |\phi(Z)|_\gamma$$

and removing equal summands on the left- and right side, yields $\#\vec{\boldsymbol{t}_i} = \sum_{Z \in t_i} |\phi(Z)|_\gamma$. Therefore

$$|t_i^\phi|_b = (\gamma \cdot \#\tau_i + \#\vec{\boldsymbol{t}_i}, \overline{\gamma} \cdot \#\tau_i + \#\vec{\boldsymbol{t}_i})_b = (b \otimes \gamma) \cdot \#\tau_i + \#\vec{\boldsymbol{t}_i} = (b \otimes \gamma) \cdot \#\tau_i + \sum_{Z \in t_i} |\phi(Z)|_\gamma$$

with $\tau_1 = \sigma \to \tau$ and $\tau_2 = \sigma$, so we may apply the induction hypothesis to the t_i, yielding that both are non-empty patterns, hence $t = t_1\, t_2$ is a non-empty pattern as well.

If t_2^ϕ is not top-γ, then we are in the case $\gamma_1 = \gamma \neq \gamma_2$ of the first claim, and the equality claim entails

$$(b \otimes \gamma) \cdot \#\tau + \#\vec{\boldsymbol{t}_1}, \boldsymbol{C}_2(\vec{\boldsymbol{t}_2}) = (b \otimes \gamma) \cdot \#\tau + \sum_{Z \in t_1} |\phi(Z)|_\gamma + \sum_{Z \in t_2} |\phi(Z)|_\gamma$$

which, combined with the induction hypotheses of the first claim specialised to this case:

$$\#\vec{\boldsymbol{t}_1} \leq \sum_{Z \in t_1} |\phi(Z)|_\gamma$$

$$\#\sigma + \#\vec{\boldsymbol{t}_2} \leq \sum_{Z \in t_2} |\phi(Z)|_\gamma$$

yields $\#\vec{\boldsymbol{t}_1} = \sum_{Z \in t_1} |\phi(Z)|_\gamma$ and $\#\sigma + \#\vec{\boldsymbol{t}_2} = \sum_{Z \in t_2} |\phi(Z)|_\gamma$. Therefore

$$|t_1^\phi|_b = (b \otimes \gamma) \cdot \#(\sigma \to \tau) + \#\vec{\boldsymbol{t}_1} = (b \otimes \gamma) \cdot \#(\sigma \to \tau) + \sum_{Z \in t_1} |\phi(Z)|_\gamma$$

so we may apply the induction hypothesis to t_1, yielding t_1 is a pattern. If t_2 is empty, then we are done. The case that t_2 is not empty cannot occur since then

$$|t_2^\phi|_b = (b \otimes \gamma) \cdot \#\sigma + \#\vec{\boldsymbol{t}_2} = (b \otimes \gamma - 1) \cdot \#\sigma + \sum_{Z \in t_2} |\phi(Z)|_\gamma$$

which, when instantiated with b such that $b \otimes \gamma = 1$, contradicts the second claim for t_2:

$$|t_2^\phi|_b < \sum_{Z \in t_2} |\phi(Z)|_\gamma$$

∎

Proof of 'equality implies no recombination' Lemma 6.17. We claim $\boldsymbol{t} \Rightarrow \boldsymbol{s}$ if $t = l^\phi$ and $s = r^\phi$, for any rule $l \to r$ with l, r of type τ, and any substitution ϕ such that $|l^\phi| = |r^\phi|$. Assuming the claim holds the result follows by induction on the derivation of the $\to$-step as follows.

Suppose $t\,u \to s\,u$ with $\boldsymbol{t} = \boldsymbol{C}(\vec{\boldsymbol{t}})$, $\boldsymbol{s} = \boldsymbol{D}(\vec{\boldsymbol{s}})$ both of type $\sigma \to \tau$ and $\boldsymbol{u} = \boldsymbol{E}(\vec{\boldsymbol{u}}) : \sigma$. The assumption $|t\,u| = |s\,u|$ combined with that both $t\,u$ and $s\,u$ have the same type τ implies that both must be top-γ for some color γ, by the observation after Remark 6.9, and therefore that both t and s are top-γ. We distinguish cases on whether the color δ of E is γ or not.

If $\delta = \gamma$ then by Definition 6.4 the interpretations of the source and target of the step are $(\boldsymbol{C}\,\boldsymbol{E})(\vec{\boldsymbol{t}}, \vec{\boldsymbol{u}})$ and $(\boldsymbol{D}\,\boldsymbol{E})(\vec{\boldsymbol{s}}, \vec{\boldsymbol{u}})$. The assumption $|t\,u| = |s\,u|$ combined with Definition 6.7 yield, $\#\vec{\boldsymbol{t}} = \#\vec{\boldsymbol{s}}$ hence $|t| = |s|$. By the induction hypothesis for $t \to s$, $\boldsymbol{t} \Rightarrow \boldsymbol{s}$ and we distinguish cases on whether this is a head-step or not.

- If $\boldsymbol{t} \Rightarrow \boldsymbol{s}$ because it is an instance of the rule $\boldsymbol{C}(\vec{Z}) \to \boldsymbol{D}(\vec{W})$, then $C[\vec{Z}] \to D[\vec{W}]$. Then $(\boldsymbol{C}\,\boldsymbol{E})(\vec{\boldsymbol{t}}, \vec{\boldsymbol{u}}) \Rightarrow (\boldsymbol{D}\,\boldsymbol{E})(\vec{\boldsymbol{s}}, \vec{\boldsymbol{u}})$ is an instance of the rule $(\boldsymbol{C}\,\boldsymbol{E})(\vec{Z}, \vec{X}) \to (\boldsymbol{D}\,\boldsymbol{E})(\vec{W}, \vec{X})$ for fresh meta-variables $\vec{X}$, from which we conclude.
- Suppose $\boldsymbol{t} \Rightarrow \boldsymbol{s}$ because $\boldsymbol{C} = \boldsymbol{D}$ and $\vec{\boldsymbol{t}} = \vec{\boldsymbol{s}}$ except that on i, $\boldsymbol{t}_i \Rightarrow \boldsymbol{s}_i$. Then also $\boldsymbol{C}\,\boldsymbol{E} = \boldsymbol{D}\,\boldsymbol{E}$ and $\vec{\boldsymbol{t}}, \vec{\boldsymbol{u}} = \vec{\boldsymbol{s}}, \vec{\boldsymbol{u}}$ except that on i, $\boldsymbol{t}_i \Rightarrow \boldsymbol{s}_i$, hence $(\boldsymbol{C}\,\boldsymbol{E})(\vec{\boldsymbol{t}}, \vec{\boldsymbol{u}}) \Rightarrow (\boldsymbol{D}\,\boldsymbol{E})(\vec{\boldsymbol{s}}, \vec{\boldsymbol{u}})$.

If $\delta \neq \gamma$ then by Definition 6.4 the interpretations of the source and target of the step are $(\boldsymbol{C}\,\boxtimes)(\vec{\boldsymbol{t}}, \boldsymbol{E}(\vec{\boldsymbol{u}}))$ and $(\boldsymbol{D}\,\boxtimes)(\vec{\boldsymbol{s}}, \boldsymbol{E}(\vec{\boldsymbol{u}}))$. The assumption $|t\,u| = |s\,u|$ combined with Definition 6.7 yield, $\#\vec{\boldsymbol{t}} = \#\vec{\boldsymbol{s}}$ hence $|t| = |s|$. By the induction hypothesis for $t \to s$, $\boldsymbol{t} \Rightarrow \boldsymbol{s}$ and we distinguish cases on whether this is a head-step or not.

- If $\boldsymbol{t} \Rightarrow \boldsymbol{s}$ because it is an instance of the rule $\boldsymbol{C}(\vec{Z}) \to \boldsymbol{D}(\vec{W})$, then $C[\vec{Z}] \to D[\vec{W}]$. Then $(\boldsymbol{C}\,\boxtimes)(\vec{\boldsymbol{t}}, \boldsymbol{E}(\vec{\boldsymbol{u}})) \Rightarrow (\boldsymbol{D}\,\boxtimes)(\vec{\boldsymbol{s}}, \boldsymbol{E}(\vec{\boldsymbol{u}}))$ is an instance of the rule $(\boldsymbol{C}\,\boxtimes)(\vec{Z}, X) \to (\boldsymbol{D}\,\boxtimes)(\vec{W}, X)$ for a fresh meta-variable X, from which we conclude.
- Suppose $\boldsymbol{t} \Rightarrow \boldsymbol{s}$ because $\boldsymbol{C} = \boldsymbol{D}$ and $\vec{\boldsymbol{t}} = \vec{\boldsymbol{s}}$ except that on i, $\boldsymbol{t}_i \Rightarrow \boldsymbol{s}_i$. Then also $\boldsymbol{C}\,\boxtimes = \boldsymbol{D}\,\boxtimes$ and $\vec{\boldsymbol{t}}, \boldsymbol{E}(\vec{\boldsymbol{u}}) = \vec{\boldsymbol{s}}, \boldsymbol{E}(\vec{\boldsymbol{u}})$ except that on i, $\boldsymbol{t}_i \Rightarrow \boldsymbol{s}_i$, hence $(\boldsymbol{C}\,\boxtimes)(\vec{\boldsymbol{t}}, \boldsymbol{E}(\vec{\boldsymbol{u}})) \Rightarrow (\boldsymbol{D}\,\boxtimes)(\vec{\boldsymbol{s}}, \boldsymbol{E}(\vec{\boldsymbol{u}}))$.

Next, suppose the step takes place on the right $u\,t \to u\,s$, with $\boldsymbol{u} = \boldsymbol{E}(\vec{\boldsymbol{u}}) : \sigma \to \tau$ of color δ, $\boldsymbol{t} = \boldsymbol{C}(\vec{\boldsymbol{t}})$ of color γ and $\boldsymbol{s} = \boldsymbol{D}(\vec{\boldsymbol{s}})$ of color ζ both of type σ. We distinguish cases on whether the color δ is γ first, and on whether γ is ζ next.

If $\delta = \gamma = \zeta$ then by Definition 6.4 the interpretations of the source and target of the step are $(\boldsymbol{E}\,\boldsymbol{C})(\vec{\boldsymbol{u}}, \vec{\boldsymbol{t}})$ and $(\boldsymbol{E}\,\boldsymbol{D})(\vec{\boldsymbol{u}}, \vec{\boldsymbol{s}})$. The assumption $|u\,t| = |u\,s|$ combined with Definition 6.7 yield, $\#\vec{\boldsymbol{t}} = \#\vec{\boldsymbol{s}}$ hence $|t| = |s|$. By the induction hypothesis for $t \to s$, $\boldsymbol{t} \Rightarrow \boldsymbol{s}$ and we distinguish cases on whether this is a head-step or not.

- If $\boldsymbol{t} \Rightarrow \boldsymbol{s}$ because it is an instance of the rule $\boldsymbol{C}(\vec{Z}) \to \boldsymbol{D}(\vec{W})$, then $C[\vec{Z}] \to D[\vec{W}]$. Then $(\boldsymbol{E}\,\boldsymbol{C})(\vec{\boldsymbol{u}}, \vec{\boldsymbol{t}}) \Rightarrow (\boldsymbol{E}\,\boldsymbol{D})(\vec{\boldsymbol{u}}, \vec{\boldsymbol{s}})$ is an instance of the rule $(\boldsymbol{E}\,\boldsymbol{C})(\vec{X}, \vec{Z}) \to (\boldsymbol{E}\,\boldsymbol{D})(\vec{X}, \vec{W})$ for fresh meta-variables $\vec{X}$, from which we conclude.
- Suppose $\boldsymbol{t} \Rightarrow \boldsymbol{s}$ because $\boldsymbol{C} = \boldsymbol{D}$ and $\vec{\boldsymbol{t}} = \vec{\boldsymbol{s}}$ except that on i, $\boldsymbol{t}_i \Rightarrow \boldsymbol{s}_i$. Then also $\boldsymbol{E}\,\boldsymbol{C} = \boldsymbol{E}\,\boldsymbol{D}$ and $\vec{\boldsymbol{u}}, \vec{\boldsymbol{t}} = \vec{\boldsymbol{u}}, \vec{\boldsymbol{s}}$ except that on i, $\boldsymbol{t}_i \Rightarrow \boldsymbol{s}_i$, hence $(\boldsymbol{E}\,\boldsymbol{C})(\vec{\boldsymbol{u}}, \vec{\boldsymbol{t}}) \Rightarrow (\boldsymbol{E}\,\boldsymbol{D})(\vec{\boldsymbol{u}}, \vec{\boldsymbol{s}})$.

If $\delta \neq \gamma = \zeta$ then by Definition 6.4 the interpretations of the source and target of the step are $(\boldsymbol{E}\,\boxtimes)(\vec{\boldsymbol{u}}, \boldsymbol{C}(\vec{\boldsymbol{t}}))$ and $(\boldsymbol{E}\,\boxtimes)(\vec{\boldsymbol{u}}, \boldsymbol{D}(\vec{\boldsymbol{s}}))$. The assumption that $|u\,t| = |u\,s|$ and that C and D have the same type, combined with Definition 6.7 yield, $\#\vec{\boldsymbol{t}} = \#\vec{\boldsymbol{s}}$ hence $|t| = |s|$. By the induction hypothesis for $t \to s$, $\boldsymbol{t} \Rightarrow \boldsymbol{s}$ hence $(\boldsymbol{E}\,\boxtimes)(\vec{\boldsymbol{u}}, \boldsymbol{t}) \Rightarrow (\boldsymbol{E}\,\boxtimes)(\vec{\boldsymbol{u}}, \boldsymbol{s})$.

If $\delta = \gamma \neq \zeta$ would hold, then by Definition 6.4 the interpretations of the source and target of the step would be $(\boldsymbol{E}\,\boldsymbol{C})(\vec{\boldsymbol{u}}, \vec{\boldsymbol{t}})$ and $(\boldsymbol{E}\,\boxtimes)(\vec{\boldsymbol{u}}, \boldsymbol{D}(\vec{\boldsymbol{s}}))$. The assumption that $|u\,t| = |u\,s|$ and that C and D have the same type, combined with Definition 6.7 then would yield $\#\vec{\boldsymbol{t}} = \#\sigma + \#\vec{\boldsymbol{s}}$. However, the assumption $t \to s$ combined with Lemma 6.15 and the observation after Remark 6.9 yield $|t| > |s|$ hence, using that t and s have distinct top-colors, $\#\vec{\boldsymbol{t}} > \#\sigma + \#\vec{\boldsymbol{s}}$. Contradiction.

If $\delta \neq \gamma \neq \zeta$ would hold, then by Definition 6.4 the interpretations of the source and target of the step would be $(\boldsymbol{E}\,\boxtimes)(\vec{\boldsymbol{u}}, \boldsymbol{C}(\vec{\boldsymbol{t}}))$. and $(\boldsymbol{E}\,\boldsymbol{D})(\vec{\boldsymbol{u}}, \vec{\boldsymbol{s}})$. The assumption that $|u\,t| = |u\,s|$ and that C and D have the same type, combined with Definition 6.7 then would yield $\#\sigma + \#\vec{\boldsymbol{t}} = \#\vec{\boldsymbol{s}}$. However, the assumption $t \to s$ combined with Lemma 6.15 and the observation after Remark 6.9 yield $|t| > |s|$ hence, using that t and s have distinct top-colors, $\#\sigma + \#\vec{\boldsymbol{t}} > \#\vec{\boldsymbol{s}}$. Contradiction.

This concludes the proof of the lemma based on the claim. It remains to prove the claim, which we do now.

Let ψ and χ be obtained by decomposing the substitution ϕ into γ and $\overline{\gamma}$-components. Formally, ψ and χ such that $\phi = \psi^\chi$, are obtained from ϕ by:

$$\begin{array}{lllllll} \psi(Z) & = & E[\vec{W}_Z] & \chi(W_{Z,i}) & = & t_i & \text{if } \boldsymbol{\phi}(Z) = \boldsymbol{E}(\vec{\boldsymbol{t}}) \text{ and } E \text{ has color } \gamma \\ \psi(Z) & = & W_Z & \chi(Z_Z) & = & \phi(Z) & \text{otherwise} \end{array}$$

Defining $\hat{l} = l^\psi$ and $\hat{r} = r^\psi$, we have by construction that $\hat{l}$ is a non-empty pattern the symbols of which have color γ, so by the same reasoning as for Equation 6.1, the assumption $|t| = |s|$ yields:

$$|t|_b = |\hat{l}^\chi|_b = (b \otimes \gamma) \cdot \#\tau + \sum_{W \in \hat{l}} |\chi(W)|_\gamma = (b \otimes \gamma) \cdot \#\tau + \sum_{W \in \hat{r}} |\chi(W)|_\gamma = |\hat{r}^\chi|_b = |s|_b$$

(From this and the assumption that rules are non-duplicating, it follows that in fact each variable must occur the same number of times in $\hat{l}$ and $\hat{r}$.) From this and Lemma 6.12 it follows that also $\hat{r}$ is a non-empty pattern with symbols of color γ, since by construction r^ψ has color γ and all $\chi(W)$ are top-$\overline{\gamma}$. Therefore, to the $\to$-step $\hat{l} \to \hat{r}$ the $\Rightarrow$-rule $\boldsymbol{C}(\vec{Z}) \to \boldsymbol{D}(\vec{W})$ is associated, for the components C and D and vectors of variables $\vec{Z}$ and $\vec{W}$, such that $C[\vec{Z}] = \hat{l}$ and $D[\vec{W}] = \hat{r}$, and the claim follows: $\boldsymbol{t} = \boldsymbol{C}(\boldsymbol{\chi}(\vec{Z})) \Rightarrow \boldsymbol{D}(\boldsymbol{\chi}(\vec{W})) = \boldsymbol{s}$. ∎

Proof of termination of the HRS of Counterexample 6.23. We prove the termination claim by induction on t and by cases on its head-symbol.

- If $t = x$, then x^ϕ is terminating by the assumption on t;
- If $t = f(t_1^\phi, t_2^\phi, t_3^\phi)$, then by the induction hypothesis each t_i^ϕ is terminating, hence an infinite reduction must look like
  $t \twoheadrightarrow f(t'_1, t'_2, t'_3) \to g(y.f(y,y,y), t'_3) \twoheadrightarrow g(y.f(y,y,y), t') \to f(t',t',t') \to \dots$
  Since by orthogonality t' cannot reduce to both a and b, $f(t',t',t')$ is in head-normal form, so must be terminating as t' is a reduct of t_3^ϕ, hence is terminating;
- If $t = g(y.t_1^\phi, t_2^\phi)$, then by the induction hypothesis each t_i^ϕ is terminating, hence an infinite reduction must be on the form:
  $t \twoheadrightarrow g(y.t'_1, t'_2) \to t'_1[y := t'_2] \to \dots$
  As $t'_1[y := t'_2]$ is a reduct of $t_1^\phi[y := t_2] = t_1^{\phi \uplus [y := t_2^\phi]}$ (note that by the variable convention y is not free in ϕ) which is terminating by the induction hypothesis for t_1 with substitution $\phi \uplus [y := t_2^\phi]$, which is first-order and terminating by the induction hypothesis for t_2; and
- In all other cases t^ϕ is in head-normal form and termination follows from the induction hypothesis for its arguments. ∎