A proof of Finite Family Developments for Higher-Order Rewriting using a Prefix Property

H. J. Sander Bruggink

Department of Philosophy, Utrecht University
Email: bruggink@phil.uu.nl
Homepage: http://www.phil.uu.nl/~bruggink

Abstract. A prefix property is the property that, given a reduction, the ancestor of a prefix of the target is a prefix of the source. In this paper we prove a prefix property for the class of Higher-Order Rewriting Systems with patterns (HRSs), by reducing it to a similar prefix property of a λ-calculus with explicit substitutions. This prefix property is then used to prove that Higher-order Rewriting Systems enjoy Finite Family Developments. This property states, that reductions in which the creation depth of the redexes is bounded are finite, and is a useful tool to prove various properties of HRSs.

1 Introduction

Higher-order Rewriting Systems (HRSs), as introduced by Nipkow in 1991 [12,10], are a powerful tool to study the metatheory of declarative programming languages, like λProlog and Haskell, on the one hand, and theorem provers and proof checkers, like Isabelle, on the other. Also, many (extensions of) λ-calculi can be encoded as instances of HRSs, so that results obtained for HRSs carry over to other interesting domains.

In this paper, we prove two properties of HRSs where left-hand sides of rule are restricted to be patterns. First we prove a prefix property, by reducing this property to a similar prefix property for a λ-calculus with explicit substitutions. The prefix property says that, given a step, the ancestor of a prefix of the target is a prefix of the source. Consider, as an example, the (first-order) rewriting system with the single rule $\mathsf{f}(x) \to \mathsf{g}(\mathsf{f}(x), x)$ and the step $\mathsf{f}(\mathsf{h}(\mathsf{a})) \to \mathsf{g}(\mathsf{f}(\mathsf{h}(\mathsf{a})), \mathsf{h}(\mathsf{a}))$. Now, $p = \mathsf{g}(\mathsf{f}(\Box), \mathsf{h}(\Box))$ is a prefix of the target. Intuitively, its ancestor is $\mathsf{f}(\mathsf{h}(\Box))$, because $s = \mathsf{f}(\mathsf{h}(\Box)) \to \mathsf{g}(\mathsf{f}(\mathsf{h}(\Box)), \mathsf{h}(\Box)) = t$, and p is contained in t. And indeed, $\mathsf{f}(\Box)$ is a prefix of the source.

Many different prefix properties are possible: we can, e.g., vary in how the notions of prefix and ancestor are formalized, and we may impose additional conditions on the form of the prefixes. Prefix properties are already known for first-order TRSs [2,13] and (a labelling of) the λ-calculus with β-reductions [2], and have many applications, such as (head) needed reductions [13, Chap. 8] and normalization of outermost-fair reductions [13, Chap. 9]. A similar property is proved in Van Daalen's Square Brackets Lemma [14].

The second contribution is that we prove Finiteness of Family Developments (FFD) for HRS, by reducing this property to the prefix property described above. FFD states that each reduction, in which the "creation depth", or *family*, of function symbols is bounded, is finite. The intuition behind the notion of family is that in a step $C[l^\sigma] \to C[r^\sigma]$, the symbols of r depend on the symbols of l, and therefore have a higher creation depth, while the symbols in C and σ do not take part in the step and have the same creation depth in both source and target. For example, consider the following infinite reduction, using the rewrite system above. We label the function symbols with their creation depth.

$$\mathsf{f}^0(\mathsf{a}^0) \to \mathsf{g}^1(\mathsf{f}^1(\mathsf{a}^0), \mathsf{a}^0) \to \mathsf{g}^1(\mathsf{g}^2(\mathsf{f}^2(\mathsf{a}^0), \mathsf{a}^0)) \to \mathsf{g}^1(\mathsf{g}^2(\mathsf{g}^3(\mathsf{f}^3(\mathsf{a}^0), \mathsf{a}^0), \mathsf{a}^0)) \to \cdots$$

Clearly, in this infinite reduction, the creation depth of the f's grows without bound. FFD states that restricting the creation depth to a finite number, yields finite reductions. FFD is a useful tool to prove various properties of rewrite systems, such as termination (e.g. termination of simply typed λ-calculus follows from FFD, cf. [7, page 31]), existence of standard reductions, etc.

Of some lemmas and theorems the proof is omitted or only sketched. Full proofs are made available in the technical report [4].

Acknowledgements. I wish to thank Vincent van Oostrom, Delia Kesner and the anonymous referees for useful remarks on preliminary versions of this paper.

2 Preliminaries

We presuppose knowledge of the simply typed λ-calculus. Here we give a short overview of Higher-Order Rewrite Systems (HRSs) [10]. In particular, we consider HRSs as HORSs [15] with the simply typed λ-calculus as substitution calculus. We refer to [13, Sect. 11.2] for a good introduction.

Simple types are generated from a set of base types by the type constructor $\to$. Let Σ be a signature of simply typed function symbols. We define a *preterm* to be a simply typed λ-terms over Σ. We want to consider $\beta\eta$-equivalence classes of preterms. Since it is well known that β-reduction, combined with restricted η-expansion ($\overline{\eta}$-reduction), is confluent and terminating, we take $\beta\overline{\eta}$-normal forms as unique representatives of the $\beta\overline{\eta}$-equivalence classes. We define: a *term* is a preterm in $\beta\overline{\eta}$-normal form. In the following, s, t will range over terms (and, whenever indicated, over preterms as well).

A sequence $a_1, \ldots, a_n$ will sometimes be written as $\overline{a_n}$, or just $\overline{a}$ if the length is not important or clear from the context. Juxtaposition of two sequences denotes concatenation.

For terms or preterms $s, t_1, \ldots, t_n$, we write $s(t_1, \ldots, t_n)$ for $st_1 \cdots t_n$, or, in the case of terms, the β-normal form thereof. We also introduce the shorthand $\lambda\overline{x_n}.s$ for $\lambda x_1. \ldots \lambda x_n.s$. With $\mathrm{FV}(s)$ we denote the set of free variables of term or preterm s, and with $\mathit{Sym}(s)$ the set of function symbols present in the term or preterm. If $\lambda\overline{x}.a(\overline{s})$ is a term, then a is called the *head* of that term (a is a function symbol or variable).

In the class of HRSs that we consider, the left-hand sides of rules are restricted to be *local patterns.* For patterns, unification is decidable and unique most general unifiers exist [11]. Local patterns, additionally, are linear (each free variable occurs at most once) and fully-extended (free variable have all bound variables in scope as argument). These extra requirements have a similar purpose as the requirement of left-linearity in first-order TRS: they keep matching local. To match a non-linear pattern, it is possible that subterms outside the pattern need to be checked for equality; to match a non fully extended pattern, it is possible that such a subterm must be checked for the non-occurrence of a variable. Because the notion of pattern depends on what the free variables are, we need to parametrize the notion with a context of variables, and obtain the following inductive definition:

Definition 2.1 (Pattern). *Let $\overline{x}$ be a sequence of variables.*

(i) A term s is an $\overline{x}$-pattern if:
  - *$s = a(s_1, \dots, s_n)$ and either $a \in \overline{x} \cup \Sigma$ and $s_1, \dots, s_n$ are $\overline{x}$-patterns; or $s_1, \dots, s_n$ is η-equivalent to a sequence of distinct variables from $\overline{x}$.*
  - *$s = \lambda y.s_0$ and s_0 is an $\overline{x}y$-pattern.*

(ii) A term s is linear outside $\overline{x}$*, if each free variable which is* not *in $\overline{x}$, occurs in it at most once. A term s is a* fully extended $\overline{x}$-pattern*, if, in the second case of the above definition, $s_1, \dots, s_n =_\eta \overline{x}$. A term s is a* local $\overline{x}$-pattern*, if s is linear outside $\overline{x}$ and a fully extended $\overline{x}$-pattern.*

Examples of local patterns are $\mathsf{f}(x)$, $\mathsf{g}(\lambda xy.\mathsf{f}(z(x, y)))$ and $\mathsf{h}(\lambda x.z(x))$. Examples of non-local patterns are $\mathsf{g}(\lambda xy.\mathsf{f}(y))$ (not fully-extended) and $\mathsf{h}(\lambda x.z(x), \lambda x.z(x))$ (not linear). An example of a non-pattern is $\mathsf{g}(z(\mathsf{a}))$.

In the following, p, q will range over patterns, and the word pattern (without the sequence of variables) will refer to a $\emptyset$-pattern.

Definition 2.2 (HRS). *A* rewrite rule *(for a signature Σ) is a pair $\lambda\overline{x}.l_0 \to \lambda\overline{x}.r_0$ of closed Σ-terms of the same type, such that $l_0 = f(s_1, \dots, s_n)$ and l_0 is a local pattern not η-equivalent to a variable. An HRS is a tuple $\mathcal{H} = \langle \Sigma, R \rangle$, where Σ is a signature and R a set of rewrite rules for Σ.*

The rewrite relation $\to_\mathcal{H}$ is defined as follows: $s \to_\mathcal{H} t$ if there exist a context C such that $s =_\beta C[l]$ and $t =_\beta C[r]$, for some rule $l \to r \in R$.

For arbitrary rewrite system $\mathcal{R}$, we denote with $\twoheadrightarrow_\mathcal{R}$ the reflexive, transitive closure of $\to_\mathcal{R}$.

Note that there is no substitution in the definition of the rewrite relation, such as in first-order term rewriting systems (but see also Remark 2.4). The leading abstractions of the rules take the role of the substitution, as can be seen in the next example:

Example 2.3. Let the HRS $\mathcal{M}ap$, implementing the higher-order function map, be defined by:

$$\lambda z.\mathsf{map}(\lambda x.z(x), \mathsf{nil}) \to \lambda z.\mathsf{nil}$$
$$\lambda zuv.\mathsf{map}(\lambda x.z(x), \mathsf{cons}(u, v)) \to \lambda zuv.\mathsf{cons}(z(\mathsf{e}(u)), \mathsf{map}(\lambda x.z(x), v))$$

Here, cons and nil are the list constructors, viz. list composition and the empty list, respectively. The reason for the symbol e is to make the HRS non-collapsing (see Def. 2.5). A reduction of two $\mathcal{M}ap$-steps is the following:

$$\begin{array}{ll}
\multicolumn{2}{l}{\mathsf{map}(\lambda x.\mathsf{f}(x), \mathsf{cons}(\mathsf{a}, \mathsf{nil}))} \\
=_\beta & (\underline{\lambda zuv.\mathsf{map}(\lambda x.z(x), \mathsf{cons}(u, v))})(\lambda x'.\mathsf{f}(x'), \mathsf{a}, \mathsf{nil}) \\
\rightarrow_{\mathcal{M}ap} & (\underline{\lambda zuv.\mathsf{cons}(z(\mathsf{e}(u)), \mathsf{map}(\lambda x.z(x), v))})(\lambda x'.\mathsf{f}(x'), \mathsf{a}, \mathsf{nil}) \\
=_\beta & \mathsf{cons}(\mathsf{f}(\mathsf{e}(\mathsf{a})), \mathsf{map}(\lambda x.\mathsf{f}(x), \mathsf{nil})) \\
=_\beta & \mathsf{cons}(\mathsf{f}(\mathsf{e}(\mathsf{a})), (\underline{\lambda z.\mathsf{map}(\lambda x.z(x), \mathsf{nil})})(\lambda x'.\mathsf{f}(x'))) \\
\rightarrow_{\mathcal{M}ap} & \mathsf{cons}(\mathsf{f}(\mathsf{e}(\mathsf{a})), (\underline{\lambda z.\mathsf{nil}})(\lambda x'.\mathsf{f}(x'))) \\
=_\beta & \mathsf{cons}(\mathsf{f}(\mathsf{e}(\mathsf{a})), \mathsf{nil})
\end{array}$$

Note how the (underlined) left-hand sides are *literally* replaced by the (also underlined) right-hand sides.

In later examples, the leading abstractions of rewrite rules will be omitted; in other words, we will write $l \rightarrow r$ for $\lambda\overline{x}.l \rightarrow \lambda\overline{x}.r$.

Substitutions are maps from variables to terms. Application of a substitution $\sigma = [x_1 \mapsto t_1, \ldots, x_n \mapsto t_n]$ to a term s is defined as: $s^\sigma = (\lambda x_1 \ldots x_n.s)t_1 \ldots t_n$ (where this term is, as always, implicitly reduced to $\beta\overline{\eta}$-normal form). In the following, $\rho, \sigma, \tau, \upsilon$ will rangle over substitutions. The composition of substitions σ and τ is denoted by $\sigma\,;\tau$, where $s^{\sigma;\tau} = (s^\sigma)^\tau$. A substitution is called linear, if each free variable occurs in its codomain at most once, i.e. if all terms of its codomain are linear and have mutually disjoint free variables. A (fully extended) $\overline{x}$-pattern substitution is a substitution of which the codomain consists of (fully extended) $\overline{x}$-patterns.

Remark 2.4. The rewrite relation of Def. 2.2 can alternatively, and more in the fashion of first-order TRSs, be defined in the following way: $s \rightarrow_{\mathcal{H}} t$ if $s =_\beta C[l_0^\sigma]$ and $t =_\beta C[r_0^\sigma]$, where $\lambda\overline{x}.l_0 \rightarrow \lambda\overline{x}.r_0 \in R$ and σ is a substitution with $Dom(\sigma) = \overline{x}$. This alternative definition, however, requires the notion of substitution to be defined, and therefore we prefer the other one. In the rest of the paper, we will sometimes implicitly switch definitions.

Intuitively, a rewrite rule is collapsing, if it can bring context and subtitution, or different parts of the substitution, together, i.e. if, after the application of the rule, a function symbol of the context can be directly connected to a function symbol of the substitution. This can happen in two specific cases, which we will use as a definition:

Definition 2.5. *A term s is* collapsing, *if one of the following applies:*

- (context-subst): $s = x(s_1, \ldots, s_n)$, *where x is a free variable; or*
- (subst-subst): $s = C[x(s_1, \ldots, s_n)]$, *and for some k, $s_k = \lambda\overline{z}.y(t_1, \ldots, t_m)$, where C is a context, x is a free variable, and y a free or bound variable.*

A rewrite rule $\lambda\overline{x}.l \rightarrow \lambda\overline{x}.r$ is collapsing, if r is collapsing, and an HRS is collapsing, if at least one of its rules is.

Example 2.6.

– The rules $\lambda x.\mathsf{f}(x) \to \lambda x.x$ and $\lambda z.\mathsf{mu}(\lambda x.z(x)) \to \lambda z.z(\mathsf{mu}(\lambda x.z(x)))$ are collapsing due to the (context-subst) condition.
– The rule $\lambda yz.\mathsf{g}(\lambda x.z(x), y) \to \lambda yz.\mathsf{f}(z(y))$ is collapsing due to the (subst-subst) condition.
– The rule $\lambda yz.\mathsf{app}(\mathsf{lam}(\lambda x.z(x)), y) \to \lambda yz.z(y)$ is collapsing due to both the (context-subst) and the (subst-subst) conditions.

3 Labelling HRSs with natural numbers

Labelling rewriting systems is a well-known method to formalize the notion of redex family; see e.g. [8,9]. In this section, we develop a labelling, in the sense of [17,13], for HRSs, analogous to the labelling for the λ-calculus used by Hyland [6] and Wadsworth [18]. Each function symbol is labelled by a natural number, representing the "creation depth" of the function symbol, and the rules are labelled such that every function symbol of the right-hand side is labelled with the largest label of the left-hand side plus one.

Definition 3.1 (ω-labelling).

(i) *The ω-labelling of a signature Σ is defined as: $\Sigma^\omega = \{f^\ell \mid f \in \Sigma, \ell \in \mathbb{N}\}$.*
(ii) *The family of a term s, denoted* $\mathrm{fam}(s)$*, is the largest label of s, i.e.:* $\mathrm{fam}(s) = \max\{\ell \mid f^\ell \in \mathit{Sym}(s)\}$
(iii) *Let s be a term, and $\ell \in \mathbb{N}$ a label. Then:*

$$\begin{aligned} x(s_1,\ldots,s_n)^\ell &= x(s_1^\ell,\ldots,s_n^\ell) \\ f(s_1,\ldots,s_n)^\ell &= f^\ell(s_1^\ell,\ldots,s_n^\ell) \\ (\lambda x.s_0)^\ell &= \lambda x.s_0^\ell \end{aligned}$$

(iv) *The projection operation $|\cdot|_\omega$ is the mapping from Σ^ω to Σ given by $|f^\ell|_\omega = f$. The mapping is homomorphically extended to terms.*
(v) *Let $\mathcal{H} = \langle \Sigma, R\rangle$. The ω-labelled version of $\mathcal{H}$ is defined as: $\mathcal{H}^\omega = \langle \Sigma^\omega, R^\omega\rangle$, where R^ω consist of all rules $l' \to r^{(\mathrm{fam}(l')+1)}$ such that $l \to r \in R$ and $|l'|_\omega = l$.*

The ω-labelling only labels *function symbols*, not variables, abstractions or applications. The reason for this is that we want the ω-labelling of an HRS to be an HRS itself (otherwise it would not be a labelling in the sense of [17,13]). Labelling variables is impossible, because α-equivalent terms are identified. Labelling abstractions and applications is impossible because we have fixed the (unlabelled) simply typed λ-calculus as substitution calculus.

Example 3.2. The labelled HRS $\mathcal{M}ap^\omega$ consists, among others, of the rules:

$$\begin{aligned} \mathsf{map}^0(\lambda x.z(x), \mathsf{nil}^0) &\to \mathsf{nil}^1 \\ \mathsf{map}^1(\lambda x.z(x), \mathsf{nil}^1) &\to \mathsf{nil}^2 \\ \mathsf{map}^0(\lambda x.z(x), \mathsf{cons}^0(u,v)) &\to \mathsf{cons}^1(z(\mathsf{e}^1(u)), \mathsf{map}^1(\lambda x.z(x), v)) \\ \mathsf{map}^0(\lambda x.z(x), \mathsf{cons}^1(u,v)) &\to \mathsf{cons}^2(z(\mathsf{e}^2(u)), \mathsf{map}^2(\lambda x.z(x), v)) \end{aligned}$$

A labelled reduction corresponding to the reduction of Ex. 2.3 is the following:

$$\begin{array}{ll}
& \mathsf{map}^0(\lambda x.\mathsf{f}^0(x), \mathsf{cons}^0(\mathsf{a}^0, \mathsf{nil}^0)) \\
=_\beta & (\underline{\lambda zuv.\mathsf{map}^0(\lambda x.z(x), \mathsf{cons}^0(u, v)))}(\lambda x'.\mathsf{f}^0(x'), \mathsf{a}^0, \mathsf{nil}^0) \\
\rightarrow_{\mathcal{M}ap} & (\underline{\lambda zuv.\mathsf{cons}^1(z(\mathsf{e}^1(u)), \mathsf{map}^1(\lambda x.z(x), v)))})(\lambda x'.\mathsf{f}^0(x'), \mathsf{a}^0, \mathsf{nil}^0) \\
=_\beta & \mathsf{cons}^1(\mathsf{f}^0(\mathsf{e}^1(\mathsf{a}^0)), \mathsf{map}^1(\lambda x.\mathsf{f}^0(x), \mathsf{nil}^0)) \\
=_\beta & \mathsf{cons}^1(\mathsf{f}^0(\mathsf{e}^1(\mathsf{a}^0)), (\underline{\lambda z.\mathsf{map}^1(\lambda x.z(x), \mathsf{nil}^0))}(\lambda x'.\mathsf{f}^0(x'))) \\
\rightarrow_{\mathcal{M}ap} & \mathsf{cons}^1(\mathsf{f}^0(\mathsf{e}^1(\mathsf{a}^0)), (\underline{\lambda z.\mathsf{nil}^2})(\lambda x'.\mathsf{f}^0(x'))) \\
=_\beta & \mathsf{cons}^1(\mathsf{f}^0(\mathsf{e}^1(\mathsf{a}^0)), \mathsf{nil}^2)
\end{array}$$

Notice how only the labels of function symbols involved in the step (i.e. the underlined ones) are increased.

The following two lemmas provide a correspondence between labelled and unlabelled reductions, and are easily proved by induction:

Lemma 3.3. *Let $\mathcal{H}$ be an HRS. $\mathcal{H}^\omega$ is orthogonal/collapsing/erasing, if and only if $\mathcal{H}$ is.*

Lemma 3.4. *Let $\mathcal{H}$ be an HRS.*

(i) If $s \rightarrow_{\mathcal{H}} t$, then, for each s' such that $|s'|_\omega = s$, there is a t' such that $s' \rightarrow_{\mathcal{H}^\omega} t'$ and $|t'|_\omega = t$.
(ii) If $s \rightarrow_{\mathcal{H}^\omega} t$, then $|s|_\omega \rightarrow_{\mathcal{H}} |t|_\omega$.

4 The Prefix Property

We call p a prefix of term t, if it is a pattern, and there exists a substitution σ such that $p^\sigma = t$. Given a step $s \rightarrow t$, a subterm q of s is the ancestor of a subterm p of t, if the symbols of t "trace to" the symbols of s. This notion is formalized here using labelling together with the rewrite relation: q is an ancestor of p, if $\mathrm{fam}(p) \geq \mathrm{fam}(q)$ and $q \twoheadrightarrow_{\mathcal{H}^\omega} p^\upsilon$. The substitution υ is necessary because q might reduce to a "bigger" term than p; typically, υ has only function symbols which are also in p. Using these formalizations, we prove in this section the following theorem (proof begins on page 12).

Theorem 4.1 (Prefix Property). *Let $\mathcal{H}^\omega$ be the ω-labelling of a non-collapsing HRS, s a term, p a local $\overline{x}$-pattern and σ a substitution. If $s \rightarrow_{\mathcal{H}^\omega} p^\sigma$, then there exist a local $\overline{x}$-pattern q and a substitution τ, such that $s = q^\tau$, $\mathrm{fam}(p) \geq \mathrm{fam}(q)$, and either:*

– *$q \rightarrow_{\mathcal{H}^\omega} p^\upsilon$, for some substitution υ such that $\upsilon\,;\tau = \sigma$; or* *(trm)*
– *$q = p$ and $\tau \rightarrow_{\mathcal{H}^\omega} \sigma$.* *(sub)*

The theorem states that, given a prefix of the target, its ancestor is a prefix of the source. There are two possibilities: either the prefix takes part in the step, or the step occurred fully in the substitution. Note that, in the first case, we do not only require that its ancestor is a prefix, but also that the suffix stays the same (except for duplicated subterms). In this regard, the lemma is stronger than e.g. the prefix property (for the λ-calculus) proved in [2, Prop. 7.4].

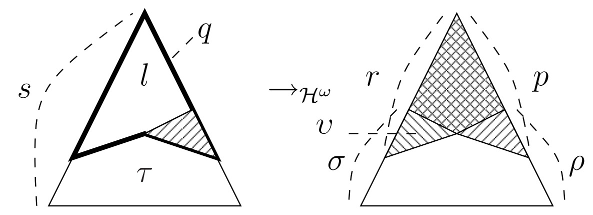

Fig. 1. The interesting case in the proof of the Prefix Property for HRSs

Example 4.2. Consider the following $\mathcal{M}ap^\omega$-step (see page 3):

$$\mathsf{h}^1(\mathsf{map}^3(\lambda x.\mathsf{f}^2(x), \mathsf{cons}^2(\mathsf{a}^5, \mathsf{nil}^1)) \to \mathsf{h}^1(\mathsf{cons}^4(\mathsf{f}^2(\mathsf{e}^4(\mathsf{a}^5)), \mathsf{map}^4(\lambda x.\mathsf{f}^2(x), \mathsf{nil}^1)))$$

First, let the prefix $p = \mathsf{h}^1(\mathsf{cons}^4(\mathsf{f}^2(y_1), y_2))$ of the target be given. The suffix is then given by $\sigma = [y_1 \mapsto \mathsf{e}^4(\mathsf{a}^5), y_2 \mapsto \mathsf{map}^4(\lambda x.\mathsf{f}^2(x), \mathsf{nil}^1)]$. Then:

$$\begin{aligned} q &= \mathsf{h}^1(\mathsf{map}^3(\lambda x.\mathsf{f}^2(x), \mathsf{cons}^2(y_1, z_2))) \\ \upsilon &= [y_1 \mapsto \mathsf{e}^4(z_1), y_2 \mapsto \mathsf{map}^4(\lambda x.\mathsf{f}(x), z_2)] \\ \tau &= [z_1 \mapsto \mathsf{a}^5, z_2 \mapsto \mathsf{nil}^1] \end{aligned}$$

satisfy the conditions of the (trm) case. Second, let $p = \mathsf{h}(y)$ and $\sigma = [y \mapsto \mathsf{cons}^4(\mathsf{f}^2(\mathsf{e}^4(\mathsf{a}^5)), \mathsf{map}^4(\lambda x.\mathsf{f}^2(x), \mathsf{nil}^1)))]$. Then:

$$q = \mathsf{h}^1(y) \quad \text{and} \quad \tau = [y \mapsto \mathsf{map}^3(\lambda x.\mathsf{f}^2(x), \mathsf{cons}^2(\mathsf{a}^5, \mathsf{nil}^1))]$$

satisfy the conditions of the (sub) case.

The interesting case in the proof of the Prefix Property is the case that the step $s \to_{\mathcal{H}^\omega} p^\sigma$ occurs at the head. In this case we have that $s = l^\rho$ and $p^\sigma = r^\rho$, for some rule $l \to r$ and substitution ρ. This situation is depicted in Fig. 1. We want to construct an ancestor q that satisfies the (trm) case. It makes sense to try to do this by adding to the pattern l the parts of p that are not in r. However, due to the implicit β-conversions, these "parts of p that are not in r" are not easily obtained. The key observation is that the β-reduction from p^σ to normal form is a variable renaming, because p is a pattern and has only bound variables as arguments of free variables. The trick is to translate the prefix and suffix in such a way, that the variable renamings are already carried out (we need variable capturing, first-order substitutions for this, called *graftings*), trace the prefix back over the β-reduction from r^ρ to normal form, and find the prefix's ancestor, which is a prefix of r^ρ. Now, we are dealing with terms that are exactly equal, instead of only equal up to β-equality, and now the problem can be solved by using first-order unification techniques.

The above proof technique suggests that we need to prove a prefix property for β-reductions in the λ-calculus. This is difficult, however, since the λ-calculus does not cope well with graftings, because of the global nature of substitution.

For example, let $C = (\lambda x.\square)\mathsf{a}$, $D = \square$ and $s = x$. Then $C \to_\beta D$, and $C[s] \to_\beta \mathsf{a}$, because the x in s is captured by the abstraction in the context and substituted for. However, $D[s] = x$ and thus $C[s] \not\to_\beta D[s]$. To tackle this problem, we use a λ-calculus with explicit substitutions, a variant of the λx-calculus, and prove a prefix property for it. Then, we simulate β-equality with this new calculus. In [5] a similar approach is taken w.r.t. higher-order unification.

4.1 The Prefix Property of the λx-Calculus

We use a variant of the λx-calculus [3], with explicit renamings. The calculus has both object variables (x, y, z) and metavariables (X, Y, Z). In the following, we will refer to it simply by λx-calculus. The terms of the λx-calculus over some signature Σ are first-order terms given by the following grammar:

$$\Lambda_{\mathrm{x}} := x \mid X \mid f \mid \lambda x.\Lambda_{\mathrm{x}} \mid \Lambda_{\mathrm{x}}\Lambda_{\mathrm{x}} \mid \Lambda_{\mathrm{x}}\{x\backslash\Lambda_{\mathrm{x}}\}$$

where $f \in \Sigma$ and the object variables are considered as constants or names. M, N will range over λx-terms. Terms of the form $M\{x\backslash N\}$ will be called *explicit substitutions*, and the $\{x\backslash N\}$ part of an explicit substitution is called a *closure*. With $\mathrm{MV}(M)$ we will denote the set of metavariables of M, and with $\mathit{Sym}(M)$ the set of function symbols of M. The reduction rules of the λx-calculus are:

$$(\lambda x.M)N \to_{\mathrm{B}} M\{x\backslash N\}$$

$$\begin{array}{ll} x\{x\backslash N\} \to_{\mathrm{x}} N & (\lambda x.M)\{x\backslash N\} \to_{\mathrm{x}} \lambda x.M \\ y\{x\backslash N\} \to_{\mathrm{x}} y & (\lambda y.M)\{x\backslash N\} \to_{\mathrm{x}} \lambda z.M\{y\backslash z\}\{x\backslash N\} \\ f\{x\backslash N\} \to_{\mathrm{x}} f & (M_1 M_2)\{x\backslash N\} \to_{\mathrm{x}} M_1\{x\backslash N\}M_2\{x\backslash N\} \end{array}$$

where $x \neq y$ and z is a fresh object variable. The subcalculus x consists of all rules except the B-rule. The reduction relations $\to_{\mathrm{Bx}}$ and $\to_{\mathrm{x}}$ are the contextual closures of the above steps. Note that there is no reduction rule for terms of the form $X\{x\backslash N\}$, where X is a metavariable, and thus x-normal forms are characterized by the fact that sequences of closures are only applied to metavariables.

A λx-term is called *passive* if no metavariable X occurs in a subterm of the form $X\overline{\mu}(M_1, \ldots, M_n)$, where $\overline{\mu}$ is a sequence of closures; it is called *linear*, if every metavariable occurs in it at most once. In the following P, Q will range over linear, passive λx-terms.

Remark 4.3. It is well-known that the λx-calculus is *not* confluent on terms containing metavariables. At first sight, non-confluence seems problematic, because we're trying to use the λx-calculus to simulate the (confluent) λ-calculus. However, the translation to λ-calculus (see page 10) will remove all closures, and will project normal forms of the same λx-term to the same λ-term (modulo α-equivalence).

A *grafting* is a mapping from metavariables to λx-terms. The greek lowercase letters $\zeta, \eta, \theta, \kappa$ will range over graftings. Applying a grafting ζ to a term M, written $M[\zeta]$, is defined exactly as first order substitution, i.e.:

$$\begin{array}{rlrl} x[\zeta] = x & & (\lambda x.M)[\zeta] = \lambda x.M[\zeta] \\ X[\zeta] = \zeta'(X) & & (M_1 M_2)[\zeta] = M_1[\zeta] M_2[\zeta] \\ f[\zeta] = f & & (M\{x\backslash N\})[\zeta] = M[\zeta]\{x\backslash N[\zeta]\} \end{array}$$

where $\zeta'(X) = \zeta(X)$, if $X \in \mathit{Dom}(\zeta)$, and $\zeta'(X) = X$, otherwise. A grafting is called *linear*, if every metavariable occurs in its codomain only once, i.e. its codomain consists of linear λx-terms with mutually disjoint metavariables. A grafting is called *passive*, if all the terms of its codomain are passive.

Because λx-terms are first-order terms, unification is decidable. In the proof of the Prefix Property, we need the following property: if two λx-terms are unifiable, there exists a most general unifier (mgu). In fact, if we assume the unifiable terms to be linear and passive, then the mgu applied to one of the terms is a linear, passive λx-term again:

Lemma 4.4. *Let M, N be linear λx-terms, where $\mathrm{MV}(M) \cap \mathrm{MV}(N) = \emptyset$, and let ζ, η be graftings such that $M[\zeta] = N[\eta]$. There exist graftings ζ_0, η_0, κ such that $M[\zeta_0] = N[\eta_0]$, $\zeta_0 \,;\, \kappa = \zeta$, $\eta_0 \,;\, \kappa = \eta$, $\mathit{Sym}(\zeta_0) \subseteq \mathit{Sym}(N)$, $\mathit{Sym}(\eta_0) \subseteq \mathit{Sym}(M)$. Moreover, if M (N) is passive, then η_0 (ζ_0) consists of passive λx-terms.*

Proof. (Sketch) Since λx-terms are basically first-order terms, we can use first-order unification techniques. Because of disjointness of the metavariables we can consider the two graftings as one unifier, and the linearity assumption is needed for the condition on the symbols. □

Example 4.5. Let:

$$\begin{array}{rr} M = \lambda x.\mathsf{g}(\mathsf{f}_1(X_1), X_2) & N = \lambda x.\mathsf{g}(Y_1, \mathsf{f}_2(Y_2)) \\ \zeta = [X_1 \mapsto \mathsf{a}, X_2 \mapsto \mathsf{f}_2(\mathsf{a})] & \eta = [Y_1 \mapsto \mathsf{f}_1(\mathsf{a}), Y_2 \mapsto \mathsf{a}] \end{array}$$

Then $M[\zeta] = \lambda x.\mathsf{g}(\mathsf{f}_1(\mathsf{a}), \mathsf{f}_2(\mathsf{a})) = N[\eta]$. We take $\zeta_0 = [X_2 \mapsto \mathsf{f}_2(Z_1)]$, $\eta_0 = [Y_1 \mapsto \mathsf{f}_1(Z_2)]$ and $\kappa = [Z_i \mapsto \mathsf{a}]$ to satisfy the conditions of the lemma.

In the next theorem, we prove the Prefix Property for the λx-calculus. P is a prefix of a λx-term M, if it is a linear, passive λx-term, and there exists a grafing ζ such that $P[\zeta] = M$. The notion of ancestor is again formalized using labelling and the rewrite relation; however, because we do not count creation depth in Bx-reductions, now the labels, or more generally, the function symbols of the prefix must be the same as those of its ancestor. Just like in Theorem 4.1, a prefix can either take part in the step, or not, resulting in two cases. Item (ii) is the extension of the Prefix Property to Bx-*reductions.*

Theorem 4.6 (λx-Prefix Property). *Let M be a closed λx-term, P a linear, passive λx-term and ζ a grafting.*

(i) If $M \to_{\mathrm{Bx}} P[\zeta]$, then there exist a linear, passive λx-term Q and a grafting η such that $M = Q[\eta]$, $\mathit{Sym}(Q) = \mathit{Sym}(P)$ and either:

– $Q \to_{\mathrm{Bx}} P[\kappa]$ *where* κ *is some grafting such that* $\kappa\,;\eta = \zeta$*; or* *(trm)*
– $Q = P$ *and* $\eta \to_{\mathrm{x}} \zeta$. *(sub)*

(ii) If $M \twoheadrightarrow_{\mathrm{Bx}} P[\zeta]$*, then there exist a linear, passive* λx*-term* Q *and a grafting* η *such that:* $M = Q[\eta]$*,* $Sym(Q) = Sym(P)$*,* $Q \twoheadrightarrow_{\mathrm{Bx}} P[\kappa]$ *where* κ *is some grafting such that* $\kappa\,;\eta \twoheadrightarrow_{\mathrm{Bx}} \zeta$.

Proof. (Sketch) Item (i) is proved by induction on the context of the step $M \to_{\mathrm{Bx}} P[\zeta]$, using a case analysis and Lemma 4.4 in the base case, and (ii) by induction on the length of the reduction. □

Example 4.7. Consider the Bx-reduction $(\lambda x.\mathsf{g}(x,x))(\mathsf{f}(\mathsf{a})) \twoheadrightarrow_{\mathrm{Bx}} \mathsf{g}(\mathsf{f}(\mathsf{a}),\mathsf{f}(\mathsf{a}))$, and the prefix $P = \mathsf{g}(\mathsf{f}(X),Y)$ of the target. The suffix is $\zeta = [X \mapsto \mathsf{a}, Y \mapsto \mathsf{f}(\mathsf{a})]$. We can take $Q = (\lambda x.\mathsf{g}(x,x))(\mathsf{f}(Y))$, $\kappa = [Y \mapsto \mathsf{f}(X)]$ and $\eta = [X \mapsto \mathsf{a}]$, satisfying the conditions of Theorem 4.6 (ii).

4.2 Translating between Terms, Preterms and λx-Terms

We are now dealing with three types of terms: terms, preterms and λx-terms. In this section we develop translations between (pre)terms and λx-terms. The "translation" between terms and preterms will be done completely implicitly, here. See [4] for a more detailed approach.

Translating terms. We introduce the operations $\cdot^{\ominus}_{\overline{x}}$ and $\cdot^{\oplus}$, which map λ-terms to λx-terms, and vice versa, as follows:

$$\begin{array}{rlrl}
 & & M^{\oplus} &= (M{\downarrow_{\mathrm{x}}})^{\oplus}_{\mathrm{N}} \\
y^{\ominus}_{\overline{x}} &= Y \text{ if } y \notin \overline{x} & (Y\overline{\sigma})^{\oplus}_{\mathrm{N}} &= y \\
x^{\ominus}_{\overline{x}} &= x \text{ if } x \in \overline{x} & x^{\oplus}_{\mathrm{N}} &= x \\
f^{\ominus}_{\overline{x}} &= f & f^{\oplus}_{\mathrm{N}} &= f \\
(\lambda y.s)^{\ominus}_{\overline{x}} &= \lambda y.s^{\ominus}_{\overline{x}y} & (\lambda y.M)^{\oplus}_{\mathrm{N}} &= \lambda y.M^{\oplus} \\
(s_1 s_2)^{\ominus}_{\overline{x}} &= (s_1)^{\ominus}_{\overline{x}}(s_2)^{\ominus}_{\overline{x}} & (M_1 M_2)^{\oplus}_{\mathrm{N}} &= M_1^{\oplus} M_2^{\oplus}
\end{array}$$

Note that $\cdot^{\oplus}$ also normalizes the term to x-normal form and removes explicit substitutions, and that, for each preterm s and sequence of variables $\overline{x}$, $(s^{\ominus}_{\overline{x}})^{\oplus} = s$. The operations above are naturally generalized to translations between substitutions and graftings.

Lemma 4.8. *Let* M, N *be* λx*-terms.* $M \twoheadrightarrow_{\mathrm{Bx}} N$ *if and only if* $M^{\oplus} \twoheadrightarrow_{\beta} N^{\oplus}$.

Proof. ($\Rightarrow$) and ($\Leftarrow$) are proved by induction on the length of the reductions $M \twoheadrightarrow_{\mathrm{Bx}} N$ and $M^{\oplus} \twoheadrightarrow_{\beta} N^{\oplus}$, respectively. □

Although the above lemma suggests that Bx-reduction in the λx-calculus can easily simulate β-reduction, there is still a problem: $\cdot^{\oplus}$ does not distribute properly over grafting application. The problem is similar to the problem given on page 8. Consider the λx-term $M := (\lambda x.\mathsf{f}(Y))\mathsf{a}$ and grafting $\zeta := [Y \mapsto x]$. Now $M[\zeta]^{\oplus} =$

$(\lambda x.\mathsf{f}(x))\mathsf{a}$, $M^{\oplus} = (\lambda x.\mathsf{f}(y))\mathsf{a}$. $\zeta^{\oplus} = [y \mapsto x]$. Note that $(M^{\oplus})^{(\zeta^{\oplus})} = \lambda z.\mathsf{f}(x)$, because substitutions are capture-avoiding, and thus $M[\zeta]^{\oplus} \neq_{\beta} (M^{\oplus})^{(\zeta^{\oplus})}$.

The solution is to add as arguments to the free variables of the preterms as many (bound) variables as necessary (or more) to make the distribution work. In the example above we would have $s = (\lambda x.\mathsf{f}(y(x)))\mathsf{a}$ and $\sigma = [y \mapsto \lambda x.x]$. Now, s and σ are, in a way that will be formalized in the next definition, similar to M and ζ, but now $M[\zeta]^{\oplus} =_{\beta} s^{\sigma}$.

Definition 4.9. *Let M be a* λx*-term and ζ a grafting. A tuple $\langle s, \sigma\rangle$ of preterm and substitution is a* λ-extension *of $\langle M, \zeta\rangle$ if there are graftings θ_1, θ_2 such that:*

– *$s = M[\theta_1]^{\oplus}$ and $\sigma = (\theta_2 \,;\, \zeta)^{\oplus}$;*
– *for each $X \in \mathrm{MV}(M)$, $\theta_1(X) = X(\overline{z})$ and $\theta_2(X) = \lambda\overline{z}.X$, where $\overline{z}$ is a list of variables containing at least the bound variables of M in scope that occur in $\zeta(X)$ (in arbitrary order).*

The notion of λ-extension is, again, naturally generalized to graftings and substitutions as the first component of the tuples.

Lemma 4.10. *Let $\langle s, \sigma\rangle$ be a λ-extension of $\langle M, \zeta\rangle$. Then:*

(i) $s^{\sigma} =_{\beta} M[\zeta]^{\oplus}$;
(ii) for each λx*-term N such that $M \twoheadrightarrow_{\mathrm{Bx}} N$, $s^{\sigma} =_{\beta} N[\zeta]^{\oplus}$.*

The lemma works, because the arguments of the free variables in the term and the leading abstractions in the substitution, take over the role of the explicit substitutions, as can be seen in the following example:

Example 4.11. Let $M = (\lambda x.(\lambda y.Z)\mathsf{b})\mathsf{a}$ be a λx-term, and $\zeta = [Z \mapsto \mathsf{f}(x, y)]$ a grafting. Now, according to Def. 4.9, $\langle s, \sigma\rangle$, where $s = (\lambda x.(\lambda y.z(x, y))\mathsf{b})\mathsf{a}$ and $\sigma = [z \mapsto \lambda xy.\mathsf{f}(x, y)]$ is a λ-extension of $\langle M, \zeta\rangle$, with, $\theta_1 = [Z \mapsto Z(x, y)]$ and $\theta_2 = [\lambda xy.Z]$. We check both cases of Lemma 4.10:

(i) $s^{\sigma} = (\lambda x.(\lambda y.(\lambda xy.\mathsf{f}(x, y))(x, y))\mathsf{b})\mathsf{a} =_{\beta} (\lambda x.(\lambda y.\mathsf{f}(x, y))\mathsf{b})\mathsf{a} = M[\zeta]^{\oplus}$.
(ii) Let $N = Z\{y\backslash\mathsf{b}\}\{x\backslash\mathsf{a}\}$. Then $M \twoheadrightarrow_{\mathrm{x}} N$. Let $t = z(\mathsf{a}, \mathsf{b})$. Now $t^{\sigma} =_{\beta} \mathsf{f}(\mathsf{a}, \mathsf{b}) = M[\zeta]^{\oplus}$. Since $s =_{\beta} t$, this means that $s^{\sigma} =_{\beta} M[\zeta]^{\oplus}$, as required. (Note that the $\cdot^{\oplus}$ operation also reduces to x-normal form.)

Translating patterns. Among the λ-extensions of a pair $\langle P, \zeta\rangle$ of linear, passive λx-term and grafting, there is, for each sequence of variables $\overline{x}$ exactly one λ-extension $\langle p, \sigma\rangle$ where p is a $\overline{x}$-pattern, viz. the one in which in p the free variables have *all* bound variables in scope as arguments. We denote by $\mathbf{P}^{+}_{\overline{x}}\langle P, \zeta\rangle$ the function which returns this specific λ-extension, and by $\mathbf{P}^{-}_{\overline{x}}$ the inverse of $\mathbf{P}^{+}_{\overline{x}}$. See [4] for a more detailed definition of these operations.

Example 4.12. Consider the linear, local λx-terms $P = \mathsf{f}(\lambda xy.\mathsf{g}(Z, x))$ and $Q = \mathsf{map}(\lambda x.Z, \mathsf{nil})$, and the grafting $\zeta = [Z \mapsto \mathsf{f}(x)]$. Then:

$$\mathbf{P}^{+}_{\emptyset}\langle P, \zeta\rangle = \langle \mathsf{f}(\lambda xy.\mathsf{g}(z(x, y), x)), [z \mapsto \lambda xy.\mathsf{f}(x)]\rangle$$
$$\mathbf{P}^{+}_{\emptyset}\langle Q, \zeta\rangle = \langle \mathsf{map}(\lambda x.Z(x), \mathsf{nil}), [z \mapsto \lambda x.\mathsf{f}(x)]\rangle$$

4.3 Proof of the Prefix Property

Proof (of Theorem 4.1). (Sketch). The interesting case is the case that the step occurs at the head, i.e.: $s = l^{\rho}$ and $p^{\sigma} = r^{\rho}$, for some rule $l \to r \in R$ and substitution ρ. We translate the terms to λx-terms: $\langle P, \zeta \rangle := \mathbf{P}^{-}_{\overline{x}} \langle p, \sigma \rangle$, $R := r^{\ominus}_{\emptyset}$, $L := l^{\ominus}_{\emptyset}$ and $\mu := \rho^{\ominus}_{\overline{x}}$. Because $p^{\sigma} =_{\beta} r^{\rho}$, and $P[\zeta]$ is a Bx-normal form by construction, it is the case that $R[\mu] \twoheadrightarrow_{\mathrm{Bx}} P[\zeta]$ (using Lemma 4.8).

Now, we use the λx-Prefix Property (Theorem 4.1) to find the ancestor P' of P in this reduction. This gives us, among other things, a graftings η, κ_1 such that $P'[\eta] = R[\mu]$, and $P' = P[\kappa]$. Equality here is first-order equality, and thus we apply first-order unification techniques (Lemma 4.4) to find an mgu $\langle \eta_0, \mu_0 \rangle$ for the unifier $\langle \eta, \mu \rangle$, and grafting κ_2 such that $\eta_0 ; \kappa_2 = \eta$ and $\mu_0 ; \kappa_2 = \mu$.

Now we translate everything back to (pre)terms, using the techniques discussed in the previous subsection: υ is the translation of κ_1 followed by η_0 (using λ-extensions to make the two composable), τ is the translation of κ_2 and Q is the translation of $L[\eta_0]$. This translation is cumbersome, but not hard in principle. The λ-extensions make sure that Bx-equality can be transformed to β-equality.

The last thing we have to prove is that $\mathrm{fam}(p) \geq \mathrm{fam}(q)$. This holds because $\mathit{Sym}(\eta_0) \subseteq \mathit{Sym}(p)$, because $\mathit{Sym}(\eta_0) \subseteq \mathit{Sym}(r)$, all labels in r are the same and p and r have at least one symbol in common because r is non-collapsing. $\square$

5 Finite Family Developments

In this section we apply the prefix property of the previous section to prove that all family developments of HRSs are finite. We restrict our attention to non-collapsing HRSs first. In the next section, we will describe a way to generalize the result to collapsing HRSs as well.

Families are formalized by labelling all function symbols with natural numbers, as defined in Def. 3.1. We prove that the resulting system is terminating if we restrict the labels to some finite bound. The proof is inspired by the proof by Van Oostrom [16]. The differences between this proof and the one by Van Oostrom are the following:

- We use a different method of labelling. Our labelling has the property that one step of the labelled HRS corresponds exactly to one step in the original. Also, our notion of labelling is an instance of the abstract notion of labelling put forth in [17,13].
- In Van Oostrom's paper, the proof of Lemma 15 is omitted. Here, we give a proof of that lemma (adapted for the different method of labelling) by reducing it to the Prefix Property.

Lemma 5.1. *Let $\mathcal{H}^{\omega}$ be the labelling of a non-collapsing HRS, s be a term, p a local pattern, $\ell \in \mathbb{N}$ a label and τ and σ substitutions such that for any $x \in \mathit{Dom}(\sigma)$, $\sigma(x)$ has a function symbol labelled with ℓ as head. If $s^{\sigma} \twoheadrightarrow_{\mathcal{H}^{\omega}} p^{\tau}$, then either:*

- $\mathrm{fam}(p) \geq \ell$*; or* *(int)*

– *$s \twoheadrightarrow_{\mathcal{H}} p^{\upsilon}$, for some υ such that $\upsilon\,;\sigma \twoheadrightarrow_{\mathcal{H}^{\omega}} \tau$.* *(ext)*

Proof. By induction on the length of the reduction $s^{\sigma} \twoheadrightarrow_{\mathcal{H}^{\omega}} p^{\tau}$. If the length is 0, the result follows easily. Otherwise, suppose $s^{\sigma} \twoheadrightarrow_{\mathcal{H}^{\omega}} s' \rightarrow_{\mathcal{H}^{\omega}} p^{\tau}$. By Theorem 4.1, there exist a local pattern q and substitution σ' such that $s' = q^{\sigma'}$, $\mathrm{fam}(p) \geq \mathrm{fam}(q)$ and either (trm) $q \twoheadrightarrow_{\mathcal{H}^{\omega}} p^{\upsilon'}$ and $\upsilon'\,;\sigma' = \tau$; or (sub) $p = q$ and $\sigma' \twoheadrightarrow_{\mathcal{H}^{\omega}} \tau$. Applying the induction hypothesis to $s^{\sigma} \twoheadrightarrow_{\mathcal{H}^{\omega}} q^{\sigma'}$ yields that one of the following cases must apply:

– *(int)* $\mathrm{fam}(q) \geq \ell$, but then $\mathrm{fam}(p) \geq \ell$ by transitivity of $\geq$.
– *(ext)* $s \twoheadrightarrow_{\mathcal{H}^{\omega}} q^{\upsilon}$ and $\upsilon\,;\sigma \twoheadrightarrow_{\mathcal{H}^{\omega}} \sigma'$, for some substitution υ. We distinguish the following cases:
  - *(trm)* $s \twoheadrightarrow_{\mathcal{H}^{\omega}} q^{\upsilon} \twoheadrightarrow_{\mathcal{H}^{\omega}} p^{\upsilon';\upsilon}$ and $\upsilon'\,;\upsilon\,;\sigma \twoheadrightarrow_{\mathcal{H}^{\omega}} \upsilon'\,;\sigma' = \tau$.
  - *(sub)* $s \twoheadrightarrow_{\mathcal{H}^{\omega}} q^{\upsilon} = p^{\upsilon}$ and $\upsilon\,;\sigma \twoheadrightarrow_{\mathcal{H}^{\omega}} \sigma' \twoheadrightarrow_{\mathcal{H}^{\omega}} \tau$. □

Theorem 5.2. *Let $\mathcal{H}^{\omega}$ be the labelling of a non-collapsing HRS, and let $\mathcal{R} : s_1 \rightarrow_{\mathcal{H}^{\omega}} s_2 \rightarrow_{\mathcal{H}^{\omega}} \cdots$ be a $\mathcal{H}^{\omega}$-reduction. $\mathcal{R}$ is finite, if and only if there is a $\ell_{\max} \in \mathbb{N}$ such that* $\mathrm{fam}(s_i) \leq \ell_{\max}$ *for all s_i.*

Proof. (Sketch) ($\Rightarrow$): Trivial. ($\Leftarrow$): We prove the theorem by showing that $\mathcal{H}^{\omega} = \langle \Sigma^{\omega}, R^{\omega} \rangle$ is terminating if we restrict it to rules $l \rightarrow r \in R^{\omega}$ where $\mathrm{fam}(r) \leq \ell_{\max}$. It suffices to show that r^{σ} terminates for all right-hand sides r and terminating substitutions σ. We do this by assuming, to the contrary, that a non-terminating term exists. Let $(s^{\ell})^{\sigma}$ be a *minimal* non-terminating term such that s is non-(subst-subst)-collapsing[1], and σ is a terminating substitution. By minimality, this reduction is of the form: $(s^{\ell})^{\sigma} \twoheadrightarrow_{\mathcal{H}^{\omega}} l^{\tau} \rightarrow_{\mathcal{H}^{\omega}} r^{\tau} \twoheadrightarrow_{\mathcal{H}^{\omega}} \cdots$, where $\lambda\overline{x}.l \rightarrow \lambda\overline{x}.r \in R^{\omega}$. We will show, by induction on $(\ell_{\max} - \ell)$, that $(s^{\ell})^{\sigma}$ is terminating, contradicting the assumption that it's not.

The interesting case is that $s = \lambda\overline{x}.y(s_1, \ldots, s_n)$, where $y \in \mathit{Dom}(\sigma)$. Let $t = \sigma(y)$, and $\sigma' = [x_i \mapsto s_i]$. Then $t^{\sigma'} \twoheadrightarrow_{\mathcal{H}} l^{\tau}$. By the fact that s is non-(subst-subst)-collapsing, the heads of the s_i are function symbols labelled with ℓ, and thus we can apply Lemma 5.1. Again, the interesting case is if $t \twoheadrightarrow_{\mathcal{H}^{\omega}} l^{\upsilon}$, and now termination of r^{τ} follows from the fact that σ is terminating by assumption. □

6 Dealing with Collapsing HRSs

In the previous sections we restricted our attention to non-collapsing HRSs. Both the Prefix Property and FFD do not hold for collapsing HRSs, as is witnessed by the following two counterexamples:

Example 6.1 (Prefix Property). Consider the collapsing HRS $\mathcal{M}u$:

$$\mathsf{mu}(\lambda x.z(x)) \rightarrow z(\mathsf{mu}(\lambda x.z(x)))$$

[1] We drop the (context-subst) condition of Def 2.5, because subterms of non (context-subst)-collapsing terms can be (constext-subst)-collapsing, meaning that an infinite reduction from a minimal counter example might not contain a head step.

and the following $\mathcal{M}u^\omega$-step:

$$\mathsf{mu}^3(\lambda x.\mathsf{f}^2(x)) \to_{\mathcal{M}u^\omega} \mathsf{f}^2(\mathsf{mu}^4(\lambda x.\mathsf{f}^2(x)))$$

It is easy to check that the prefix $p = \mathsf{f}^2(u)$ of the target of the step has no ancestor q that satisfies the requirements of the Prefix Property (Theorem 4.1).

Example 6.2 (FFD). Consider the collapsing HRS $\mathcal{L}am$:

$$\mathsf{app}(\mathsf{lam}(\lambda x.z(x), y)) \to z(y)$$

Then one $\mathcal{L}am^\omega$-step is the following:

$$\begin{aligned}&\mathsf{app}^1(\mathsf{lam}^1(\lambda x.\mathsf{app}^1(x,x)), \mathsf{lam}^1(\lambda x.\mathsf{app}^1(x,x)))\\ &\to_{\mathcal{L}am^\omega} \mathsf{app}^1(\mathsf{lam}^1(\lambda x.\mathsf{app}^1(x,x)), \mathsf{lam}^1(\lambda x.\mathsf{app}^1(x,x)))\end{aligned}$$

So we see that $\mathcal{L}am^\omega$ has a one-step cycle, and thus an infinite reduction with bounded labels.

The problem in both cases is that, because of applying a collapsing rule, a function symbol can be directly connected to a previously unconnected function symbol from the context or substitution, or to the root of the term, without the rule leaving any trace in between, in the form of a label. This can be remedied by including "empty" function symbols, named ϵ_α, for each base type α, in the right-hand sides of rules, and "saturating" the left-hand sides of rules with those empty function symbols. The same approach is taken for the first-order case in [13, Chap. 8]. We sketch the idea of this "ϵ-lifting", $\mathcal{H}^\epsilon$, by giving two examples; for a formal definition, see [4].

Example 6.3. The ϵ-lifting of $\mathcal{M}u$ (types of ϵ's omitted):

$$\mathsf{mu}(\lambda x.z(x)) \to \epsilon(z(\epsilon(\mathsf{mu}(\lambda x.\epsilon(z(\epsilon(x)))))))$$

Note that more ϵ's are added than strictly necessary; this is for ease of definition (see [4] for details). A $(\mathcal{M}u^\epsilon)^\omega$ step corresponding to the step of Ex. 6.1 is:

$$\mathsf{mu}^3(\lambda x.\mathsf{f}^2(x)) \to_{(\mathcal{M}u^\epsilon)^\omega} \epsilon^4(\mathsf{f}^2(\epsilon^4(\mathsf{mu}^4(\lambda x.\epsilon(\mathsf{f}^2(\epsilon(x)))))))$$

Take the corresponging prefix $p = \epsilon^4(\mathsf{f}^2(y))$. Now, the Prefix Property is satisfied with $q = \mathsf{mu}^3(\lambda x.\mathsf{f}^2(x))$, $\tau = \emptyset$ and $\upsilon = [z \mapsto \epsilon^4(\mathsf{mu}^4(\lambda x.\epsilon(\mathsf{f}^2(\epsilon(x)))))]$.

Example 6.4. The ϵ-lifting of $\mathcal{L}am$ consists of (among others) the following rules:

$$\begin{aligned}\mathsf{app}(\mathsf{lam}(\lambda x.z(x), y)) &\to \epsilon(z(\epsilon(y)))\\ \mathsf{app}(\epsilon(\mathsf{lam}(\lambda x.z(x))), y) &\to \epsilon(z(\epsilon(y)))\\ \mathsf{app}(\epsilon(\epsilon(\mathsf{lam}(\lambda x.z(x)))), y) &\to \epsilon(z(\epsilon(y)))\end{aligned}$$

Then a $(\mathcal{L}am^\epsilon)^\omega$-step corresponding to the step of Ex. 6.2 is the following:

$$\begin{aligned}&\mathsf{app}^1(\mathsf{lam}^1(\lambda x.\mathsf{app}^1(x,x)), \mathsf{lam}^1(\lambda x.\mathsf{app}^1(x,x)))\\ &\to_{(\mathcal{L}am^\epsilon)^\omega} \epsilon^2(\mathsf{app}^1(\epsilon^2(\mathsf{lam}^1(\lambda x.\mathsf{app}^1(x,x))), \epsilon^2(\mathsf{lam}^1(\lambda x.\mathsf{app}^1(x,x)))))\end{aligned}$$

Now, all redex patterns have a maximum label of 2, instead of 1.

Theorem 6.5 (FFD). *Let $(\mathcal{H}^\epsilon)^\omega$ be the $\epsilon\omega$-labelling of an HRS, and let $\mathcal{R}$: $s_1 \to_{(\mathcal{H}^\epsilon)^\omega} s_2 \to_{(\mathcal{H}^\epsilon)^\omega} \cdots$ be a $(\mathcal{H}^\epsilon)^\omega$-reduction. $\mathcal{R}$ is finite, if and only if there is a $\ell_{\max} \in \mathbb{N}$ such that* $\mathrm{fam}(s_i) \leq \ell_{\max}$ *for all s_i.*

7 Applications and Further Research

The Prefix Property and Finite Family Developments are useful tools for proving various properties of HRSs. For example, an alternative proof of termination of the simply typed λ-calculus (encoded as an HRS) uses FFD. Also, in a work in progress by the author, FFD is used to prove the termination of a higher-order standardization procedure. This result can be used to formalize the notion of equivalence of reductions, in a similar way as is done in [13].

For future research, it might be interesting to further investigate the relation between FFD and the Dependency Pair method [1], both in the higher-order and first-order case. Since FFD and the Dependency Pair method both essentially depend on the same principle, that an infinite reduction must have an unbounded creation depth, it the author's conjecture that FFD, or the Prefix Property, can be used to design a higher-order Dependency Pair method.

References

1. Thomas Arts and Jürgen Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236(1–2):133–178, 2000.
2. Inge Bethke, Jan Willem Klop, and Roel de Vrijer. Descendants and origins in term rewriting. *Information and Computation*, 159(1–2):59–124, 2000.
3. Roel Bloo. *Preservation of Termination for Explicit Substitution.* PhD thesis, Technische Universiteit Eindhoven, 1997.
4. H. J. Sander Bruggink. A proof of finite family developments for higher-order rewriting using a prefix property. Preprint, LGPS 245, Zeno Inst. of Phil., 2006.
5. Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Higher order unification via explicit substitutions. *Information and Computation*, 157(1–2):184–233, 2000.
6. J.M.E. Hyland. A syntactic characterization of the equality in some models of the λ-calculus. *Journal of the London Mathematical Society*, 12(2):361–370, 1976.
7. J. W. Klop. *Combinatory Reduction Systems.* PhD thesis, Utrecht Univ., 1980.
8. Jean-Jacques Lévy. *Réductions correctes et optimales dans le λ-calcus.* PhD thesis, Université Paris VII, 1978.
9. Luc Maranget. Optimal derivations in weak lambda-calculi and in orthogonal term rewriting systems. In *POPL*, 1991.
10. Richard Mayr and Tobias Nipkow. Higher-order rewrite systems and their confluence. *Theoretical Computer Science*, 192:3–29, 1998.
11. Dale Miller. A logic programming language with lambda abstraction, function variables and simple unification. *Journal of Logic and Computation*, 1(4), 1991.
12. Tobias Nipkow. Higher-order critical pairs. In *LICS*, 1991.
13. Terese. *Term Rewriting Systems.* Number 55 in CTTCS. CUP, 2003.
14. D.T. van Daalen. *The language theory of Automath.* PhD thesis, Technische Universiteit Eindhoven, 1980.
15. Vincent van Oostrom. *Confluence for Abstract and Higher-Order Rewriting.* PhD thesis, Vrije Universiteit Amsterdam, 1994.
16. Vincent van Oostrom. Finite family developments. In *RTA*, 1997.
17. Vincent van Oostrom and Roel de Vrijer. Four equivalent equivalences of reductions. *ENTCS*, 70(6), 2002.
18. C. P. Wadsworth. The relation between computational and denotational properties for Scott's D_∞-models of the λ-calculus. *SIAM Journal on Computing*, 5, 1976.