Dynamic Higher-Order Dependency Pairs with Argument Filterings

Cynthia Kop and Femke van Raamsdonk

Vrije Universiteit, Department of Computer Science,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
kop,femke@few.vu.nl

Abstract. We study the termination method using dynamic dependency pairs for higher order rewrite systems (HRSs), but restrict attention to the subclass of *weak* HRSs. For this class we define the notions of dependency pair and chain, and demonstrate how the subterm property can be weakened to allow a definition of argument filterings.

Key words: higher-order rewriting, termination, dependency pairs, subterm property, argument filtering

1 Introduction

An important technique to (automatically) prove termination of first-order term rewriting is the dependency pair approach by Arts and Giesl [3]. This method transforms a term rewriting system into a set of ordering constraints, which can be simplified using for instance argument filterings. Various optimisations of the dependency pair approach have been studied, see for example [6].

This paper contributes to the study of dependency pairs for higher-order rewriting. It is not easy to adapt the approach to this setting. A natural extension is given in [17], but it relies on the *subterm property* as an ordering constraint. Due to this property it is impossible to define optimisations like argument filterings or usable rules. Moreover, unlike the first order case, the method is not complete: a terminating system may have an infinite dependency chain. Other definitions of higher order dependency pairs crucially rely on some restriction, as discussed in Section 2. They either consider systems without β-reduction or impose limitations on the rewrite rules. No existing approach is complete.

The present work continues on this second line of research. We consider higher-order rewriting systems (HRSs) [15] and restrict attention to the class of *weak* HRSs. In such systems the left-hand sides of rewrite rules are linear and only contain abstractions in a very simple form; right-hand sides are not restricted. Weak HRSs hence form an intermediate class between systems without bound variables and pattern HRSs, which yields a natural class that contains most common examples. Our definitions of dependency pair and dependency chain conservatively extend the original ones for left-linear first-order rewriting, and use a weakened version of the subterm property. We can define argument filterings in a natural way, and obtain a termination method that is complete on weak HRS. The method has been implemented in the termination tool WANDA.

2 Related Work

Several variants of higher order dependency pairs exist. We discuss three categories: dependency pairs for applicative rewriting, the dynamic approach in HRSs, and the static approach in HRSs. Our contribution is in the second part.

Dependency Pairs for Applicative Rewriting In applicative term rewriting systems, terms are built from variables, constants and a binary application operator. Functional variables may be present, as in $x \cdot a$, but no abstractions, as in $\lambda x. x$. An important difference between the various approaches for dependency pairs is in the treatment of leading variables in right-hand sides of rewrite rules. In [7], an uncurrying transformation from untyped applicative systems to functional systems that preserves and reflects termination is used; after transforming, the first-order method can be applied. In [1,2], simply typed applicative systems (STTRSs) are considered; leading variables are eliminated by instantiating them with the relevant patterns. In [9,11,12], considering another version of simply typed applicative rewriting (STRSs), leading variables are not eliminated, but the method is restricted to 'plain function passing' systems where, intuitively, leading variables are harmless.

The issue of leading variables is also relevant for HRSs. However, the results for applicative systems are not useful in this setting, because termination may be lost by adding λ-abstraction (and hence anonymous functions). For instance, the (weak) HRS $\mathsf{app}(\mathsf{abs}(\lambda x. Z(x)), Y) \to Z(Y)$ for untyped λ-calculus is not terminating, whereas the corresponding applicative system $\mathsf{app} \cdot (\mathsf{abs} \cdot Z) \cdot Y \to Z \cdot Y$ does terminate (since the size of a term decreases with every rewrite step).

Dynamic Dependency Pairs A first, natural definition of dependency pairs on HRSs is given in [17]. Here, termination is not equivalent to the absence of infinite dependency chains. Also, a term is required to be greater than its subterms (the *subterm property*), which makes it impossible to define argument filterings.

Static Dependency Pairs The approach using static dependency pairs in [11] is extended to the setting of HRSs in [10]. Unlike the dynamic approach, the static approach does not admit dependency pairs of the form $f^{\#}(\boldsymbol{l}) \leadsto x(\boldsymbol{r})$ with x a variable. As a consequence, fewer ordering constraints are generated, and in addition no subterm property is needed; hence it is possible to define argument filterings and usable rules [18]. However, the approach is limited to plain function passing systems, and is moreover not complete; it will often fail on rules with abstractions in the right-hand side. For example, an HRS with a rule $\mathsf{I}(\mathsf{s}(n)) \to \mathsf{twice}(\lambda x. \mathsf{I}(x), n)$ generates a dependency pair $\mathsf{I}^{\#}(\mathsf{s}(n)) \leadsto \mathsf{I}^{\#}(x)$, which is impossible to handle.

The present work considers dynamic dependency pairs for HRSs and hence is most related to [17]. We restrict attention to *weak* HRSs, for which we can weaken the subterm property, characterize termination (Theorems 1,2), and introduce argument filterings. The restrictions *weak* and *plain function passing* are incomparable in the sense that neither implies the other.

3 Preliminaries

We recall some standard definitions and notations concerning Nipkow's higher-order rewriting systems (HRSs) [15]. Here the objects that are rewritten are simply typed λ-terms in long $\beta\eta$-normal form, and rewriting is modulo β. We assume some familiarity with HRSs; see for instance [15,13,19]. The set of natural numbers is denoted by $\mathbb{N}$ and the notation $[\ldots]$ is used for (possibly infinite) lists.

Types and Terms. The *simple types* (or just *types*) are built from base types, written as ι, κ, and the binary type constructor $\Rightarrow$ which is right-associative. Types are written as A, B, C. Every type has the form $A_1 \Rightarrow \ldots \Rightarrow A_n \Rightarrow \iota$ for some $n \geq 0$, types $A_1, \ldots, A_n$ and base type ι. A type $A \Rightarrow B$ is called *functional.*

We assume a set $\mathcal{V}$ containing infinitely many *variables* of each type. Variables are written as $x, y, z, X, Y, Z, \ldots$ or sometimes labelled with their type as in x^A. A *signature* is a non-empty set $\mathcal{F}$, assumed to be disjoint with $\mathcal{V}$, consisting of *function symbols*, each equipped with a unique type. Function symbols are written as f, g, h, or using more suggestive notation. If a symbol $a \in \mathcal{F} \cup \mathcal{V}$ has type A we write $a : A$. The set of *pre-terms* over $\mathcal{F}$ and $\mathcal{V}$ consists of all s for which we can infer $s : A$ for some type A, using the following clauses:

(sym) $a(s_1, \ldots, s_n) : \iota$ for $a : A_1 \Rightarrow \ldots \Rightarrow A_n \Rightarrow \iota$ in $\mathcal{V} \cup \mathcal{F}$
and $s_1 : A_1, \ldots, s_n : A_n$ with $n \geq 0$,

(abs) $\lambda x.\, s : A \Rightarrow B$ for $x : A$ and $s : B$,

(app) $t(s_1, \ldots, s_n) : \iota$ if $t : A_1 \Rightarrow \ldots \Rightarrow A_n \Rightarrow \iota$
and $s_1 : A_1, \ldots, s_n : A_n$ with $n \geq 1$.

The λ binds occurrences of variables as in the λ-calculus, and (pre-)term equality is modulo α-conversion; bound variables are renamed if necessary. Pre-terms are simply typed λ-terms in η-long form. Let $FV(s)$ be the set of free variables of a pre-term s. We write a instead of $a()$ and denote the unique η-long form of a symbol a as $a\!\uparrow$ (if $a : A_1 \Rightarrow \ldots \Rightarrow A_n \Rightarrow \iota$, then $a\!\uparrow = \lambda x_1 \ldots x_n.\, a(x_1\!\uparrow, \ldots, x_n\!\uparrow)$).

A *pre-substitution* $[\boldsymbol{x} := \boldsymbol{s}]$, with $\boldsymbol{x}$ and $\boldsymbol{s}$ non-empty finite vectors of equal length, is the homomorphic extension of the type-preserving mapping $\boldsymbol{x} \mapsto \boldsymbol{s}$ from variables to pre-terms. The set consisting of the variables $\boldsymbol{x}$ is called the *domain* of the substitution, notation $\mathrm{dom}(\cdot)$. Pre-substitutions are denoted by $\gamma, \delta, \ldots$, and the result of applying the pre-substitution γ to the pre-term s is denoted by $s\gamma$. Pre-substitutions do not capture free variables.

The *β-reduction relation*, denoted by $\rightarrow_\beta$, is induced by the following β-reduction rule: $(\lambda \boldsymbol{x}.\, s)\,(\boldsymbol{t}) \rightarrow_\beta s[\boldsymbol{x} := \boldsymbol{t}]$, where the vectors $\boldsymbol{x}$ and $\boldsymbol{t}$ are non-empty, finite, and of equal length. Note that long η-normal forms are preserved under β-reduction, so the set of pre-terms is closed under β-reduction.

Terms are equivalence classes of pre-terms modulo β. Usually we take the (unique) β-normal form as the representative of such a class. The β-normal form of a pre-term s is denoted by $s\!\downarrow$. The set of terms over $\mathcal{F}$ and $\mathcal{V}$ is denoted by $\mathcal{T}(\mathcal{F}, \mathcal{V})$ or, considering $\mathcal{V}$ fixed, just by $\mathcal{T}(\mathcal{F})$. Note that a term has the form $\lambda \boldsymbol{x}.\, a(\boldsymbol{s})$ with $a \in \mathcal{V} \cup \mathcal{F}$ called the *head* of the term, $\boldsymbol{x}$ zero or more variables, and $\boldsymbol{s}$ zero or more terms. Equivalently, it is a pre-term built without (app). A *substitution* is a pre-substitution mapping all variables in its domain to terms.

Rules and Rewriting. A term l is a *pattern* if every free variable x in l occurs in the form $x(y_1\uparrow, \dots, y_n\uparrow)$ with the y_i different variables which are bound in l. Patterns are defined by Miller [14], who proves that unification (and hence matching) modulo β is decidable for patterns. A *rewrite rule* (or just *rule*) over $\mathcal{T}(\mathcal{F})$ is a pair of terms $l \to r$ in $\mathcal{T}(\mathcal{F})$ such that l and r have the same base type, all free variables of r also occur in l, and l is a pattern of the form $f(l_1, \dots, l_n)$ for some $n \geq 0$ and function symbol f. A higher-order rewriting system (HRS) is usually specified by a set of rules. The *reduction relation* is generated by:

(head) $l\gamma{\downarrow} \to r\gamma{\downarrow}$ for a rewrite rule $l \to r$ and a substitution γ,
(sym) if $s \to t$ then $a(\dots, s, \dots) \to a(\dots, t, \dots)$ for $a \in \mathcal{V} \cup \mathcal{F}$,
(abs) if $s \to t$ then $\lambda x.\, s \to \lambda x.\, t$.

A rewrite rule $l \to r$ is *left-linear* if every free variable occurs at most once in l; an HRS is left-linear if all its rewrite rules are left-linear. If a function symbol is the head of a left-hand side then it is said to be a *defined symbol*, otherwise it is a *constructor*. The sets of constructors and defined symbols are denoted $\mathcal{C}$ and $\mathcal{D}$ respectively. We call $s \to t$ a *rewrite* or *reduction step*, and a sequence of rewrite steps is called a *reduction*.

Example 1. Throughout this paper, we will consider as an example the HRS eval. Its signature $\mathcal{F}_{\mathsf{eval}}$ consists of the following symbols:

$$\begin{array}{lll} \mathsf{o} : \mathsf{M} & \mathsf{dom} : \mathsf{M} \Rightarrow \mathsf{M} \Rightarrow \mathsf{M} \Rightarrow \mathsf{M} & \mathsf{eval} : \mathsf{M} \Rightarrow \mathsf{M} \Rightarrow \mathsf{M} \\ \mathsf{s} : \mathsf{M} \Rightarrow \mathsf{M} & \mathsf{fun} : (\mathsf{M} \Rightarrow \mathsf{M}) \Rightarrow \mathsf{M} \Rightarrow \mathsf{M} \Rightarrow \mathsf{M} & \end{array}$$

And it has the following rewrite rules:

$$\begin{array}{rr} \mathsf{eval}(\mathsf{fun}(\lambda x.\, F(x), X, Y), Z) \to F(\mathsf{dom}(X, Y, Z)) & \mathsf{dom}(X, Y, \mathsf{o}) \to X \\ \mathsf{dom}(\mathsf{s}(X), \mathsf{s}(Y), \mathsf{s}(Z)) \to \mathsf{s}(\mathsf{dom}(X, Y, Z)) & \mathsf{dom}(\mathsf{o}, \mathsf{o}, Z) \to \mathsf{o} \\ \mathsf{dom}(\mathsf{o}, \mathsf{s}(Y), \mathsf{s}(Z)) \to \mathsf{s}(\mathsf{dom}(\mathsf{o}, Y, Z)) & \end{array}$$

An example of a rewrite sequence in `eval`:

$$\begin{array}{l} \mathsf{eval}(\mathsf{fun}(\lambda x.\, \mathsf{fun}(\lambda y.\, y, \mathsf{o}, \mathsf{s}(\mathsf{s}(x))), \mathsf{s}(\mathsf{o}), \mathsf{s}(\mathsf{s}(\mathsf{o}))), \mathsf{s}(\mathsf{o})) \\ \to (\lambda x.\, \mathsf{fun}(\lambda y.\, y, \mathsf{o}, \mathsf{s}(\mathsf{s}(x))))\ (\mathsf{dom}(\mathsf{s}(\mathsf{o}), \mathsf{s}(\mathsf{s}(\mathsf{o})), \mathsf{s}(\mathsf{o}))) \\ = \ \mathsf{fun}(\lambda y.\, y, \mathsf{o}, \mathsf{s}(\mathsf{s}(\ \mathsf{dom}(\mathsf{s}(\mathsf{o}), \mathsf{s}(\mathsf{s}(\mathsf{o})), \mathsf{s}(\mathsf{o}))\ ))) \\ \to \mathsf{fun}(\lambda y.\, y, \mathsf{o}, \mathsf{s}(\mathsf{s}(\ \mathsf{s}(\mathsf{dom}(\mathsf{o}, \mathsf{s}(\mathsf{o}), \mathsf{o}))\ ))) \\ \to \mathsf{fun}(\lambda y.\, y, \mathsf{o}, \mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{o})))) \end{array}$$

The fun symbol represents a function over the natural numbers with an interval it is defined on; eval calculates its value in a point, provided that the point is in the domain of the function. All symbols have the same output type M, to enable functions mapping to any kind of output.

4 Weak HRSs

The notion of a *weak* reduction in the λ-calculus, first defined by Howard in 1968 and revisited in [5], simplifies the standard λ-calculus, while preserving most pleasant properties and expressivity. In the weak λ-calculus, a subterm below an abstraction cannot be reduced.

If we restrict attention to weak reductions, the subterm property from [17] can be weakened enough to define argument filterings (by not filtering inside an abstraction, see Section 6). However, since the goal of this paper is to define a dependency pair method for HRSs with the usual reduction relation, we instead impose a restriction on the rewrite rules, which guarantees that we only need weak reductions. Intuitively, no matching is done on an abstraction.

Definition 1. A *weak pattern* is a linear term either of the form $x\uparrow$ with x a free variable, or the form $f(s_1, \dots, s_n)$ with $f \in \mathcal{F}$ and all s_i weak patterns. A rule $l \to r$ is weak if l is a weak pattern. An HRS is weak if all of its rules are.

Obviously, a weak HRS is left-linear. Note that every weak pattern is a pattern, but not every (linear) pattern is weak. Examples of linear, yet non-weak patterns are $f(\lambda x.\, \mathsf{o})$, $f(\lambda x.\, \mathsf{s}(x))$ and $f(\lambda x.\, Z)$. All left-hand sides of the HRS eval are weak. Weak rewrite rules are fully extended.

With this definition we can limit attention to a strict form of weak reduction, where a reduction step does not take place below an abstraction.

Definition 2. A step $s \to t$ is *weak*, notation $s \to_{\mathsf{weak}} t$, if it is generated using only the clauses (head) and (sym). Otherwise, it is *strong*, notation $\to_{\mathsf{strong}}$.

The following lemma expresses that a strong step does not create a weak redex. Thus, in an infinite reduction either strong steps do not play a role, or eventually every step is inside an abstraction.

Lemma 1. *Let $s \to_{\mathsf{strong}} t \to_{\mathsf{weak}} u$ be a reduction in a weak HRS. Then there exists a term w such that $s \to_{\mathsf{weak}} w \to^* u$.*

Proof. There can be no overlap in the redexes of an weak and a strong step. If the weak redex occurs above the strong one, the latter can be postponed (although this may duplicate or remove the redex). If the redexes occur in a different part of the term, swap the steps. A strong step cannot occur above a weak one. □

To illustrate the *weak* requirement, consider the following examples of non-weak HRSs. In the left-linear HRS $\{f(\lambda x.\, Z) \to a, g(X) \to a\}$ we cannot swap the steps $f(\lambda x.\, g(x)) \to f(\lambda x.\, a) \to a$, because the first creates the second. In the left-linear and fully extended HRS $\{a \to b, f(\lambda x.\, b) \to c\}$, we cannot swap the steps $f(\lambda x.\, a) \to f(\lambda x.\, b) \to c$, for the same reason. In the HRS $\{f(X, X) \to b, a \to b\}$ we cannot swap the steps $f(\lambda x.\, a, \lambda x.\, b) \to f(\lambda x.\, b, \lambda x.\, b) \to b$.

As discussed in Section 2, weak HRSs differ from frameworks for applicative term rewriting because binders may be used in the term formation. The class of weak HRSs contains for instance extensions of λ-calculus with β-reduction.

Note that we can often simulate a non-weak HRS by a weak one, without affecting termination. For example, the rule $f(\lambda x.\, g(x), Y) \to r$ might be replaced by two weak rules $f(\lambda x.\, Z(x), Y) \to \mathsf{test}(Z(c), f'(\lambda x.\, Z(x), Y))$ and $\mathsf{test}(g(c), f'(\lambda x.\, Z(x), Y)) \to r$. However, it seems hard to simulate a rule with left-hand side $f(\lambda x.\, g(Z))$ without potentially affecting termination.

In the remainder of this paper we will work with the class of weak HRSs.

5 Dependency Pairs

The definition of a dependency pair uses the notion of candidate terms, intuitively those base type subterms which might cause non-termination. Their root symbol is either a defined symbol or a variable of functional type. To avoid bound variables becoming free when we take the body of a λ-abstraction, we use fresh constants $c_A : A$ for each type A. The set of all such constants is denoted *Const*.

Definition 3. The set of *candidate terms* of a term s, notation $Cand(s)$, is defined inductively as follows:

1. $Cand(x(t_1,\dots,t_n)) = \begin{cases} \emptyset & \text{if } n = 0 \\ \{x(t_1,\dots,t_n)\} & \text{if } n > 0 \end{cases}$
2. $Cand(f(t_1,\dots,t_n)) = \begin{cases} \{f(t_1,\dots,t_n)\} \cup Cand(t_1) \cup Cand(t_n) & \text{if } f \in \mathcal{D} \\ Cand(t_1) \cup \ldots \cup Cand(t_n) & \text{if } f \in \mathcal{C} \end{cases}$
3. $Cand(\lambda x^A.\, t) = Cand(t[x := c_A])$.

Example 2. As an example we consider two right-hand sides of the HRS eval:

$$\begin{aligned} Cand(F(\mathsf{dom}(X,Y,Z))) &= \{F(\mathsf{dom}(X,Y,Z))\} \\ Cand(\mathsf{s}(\mathsf{dom}(X,Y,Z))) &= \{\mathsf{dom}(X,Y,Z)\} \end{aligned}$$

Besides candidate terms, the definition of dependency pair uses marked function symbols. Let $\mathcal{F}^{\#} = \mathcal{F} \cup \{f^{\#} : \sigma \mid f : \sigma \text{ in } \mathcal{D}\}$, so the set of function symbols extended with for every defined symbol a marked symbol of the same type. Let $\mathcal{F}_c^{\#}$ be the union of $\mathcal{F}^{\#}$ and *Const*. The marked version of a term s, notation $s^{\#}$, is defined as $f^{\#}(s_1,\dots,s_n)$ if $s = f(s_1,\dots,s_n)$ with $f \in \mathcal{D}$, and as s (not introducing marks) otherwise.

Definition 4. The set of *dependency pairs of a rewrite rule* $l \to r$, notation $\mathsf{DP}(l \to r)$, is defined as $\{l^{\#} \rightsquigarrow p^{\#} \mid p \in Cand(r)\}$. The set of *dependency pairs of an HRS* $\mathcal{R}$, notation $\mathsf{DP}(\mathcal{R})$, is defined as $\bigcup_{l \to r \in \mathcal{R}} \mathsf{DP}(l \to r)$.

Note that for a dependency pair $l \rightsquigarrow p$ both l and p have base type, but those base types might not be the same. Term orderings such as for instance CPO [4] can usually compare terms of different base types.

Example 3. The set of dependency pairs of the HRS eval consists of:

$$\begin{aligned} \mathsf{eval}^{\#}(\mathsf{fun}(\lambda x.\, F(x), X, Y), Z) &\rightsquigarrow F(\mathsf{dom}(X,Y,Z)) \\ \mathsf{dom}^{\#}(\mathsf{s}(X),\mathsf{s}(Y),\mathsf{s}(Z)) &\rightsquigarrow \mathsf{dom}^{\#}(X,Y,Z) \\ \mathsf{dom}^{\#}(\mathsf{o},\mathsf{s}(Y),\mathsf{s}(Z)) &\rightsquigarrow \mathsf{dom}^{\#}(\mathsf{o},Y,Z) \end{aligned}$$

Termination is characterised by means of *dependency chains* in Theorem 1. These are sequences of dependency pairs with certain properties. First we need a definition of subterm which does not cause bound variables to become free.

Definition 5. The *subterm relation*, notation $\trianglerighteq$, is generated by the clauses:

1. $s \trianglerighteq s$,
2. $a(s_1, \dots, s_n) \trianglerighteq t$ if $s_i \trianglerighteq t$ for some i $\quad (a \in \mathcal{F} \cup \mathcal{V})$
3. $\lambda x^A.\, s \trianglerighteq t$ if $s[x := c_A] \trianglerighteq t$.

The strict part of $\trianglerighteq$ is denoted by $\triangleright$.

The subterms of $f(\lambda x.\, X(x))$ are $f(\lambda x.\, X(x))$, $\lambda x.\, X(x)$, $X(c)$, and c. Contrary to [17, Definition 1] we only have a single constant c_A for every type A, whereas they use a constant c_x for every variable x. Thus, here $\lambda x^A y^A.\, f(x, y)$ has $f(c_A, c_A)$ as a subterm, while the corresponding subterm in [17] is $f(c_x, c_y)$. We do not need different constants for all variables because we consider only left-linear rules.

Definition 6. A *dependency chain* for a HRS $\mathcal{R}$ is a sequence $[(l_i, p_i, t_i, \gamma_i) \mid i \in S]$ with either $S = \mathbb{N}$ or $S = \{1, \dots, n\}$ for some $n \in \mathbb{N}$, such that for all $i \in S$:

1. $l_i \rightsquigarrow p_i \in \mathsf{DP}(\mathcal{R})$, γ_i a substitution, t_i a term;
2. if $p_i = f^\#(r_1, \dots, r_n)$ then $t_i = p_i\gamma_i$;
3. if $p_i = x(r_1, \dots, r_n)$ then there is q such that $t_i = q^\#$ and $p_i\gamma_i \trianglerighteq q$ but not $\gamma_i(y) \trianglerighteq q$ for any variable y;
4. $t_i \rightarrow^*_{\mathsf{weak}} l_{i+1}\gamma_{i+1}$.

The reduction in the last clause does not contain head-steps as the head of l_{i+1} is some $f^\#$. The second clause is like in the first-order case, the third clause is typical for higher-order dynamic dependency pairs. Omitting the subscripts, we might for instance have $p = X(Y)$ and $\gamma = [X := \lambda x.\, h(x(a)), Y := \lambda x.\, f(x)]$; then $p\gamma = h(f(a))$, where each of $h^\#(f(a))$, $f^\#(a)$, but not $a^\#$, could be chosen for t_i. As an example of an infinite dependency chain, the dependency pair $\mathsf{app}^\#(\mathsf{abs}(\lambda x.\, Z(x)), Y) \rightsquigarrow Z(Y)$ of the HRS for untyped λ-calculus yields an infinite dependency chain with $t_i = \mathsf{app}^\#(\mathsf{abs}(\lambda x.\, \mathsf{app}(x, x)), \mathsf{abs}(\lambda x.\, \mathsf{app}(x, x)))$ and $\gamma_i = [Z := \lambda u.\, \mathsf{app}(u, u), Y := \mathsf{abs}(\lambda u.\, \mathsf{app}(u, u))]$ for all i.

Theorem 1. *A weak HRS $\mathcal{R}$ is terminating if and only if it does not have an infinite dependency chain.*

Proof. (Sketch) Assume that $\mathcal{R}$ is not terminating and consider a minimal term that admits an infinite reduction. Such a term has the form $f(s_1, \dots, s_n)$ with $f \in \mathcal{D}$. Because it is minimal non-terminating its infinite reduction contains a head-step at some point, and using Lemma 1 we can reach one with $\rightarrow_{\mathsf{weak}}$ steps. The head-step results in a non-terminating term s again; taking a minimal non-terminating subterm of s we can continue this process ad infinitum. This analysis naturally suggests the components of an infinite dependency chain.

For the other direction, we derive from an infinite dependency chain an infinite $\rightarrow \cup \triangleright$ reduction (erasing the marks); this implies non-termination of $\rightarrow$. $\square$

Remark 1. Note that in clause 1 of the definition of candidate terms we do not take the direct subterms of a term $x(s_1, \dots, s_n)$ as candidate terms. Had we done that, the approach would not be complete; there would be terminating HRSs with an infinite dependency chain. Consider for example the (terminating) HRS with rewrite rules $\{\mathsf{a} \to \mathsf{f}(\lambda x.\, \mathsf{o}), \mathsf{f}(\lambda x.\, Z(x)) \to Z(\mathsf{a})\}$. Its dependency pairs are $\mathsf{a}^\# \rightsquigarrow \mathsf{f}^\#(\lambda x.\, \mathsf{o})$ and $\mathsf{f}^\#(\lambda x.\, Z(x)) \rightsquigarrow Z(\mathsf{a})$ which indeed do not admit an infinite dependency chain. However, if a would also be a candidate term of $Z(\mathsf{a})$, then there is an additional dependency pair $\mathsf{f}^\#(\lambda x.\, Z(x)) \rightsquigarrow \mathsf{a}^\#$, which leads to the infinite dependency chain $\mathsf{a}^\# \rightsquigarrow \mathsf{f}^\#(\lambda x.\, \mathsf{o}) \to^* \mathsf{f}^\#(\lambda x.\, \mathsf{o}) \rightsquigarrow \mathsf{a}^\# \dots$.

Definition 7. A *reduction pair* consists of a well-founded ordering $>$ and a quasi-ordering (transitive and reflexive) $\geq$ on base type terms in $\mathcal{T}(\mathcal{F}^\#)$ where:

1. $> \cdot \geq$ is contained in $>$ (the pair is *compatible*),
2. if $s_i \geq s_i'$ then $a(s_1, \dots, s_i, \dots, s_n) \geq a(s_1, \dots, s_i', \dots, s_n)$ for all $a \in \mathcal{F}_c^\# \cup \mathcal{V}$ ($\geq$ is *base-monotonic*),
3. for all terms s and t with s a weak pattern, for all substitutions γ and $R \in \{>, \geq\}$: if sRt then also $s\gamma R t\gamma$ (the pair is *weakly stable*).

Note that a reduction pair is not required to be fully monotonic or stable. As we will see, the former is essential: the ordering generated by an argument filtering will only be base-monotonic. The requirement on stability is lax because substitution might cause β-reduction; even the subterm relation $\rhd$ and $\to_\mathcal{R}$ itself are not fully stable.

Theorem 2. *A weak HRS $\mathcal{R}$ on a signature $\mathcal{F}$ is terminating if and only if there exists a reduction pair $(>, \geq)$ such that:*

1. *$l > p$ for every dependency pair $l \rightsquigarrow p \in \mathsf{DP}(\mathcal{R})$,*
2. *$l \geq r$ for every rewrite rule $l \to r \in \mathcal{R}$,*
3. *$l\gamma > t^\#$ for every term t, substitution γ and dependency pair $l \rightsquigarrow p \in \mathsf{DP}(\mathcal{R})$ such that $p\gamma \unrhd t$ and $p = x(p_1, \dots, p_n)$ and not $\gamma(y) \unrhd t$ for any variable y.*

The third requirement of this theorem is guaranteed by the first if the subterm relation $\unrhd$ is contained in $\geq$ (that is, if $\geq$ satisfies the *subterm property*).

Proof. Given a dependency chain we can see that each $l_i\gamma_i > l_{i+1}\gamma_{i+1}$; thus well-foundedness of $>$ proves termination of $\mathcal{R}$ by Theorem 1. For the other direction, let $(>, \geq)$ be $(\to_\mathcal{R}^+ \cdot \unrhd, \to_\mathcal{R}^*)$. Then $(>, \geq)$ is a reduction pair satisfying the requirements, and well-founded when the system is terminating. □

Example 4. To prove termination of the HRS eval from Example 1 using the method given by Theorem 2, we need a reduction pair $(>, \geq)$ that satisfies requirement Th.2(3) and in addition the following constraints:

$$\begin{aligned}
\mathsf{eval}^\#(\mathsf{fun}(\lambda x.\, F(x), X, Y), Z) &> F(\mathsf{dom}(X, Y, Z))\\
\mathsf{dom}^\#(\mathsf{s}(X), \mathsf{s}(Y), \mathsf{s}(Z)) &> \mathsf{dom}^\#(X, Y, Z)\\
\mathsf{dom}^\#(\mathsf{o}, \mathsf{s}(Y), \mathsf{s}(Z)) &> \mathsf{dom}^\#(\mathsf{o}, Y, Z)\\
\mathsf{eval}(\mathsf{fun}(\lambda x.\, F(x), X, Y), Z) &\geq F(\mathsf{dom}(X, Y, Z))\\
\mathsf{dom}(\mathsf{s}(X), \mathsf{s}(Y), \mathsf{s}(Z)) &\geq \mathsf{s}(\mathsf{dom}(X, Y, Z))\\
\mathsf{dom}(\mathsf{o}, \mathsf{s}(Y), \mathsf{s}(Z)) &\geq \mathsf{s}(\mathsf{dom}(\mathsf{o}, Y, Z))\\
\mathsf{dom}(X, Y, \mathsf{o}) &\geq X\\
\mathsf{dom}(\mathsf{o}, \mathsf{o}, Z) &\geq \mathsf{o}
\end{aligned}$$

A priori it seems difficult to find a reduction pair satisfying these constraints, because if $\geq$ satisfies the subterm property (a very common property), we have:

$$\mathsf{eval}^{\#}(\mathsf{fun}(\lambda x.\, F(x), X, Y), Z) > F(\mathsf{dom}(X, Y, Z)) \geq F(Z)$$

Taking $\omega := \mathsf{fun}(\lambda x.\, \mathsf{eval}^{\#}(x, x), c_{\mathsf{nat}}, c_{\mathsf{nat}})$ we then have $\mathsf{eval}^{\#}(\omega, \omega) > \mathsf{eval}^{\#}(\omega, \omega)$. Fortunately, we don't *need* the subterm property to satisfy Th.2(3); as we shall see, we can avoid this problem with an argument filtering.

6 Argument Filterings

6.1 Beta-oblivious Reduction Pairs

We aim to simplify the constraints from the dependency pair method by using an argument filtering. However, to enable this we need a further weakening of property Th.2(3), which we can obtain by adding some dependency pairs we previously omitted. Consider the following modification of Definition 3.

Definition 8. The set of *β-oblivious candidate terms* of a term s, notation $Cand^{\beta o}(s)$, is defined inductively as follows:

1. $Cand^{\beta o}(x(t_1, \ldots, t_n)) = \begin{cases} \emptyset & \text{if } n = 0 \\ \{x(t_1, \ldots, t_n)\} \cup Cand^{\beta o}(t_1) \cup Cand^{\beta o}(t_n) & \text{if } n > 0 \end{cases}$
2. $Cand^{\beta o}(f(t_1, \ldots, t_n)) = \begin{cases} \{f(t_1, \ldots, t_n)\} \cup Cand^{\beta o}(t_1) \cup Cand^{\beta o}(t_n) & \text{if } f \in \mathcal{D} \\ Cand^{\beta o}(t_1) \cup \ldots \cup Cand^{\beta o}(t_n) & \text{if } f \in \mathcal{C} \end{cases}$
3. $Cand^{\beta o}(\lambda x^A.\, t) = Cand^{\beta o}(t[x := c_A])$,

Example 5. We consider the two right-hand sides of eval as in Example 2:

$$Cand^{\beta o}(F(\mathsf{dom}(X, Y, Z))) = \{F(\mathsf{dom}(X, Y, Z)), \mathsf{dom}(X, Y, Z)\}$$
$$Cand^{\beta o}(\mathsf{s}(\mathsf{dom}(X, Y, Z))) = \{\mathsf{dom}(X, Y, Z)\}$$

The following notion of β-oblivious dependency pair follows Definition 4 closely.

Definition 9. The set of *β-oblivious dependency pairs of a rewrite rule* $l \to r$, notation $\mathsf{DP}^{\beta o}(l \to r)$, is $\{l^{\#} \rightsquigarrow p^{\#} \mid p \in Cand^{\beta o}(r)\}$. The set of *$\beta$-oblivious dependency pairs of an HRS* $\mathcal{R}$, notation $\mathsf{DP}^{\beta o}(\mathcal{R})$, is $\bigcup_{l \to r \in \mathcal{R}} \mathsf{DP}^{\beta o}(l \to r)$.

The difference with Definitions 3 and 4 is that we take subterms of a term $x(t_1, \ldots, t_n)$ as candidates, even though such subterms might disappear by β-reduction. Consequently, the *if and only if* in Theorems 1 and 2 does not hold, as described in Remark 1.

Example 6. The set of β-oblivious dependency pairs of the HRS eval consists of its dependency pairs given in Example 3, with in addition:
$\mathsf{eval}^{\#}(\mathsf{fun}(\lambda x.\, F(x), X, Y), Z) \rightsquigarrow \mathsf{dom}^{\#}(X, Y, Z)$

Now we can prove a β-oblivious version of Theorem 2.

Theorem 3. *A weak HRS $\mathcal{R}$ is terminating if there is a reduction pair $(>, \geq)$ such that:*

1. *$l > p$ for every dependency pair $l \rightsquigarrow p \in \mathsf{DP}^{\beta o}(\mathcal{R})$,*
2. *$l \geq r$ for every rewrite rule $l \to r \in \mathcal{R}$,*
3. *$l\gamma > t^{\#}$ for every term t, substitution γ and $l \rightsquigarrow p \in \mathsf{DP}^{\beta o}(\mathcal{R})$ such that $p\gamma \trianglerighteq t$ and $p = x(p_1, \dots, p_n)$ for some variable x but neither $\gamma(y) \trianglerighteq t$ for any $y \in \mathrm{dom}(\gamma)$ nor $p_i\gamma \trianglerighteq t$ for any p_i.*

Compared to the statement of Theorem 2, there are more ordering constraints to be satisfied, but we have gained the assumption $p_i\gamma \trianglerighteq t$ in requirement 3.

Proof. (Sketch) We use the 'if' part of Theorem 2. The first two requirements follow easily by noting that every dependency pair is also a β-oblivious dependency pair. The third requirement requires a little analysis. $\square$

6.2 Argument Filterings

Argument filterings are used to simplify the constraints from the dependency pair method. An argument filtering either eliminates some direct arguments s_i of a term $f(s_1, \dots, s_n)$, or replaces the term by one of the s_i. An argument filtering is defined as a partial mapping on a signature, which then induces a mapping on terms, as given in the following definitions.

Definition 10. An *argument filtering* for a signature $\mathcal{F}$ is a partial mapping $\pi : \mathcal{F}^{\#} \to \mathsf{lists}(\mathsf{nat}) \cup \mathsf{nat}$ such that for a function symbol $f : A$ in the domain of π with $A_1 \Rightarrow \dots A_n \Rightarrow \iota$ we have: either $\pi(f) = [i_1, \dots, i_k]$ with $1 \leq i_1 < \dots < i_k \leq n$ or $\pi(f) = i$ with $i \in \{1, \dots, n\}$ and $A_i = \iota$.

In the first case (and only then), we introduce f_π of type $A_{i_1} \Rightarrow \dots \Rightarrow A_{i_k} \Rightarrow \iota$, the *filtered version* of the function symbol f. The set $\mathcal{F}_\pi$ is defined as $\mathcal{F}_c^{\#}$ extended with f_π for all f in the domain of π.

The induced mapping to obtain filtered terms is straightforward except in that arguments cannot be eliminated if one of them contains a bound variable. Therefore we need to keep track of the bound variables on a path from the root.

Definition 11. Given an argument filtering π for a signature $\mathcal{F}$ and a term t over $\mathcal{F}$, the *filtered version* $\pi(t)$ of t is defined as $\pi_\emptyset(t)$, where $\pi_S(t)$ for $S \subset \mathcal{V}$ is defined as follows:

1. $\pi_S(x(t_1, \dots, t_n)) = x(\pi_S(t_1), \dots, \pi_S(t_n))$ for x a (free or bound) variable,
2. $\pi_S(f(t_1, \dots, t_n)) = \begin{cases} f_\pi(\pi_S(t_{i_1}), \dots, \pi_S(t_{i_k})) & \text{if } \pi(f) = [i_1, \dots, i_k] \\ & \text{and } FV(f(t_1, \dots, t_n)) \cap S = \emptyset \\ \pi_\emptyset(t_i) & \text{if } \pi(f) = i \\ & \text{and } FV(f(t_1, \dots, t_n)) \cap S = \emptyset \\ f(\pi_S(t_1), \dots, \pi_S(t_n)) & \text{otherwise} \end{cases}$
3. $\pi_S(\lambda y.\, t) = \lambda y.\, \pi_{S \cup \{y\}}(t)$.

Note that the third possibility in clause 2 is applied if either the argument filtering is not defined for f, or a variable from S appears in some t_i.

Example 7. We use the signature $\mathcal{F}_{\mathsf{eval}}$ from Example 1. Let π be an argument filtering with $\pi(\mathsf{dom}) = [1, 2]$ and $\pi(\mathsf{s}) = 1$. Then we have:

$$\pi(\lambda x.\, \mathsf{dom}(\mathsf{s}(x), \mathsf{o}, \mathsf{dom}(\mathsf{s}(y), \mathsf{s}(z), u))) = \lambda x.\, \mathsf{dom}(\mathsf{s}(x), \mathsf{o}, \mathsf{dom}_\pi(y, z))$$

Note that in the filtered term both the normal function symbol $\mathsf{dom} : \mathsf{nat} \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}$ and its filtered version $\mathsf{dom}_\pi : \mathsf{nat} \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}$ occur.

We present the main result of this section: a weak HRS is terminating if a suitable argument and reduction pair exist. Such a reduction pair should orient both the filtered β-oblivious dependency pairs and the filtered rules, and should additionally satisfy that an unfiltered symbol f is "heavier" than f_π and $f^\#_\pi$.

Theorem 4 (Argument Filtering). *A weak HRS $\mathcal{R}$ over a signature $\mathcal{F}$ is terminating if there is an argument filtering π and a reduction pair $(>, \geq)$ with:*

- $\pi(l) > \pi(p)$ *for every* $l \rightsquigarrow p \in \mathsf{DP}^{\beta o}(\mathcal{R})$,
- $\pi(l) \geq \pi(r)$ *for every* $l \to r \in \mathcal{R}$,
- $>$ *contains* $\rhd$ *(but doesn't have to be monotonic),*
- $f(X_1\!\uparrow, \dots, X_n\!\uparrow) \geq f_\pi(X_{i_1}\!\uparrow, \dots, X_{i_k}\!\uparrow)$ *for every* $f \in \mathcal{F}$ *if* f_π *is defined,*
- $f(X_1\!\uparrow, \dots, X_n\!\uparrow) \geq f^\#_\pi(X_{i_1}\!\uparrow, \dots, X_{i_k}\!\uparrow)$ *for every* $f \in \mathcal{D}$ *if* $f^\#_\pi$ *is defined.*

The proof of Theorem 4 depends on a number of properties of the π function. Let $\sqsupseteq_{AF}$ be the filtering relation: $\sqsupseteq_{AF}$ is the smallest transitive, reflexive and monotonic relation on $\mathcal{T}(\mathcal{F}_\pi)$ such that $f(s_1, \dots, s_n) \sqsupseteq_{AF} f_\pi(s_{i_1}, \dots, s_{i_k})$ if $\pi(f) = [i_1, \dots, i_k]$ and $f(s_1, \dots, s_n) \sqsupseteq_{AF} s_i$ if $\pi(f) = i$. Note that, given a reduction pair as described in the theorem, $\sqsupseteq_{AF}$ is included in $\geq$.

Lemma 2 (properties of $\sqsupseteq_{AF}$ and π_X).

1. $\pi_S(x\!\uparrow) = x\!\uparrow$ *for all variables* x.
2. *If* $Y \cap FV(s) = \emptyset$, *then* $\pi_{X \cup Y}(s) = \pi_X(s)$
3. *If* l *is a weak pattern, so is* $\pi(l)$.
4. $s \sqsupseteq_{AF} \pi_X(s)$ *(for* $s \in \mathcal{T}(\mathcal{F}^\#_c)$, $X \subseteq \mathcal{V}$*).*
5. *If* $a(s_1, \dots, s_n) \sqsupseteq_{AF} q$, *then either* $q = a(q_1, \dots, q_n)$ *(with each* $s_j \sqsupseteq_{AF} q_j$*) or* $q = a_\pi(q_{i_1}, \dots, q_{i_k})$ *(if* $a \in \mathcal{F}^\#, \pi(a) = [i_1, \dots, i_k]$ *and each* $s_{i_j} \sqsupseteq_{AF} q_{i_j}$*) or* $s_i \sqsupseteq_{AF} q$ *(if* $\pi(a) = i$*). If* $\lambda x.\, s \sqsupseteq_{AF} q$ *then* $q = \lambda x.\, q'$ *with* $s \sqsupseteq_{AF} q'$.
6. $s\gamma \sqsupseteq_{AF} t\delta$ *if* $s \sqsupseteq_{AF} t$, $\mathrm{dom}(\gamma) = \mathrm{dom}(\delta)$ *and each* $\gamma(x) \sqsupseteq_{AF} \delta(x)$
7. $\pi_{X \cup Y}(s)\pi_X(\gamma) \sqsupseteq_{AF} \pi_X(s\gamma)$ *(X, Y disjunct, dom(*γ*)* $= Y$, $s \in \mathcal{T}(\mathcal{F}^\#_c)$*)*
8. $\pi_X(s)\pi(\gamma) \sqsupseteq_{AF} \pi_X(s\gamma)$ *(*$X \subseteq \mathcal{V}$, $s \in \mathcal{T}(\mathcal{F}^\#_c)$ *and* γ *a substitution such that* $\mathrm{dom}(\gamma)$ *and all* $FV(\gamma(x))$ *are disjunct from* X*)*
9. $\pi(l)\pi(\gamma) = \pi(l\gamma)$ *for weak patterns* l
10. *If* $s := f(s_1, \dots, s_n) \sqsupseteq_{AF} q \sqsupseteq_{AF} \pi_X(s)$, *and* $FV(s) \cap X \neq \emptyset$, *then* q *has the form* $f(q_1, \dots, q_n)$ *with* $s_i \sqsupseteq_{AF} q_i \sqsupseteq_{AF} \pi_X(s_i)$ *for all* i.

Proof. Most of the details of this proof are sheer drudgery, so we present a sketch of the methods. Full proofs are available in the appendix.

(1), (2), (3) and (4) are proved by a straightforward induction on the type of x, definition of π_X, form of l and definition of π_X respectively (the base case of (3) is given by (1)). For (5), the relation given by those rules is monotonic, reflexive, transitive and contained in $\sqsupseteq_{AF}$, and therefore equal to it. (6) holds by induction on the pre-term $s\gamma$, ordered by the union of $\rightarrow_\beta$ and the standard subterm relation. We need $\rightarrow_\beta$ in the case $s = x(s_1, \dots, s_n)$ with $x \in \mathrm{dom}(\gamma)$ and $n > 0$. (7) is proved by a similar induction on the pre-term $s\gamma$ (with $\rightarrow_\beta \cup \rhd$). The interesting cases are when $FV(s) \cap Y = \emptyset$ (and therefore $\pi_{X\cup Y}(s)\pi_X(\gamma) = \pi_{X\cup Y}(s) = \pi_X(s)$ by (2)), and when $s = x(s_1, \dots, s_n)$ with $x \in Y$ and $n > 0$ (here we use the $\rightarrow_\beta$ part of the induction hypothesis). (8) holds by induction on the size of s, using (7) if s is headed by a variable in $\mathrm{dom}(\gamma)$. The requirement for the range of γ to be disjunct from γ is to have $FV(s\gamma) \cap X = \emptyset$ when $FV(s) \cap X = \emptyset$. (9) is a straightforward induction on l again.

For (10) we first prove (10a): *if $s \rhd t$ and s has base type, $FV(s) \cap X \neq \emptyset$, then not $t \sqsupseteq_{AF} \pi_X(s)$.* Let $s = a(s_1, \dots, s_n)$ and, towards a contradiction, choose t minimal such that $s \rhd t \sqsupseteq_{AF} \pi_X(s) = a(\pi_X(s_1), \dots, \pi_X(s_n))$. Then by (5) and minimality, $t = a(t_1, \dots, t_n)$ with each $t_j \sqsupseteq_{AF} \pi_X(s_j)$ and since some $s_i \unrhd t \rhd t_i$ and $FV(s_i) \cap X \supseteq FV(t) \cap X \supseteq FV(\pi_X(s)) \cap X \neq \emptyset$ we can always reduce the example to a smaller one, contradiction. As to (10) itself, by (10a) never $s_i \sqsupseteq_{AF} \pi_X(s)$, so neither $s \sqsupseteq_{AF} q$ nor $q \sqsupseteq_{AF} \pi_X(s)$ uses the third possibility of (5). Since $\pi_X(s) = f(\pi_X(s_1), \dots, \pi_X(s_n))$ with f unfiltered, the only possibility for q is $f(q_1, \dots, q_n)$ with each $q_i \sqsupseteq_{AF} \pi_X(s_i)$. □

Lemma 2 provides most of the necessities to prove Theorem 4. The only remaining problem is Thm.3(3). Lemma 3 will solve this problem.

Lemma 3. *Let $s\gamma \unrhd t$ with t of base type and neither $s \unrhd t$ nor any $\gamma(x) \unrhd t$. Then $\pi_X(s)\pi(\gamma) \unrhd \cdot \sqsupseteq_{AF} h\pi(t)$, where $X = \mathrm{dom}(\gamma)$ and $h\pi(a(u_1, \dots, u_n)) = a(\pi(u_1), \dots, \pi(u_n))$.*

Proof. The fundamental idea is that the head symbol of subterm t in $s\gamma$ has to occur somewhere in either s or γ, say $s = C[a(\boldsymbol{u})]$ or some $\gamma(x) = C[a(\boldsymbol{u})]$. By the assumption $a(\boldsymbol{u}) \neq t$, so some u_i contains a variable in X or a bound variable; consequently, π_X will not filter this term away, nor will it affect the head symbol. For the formal proof, we introduce a definition that allows for a somewhat stronger induction hypothesis: *u preserves v in w if either $w = v$ and $u \sqsupseteq_{AF} h\pi(v)$, or $w \rhd v$ and 1) if $w = \lambda x. w'$, then $u = \lambda x. u'$ and $u'[x := c]$ preserves v in $w'[x := c]$, 2) if $w = a(w_1, \dots, w_n)$ for $a \in \mathcal{F}_c^\# \cup \mathcal{V}$, then $u = a(u_1, \dots, u_n)$ and u_i preserves v in w_i for each $i \leq n$.* It is easy to see that a statement "u preserves v in w" implies that if $w \rhd v$, then also $u \rhd \cdot \sqsupseteq_{AF} h\pi(v)$. The lemma follows if we can prove: *given terms s, s', base type term t, and substitutions γ, γ' on the same domain X, then $s'\gamma'$ preserves t in $s\gamma$ if:*

1. s, t and each $\gamma(x)$ are terms in $\mathcal{T}(\mathcal{F}_c^\#)$; s' and each $\gamma'(x)$ are in $\mathcal{T}(\mathcal{F}_\pi)$;
2. $s \sqsupseteq_{AF} s' \sqsupseteq_{AF} \pi_X(s)$ and all $\gamma(x) \sqsupseteq_{AF} \gamma'(x) \sqsupseteq_{AF} \pi(\gamma(x))$;

3. s' preserves t in s and each $\gamma'(x)$ preserves t in $\gamma(x)$;

This follows with induction on the pre-term $s\gamma$, ordered by the union of $\rightarrow_\beta$ and the proper subterm relation. The proof uses Lemma 2(5,6,7,10). $\square$

Combining Lemmas 2 and 3, we are sufficiently prepared to prove Theorem 4.

Proof (proof of Theorem 4). Let $s \succ t$ iff $\pi(s) > \pi(t)$ and $s \succeq t$ iff $\pi(s) \geq \pi(t)$. It is easy to see that $\succ$ and $\succeq$ are compatible and that $\succeq$ is base-monotonic (even though it is not fully monotonic). If $l \succ r$ then also $\pi(l\gamma) = \pi(l)\pi(\gamma) > \pi(r)\pi(\gamma) \geq \pi(r\gamma)$ by Lemma 2(9,3,8). Similarly $\succeq$ is weakly stable, so $(\succ, \succeq)$ is a reduction pair. The first requirements of Theorem 3 are clearly met, for the last we use Lemma 3 (noting that $\rhd$ is part of $>$ and $\sqsupseteq_{AF}$ part of $\geq$) and the last two requirements of Theorem 4. $\square$

We apply Theorem 4 to our running example.

Example 8. We consider the weak HRS eval from example 1. We use the argument filtering π with $\pi(\mathsf{dom}) = \pi(\mathsf{dom}^\#) = [1, 2]$. In order to prove termination of eval we need to satisfy the following constraints:

$$\begin{array}{lll}
- & \mathsf{eval}^\#(\mathsf{fun}(\lambda x.\,F(x), X, Y), Z) & > F(\mathsf{dom}_\pi(X, Y)) \\
 & \mathsf{eval}^\#(\mathsf{fun}(\lambda x.\,F(x), X, Y), Z) & > \mathsf{dom}^\#_\pi(X, Y) \\
 & \mathsf{dom}^\#_\pi(\mathsf{s}(X, \mathsf{s}(Y)) & > \mathsf{dom}^\#_\pi(X, Y) \\
 & \mathsf{dom}^\#_\pi(\mathsf{o}, \mathsf{s}(Y)) & > \mathsf{dom}^\#_\pi(\mathsf{o}, Y) \\
- & \mathsf{eval}(\mathsf{fun}(\lambda x.\,F(x), X, Y), Z) & \geq F(\mathsf{dom}_\pi(X, Y)) \\
 & \mathsf{dom}_\pi(\mathsf{s}(X), \mathsf{s}(Y)) & \geq \mathsf{s}(\mathsf{dom}_\pi(X, Y, Z)) \\
 & \mathsf{dom}_\pi(\mathsf{o}, \mathsf{s}(Y)) & \geq \mathsf{s}(\mathsf{dom}_\pi(\mathsf{o}, Y)) \\
 & \mathsf{dom}_\pi(X, Y, \mathsf{o}) & \geq X \\
 & \mathsf{dom}_\pi(\mathsf{o}, \mathsf{o}, Z) & \geq \mathsf{o} \\
- & \mathsf{dom}(X_1, X_2, X_3) & \geq \mathsf{dom}_\pi(X_1, X_2) \\
- & \mathsf{dom}(X_1, X_2, X_3) & \geq \mathsf{dom}^\#_\pi(X_1, X_2) \\
- & \rhd \text{ is contained in } > &
\end{array}$$

This is easy with a higher order path ordering as in [4], using a precedence $\mathsf{eval}^\# =_{\mathbb{T}} \mathsf{eval} >_{\mathbb{T}} \mathsf{dom}_\pi =_{\mathbb{T}} \mathsf{dom}^\#{}_\pi >_{\mathbb{T}} \mathsf{s}$.

7 Implementation

We have implemented these results in C++ in the tool WANDA [8], which has participated in the termination competition of 2010, the first year a higher-order category was present. Unfortunately, the termination problem database does not yet have many benchmarks for higher-order rewriting, and the random selection of the competition chose no examples where dependency pairs are needed. At present, the tool supports dependency pairs with argument filterings and a recursive path ordering. An external SAT-solver is employed for choosing the argument filtering and the RPO precedence.

8 Concluding Remarks

In this paper, we have given a modified definition of the dependency pair approach for HRSs and demonstrated how argument filterings can be defined, as long as these HRSs satisfy the *weakness* constraint. Other optimisations of the dependency pair approach, such as using a dependency graph, usable rules and an extension of the monotone algebra approach [16], are left for separate work. We also believe the notion of *argument filtering* can be generalised, to allow argument filterings where for instance a term $f(s, t)$ can be replaced by $s(t)$.

It is worth noting that, although we have used a very strict definition of weak reduction, our notion of argument filterings uses only that a term containing a bound variable is not reduced. Therefore, if we consider HRSs without limitations on the rules, but where the rewrite relation is restricted to weak reductions as in [5], we have little doubt that the notions and proofs in this paper can be applied without much additional effort.

Acknowledgments. We thank Vincent van Oostrom for discussions and suggestions, and the anonymous referees of an earlier version of this paper for their constructive remarks.

References

1. T. Aoto and Y. Yamada. Dependency pairs for simply typed term rewriting. In J. Giesl, editor, *Proceedings of RTA 2005*, volume 3467 of *LNCS*, pages 120–134, Nara, Japan, April 2005. Springer.
2. T. Aoto and Y. Yamada. Argument filterings and usable rules for simply typed dependency pairs. In S. Ghilardi and R. Sebastiani, editors, *Proceedings of FroCoS 2009*, volume 5749 of *LNCS*, pages 117–132, Trento, Italy, September 2009. Springer.
3. T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.
4. F. Blanqui, J.-P. Jouannaud, and A. Rubio. The computability path ordering: the end of a quest. In *Proceedings of CSL 2008*, volume 5213 of *LNCS*, pages 1–14, Bertinoro, Italy, September 2008. Springer.
5. N. Çağman and Roger Hindley. Combinatory weak reduction in lambda-calculus. *Theoretical Computer Science*, 198:239–247, 1998.
6. N. Hirokawa and A. Middeldorp. Automating the dependency pair method. *Information and Computation*, 205(4):474–511, 2007.
7. N. Hirokawa, A. Middeldorp, and H. Zankl. Uncurrying for termination. In *Proceedings of LPAR 2008*, volume 5330 of *LNAI*, pages 667–681, Doha, 2008. Springer.
8. C. Kop. Wanda. `http://www.few.vu.nl/~kop/code.html`.
9. K. Kusakari. On proving termination of term rewriting systems with higher-order variables. *IPSJ Transactions on Programming*, 42(SIG 7 PRO11):35–45, 2001.
10. K. Kusakari, Y. Isogai, M. Sakai, and F. Blanqui. Static dependency pair method based on strong computability for higher-order rewrite systems. *IEICE Transactions on Information and Systems*, 92(10):2007–2015, 2009.

11. K. Kusakari and M. Sakai. Enhancing dependency pair method using strong computability in simply-typed term rewriting. *Applicable Algebra in Engineering, Communication and Computing*, 18(5):407–431, 2007.
12. K. Kusakari and M. Sakai. Static dependency pair method for simply-typed term rewriting and related techniques. *IEICE Transactions*, 2(92-D):235–247, 2009.
13. R. Mayr and T. Nipkow. Higher-order rewrite systems and their confluence. *Theretical Computer Science*, 192:3–29, 1998.
14. D. Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. *Journal of Logic and Computation*, 1(4):497–536, 1991.
15. T. Nipkow. Higher-order critical pairs. In *Proceedings of LICS 1991*, pages 342–349, Amsterdam, The Netherlands, July 1991.
16. J.C. van de Pol. *Termination of Higher-order Rerwite Systems.* PhD thesis, University of Utrecht, 1996.
17. M. Sakai, Y. Watanabe, and T. Sakabe. An extension of the dependency pair method for proving termination of higher-order rewrite systems. *IEICE Transactions on Information and Systems*, E84-D(8):1025–1032, 2001.
18. S. Suzuki, K. Kusakari, and F. Blanqui. Argument filterings and usable rules in higher-order rewrite systems. `http://www-rocq.inria.fr/~blanqui/`.
19. Terese. *Term Rewriting Systems.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2003.

Appendix: complete proofs

.1 Proofs relating to Section 4

We split up Lemma 1 in two parts: Lemmas 7 and 8. Lemmas 4, 5 and 6 are needed in these proofs. We use the notation $\rightarrow_{head}$ for a reduction at the head, $\rightarrow_{in}$ for a reduction not at the head and $\rightarrow_{\mathsf{weak},in}$ for a weak reduction not at the head.

Lemma 4. *If $s \rightarrow_{\mathsf{strong}} t$ we can find a context C and terms $\lambda x.\, u, \lambda x.\, v$ such that $s = C[\lambda x.\, u], t = C[\lambda x.\, v], u \rightarrow_{\mathcal{R}} v$ and $\lambda x.\, u$ does not freely contain variables bound by C.*

Proof. Suppose $s \rightarrow_{\mathsf{strong}} t$; we must find context C, variable x and terms u, v such that $s = C[\lambda x.\, u]$ and $t = C[\lambda x.\, v]$. Noting that $\rightarrow_{\mathsf{strong}}$ is a subrelation of $\rightarrow_{\mathcal{R}}$, we can perform induction on its derivation. It can not be that $s \rightarrow_{\mathcal{R},\text{head}} t$, since clause (abs) must have been used in the derivation. If $s = a(s_1, \ldots, s_i, \ldots, s_n)$ with $a \in \mathcal{F} \cup \mathcal{V}$, $t = a(s_1, \ldots, s_i', \ldots, s_n)$ and $s_i \rightarrow_{\mathcal{R}} s_i'$, then an (abs) step must have occurred in the derivation of $s_i \rightarrow_{\mathcal{R}} s_i'$, so by induction hypothesis $s_i = D[\lambda x.\, u]$, $s_i' = D[\lambda x.\, v]$; take $C[] = a(s_1, \ldots, D[], \ldots, s_n)$. Finally, if $s = \lambda x.\, u$, then t must be $\lambda x.\, v$ with $u \rightarrow_{\mathcal{R}} v$, so choose for C the empty context. □

Lemma 5. *For a given context C and term q such that $FV(q) \subseteq FV(C[q])$, for a fresh variable x of the same type as q, $C[q] = C[x\!\uparrow][x := q]$.*

Proof. By induction on the size of C. The base case, $x\!\uparrow [x := q] = q$, is a well-known property of the $\beta\eta$-normal form. If $C[q] = a(\ldots, D[q], \ldots)$, the induction hypothesis gives that $C[q] = a(\ldots, D[q], \ldots) = a(\ldots, D[x\!\uparrow][x := q], \ldots) = a(\ldots, D[x\!\uparrow], \ldots)[x := q]$ (since x is fresh), $= C[x\!\uparrow][x := q]$. Finally, if $C[q] = \lambda y.\, D[q]$ the induction hypothesis on D provides $D[q] = D[x\!\uparrow][x := q]$; since y does not occur in q this gives $C[q] = \lambda y.\, D[q] = \lambda y.\, (D[x\!\uparrow][x := q]) = (\lambda y.\, D[x\!\uparrow])[x := q] = C[x\!\uparrow][x := q]$. □

Lemma 6. *Suppose l is a weak linear pattern and $l\gamma = C[\lambda x.\, u]$, with $FV(\lambda x.\, u) \subseteq FV(l\gamma)$. Let y be a fresh variable of the same type as $\lambda x.\, u$. Then we can find a substitution δ such that $l\delta = C[y\!\uparrow]$ and $\delta[y := \lambda x.\, u] = \gamma$.*

Proof. By induction on the form of l. Either $l = z\!\uparrow$ for some variable z, or $l = f(l_1, \ldots, l_n)$ with $f \in \mathcal{F}$. In the first case, $C[\lambda x.\, u] = l\gamma = \gamma(z)$. Write $\gamma = [a_1 := q_1, \ldots, a_n := q_n, z := C[\lambda x.\, u]]$ and let $\delta := [a_1 := q_1, \ldots, a_n := q_n, z := C[y\!\uparrow]]$. Then evidently $l\delta = C[y\!\uparrow]$, and $\delta[y := \lambda x.\, u] = [a_1 := q_1, \ldots, a_n := q_n, z := C[y\!\uparrow][y := \lambda x.\, u]]$ (because y is fresh), $= [a_1 := q_1, \ldots, a_n := q_n, z := C[\lambda x.\, u]]$ (by Lemma 5), $= \gamma$.

In the second case, each l_i has a distinct set of free variables because l is linear (this is part of the weakness constraint); let γ_i be the restriction of γ to $FV(l_i)$. We can write $C[] = f(l_1\gamma_1, \ldots, D[], \ldots, l_n\gamma_n)$ with $D[\lambda x.\, u] = l_i\gamma_i$. By the induction hypothesis on l_i (which is also a weak pattern) we find δ' such that $l_i\delta' = D[y\!\uparrow]$ and $\delta'[y := \lambda x.\, u] = \gamma_i$. Taking $\delta := \gamma_1 \cup \ldots \delta' \cup \ldots \gamma_n$ (all these have disjunct domains) we have $l\delta = C[y\!\uparrow]$ and $\delta[\lambda x.\, u] = \gamma$! □

Lemma 7. *Let $\mathcal{R}$ be a set of weak rules. If $s \rightarrow_{\mathcal{R},\mathsf{strong}} w \rightarrow_{\mathcal{R},head} t$, then there exists q such that $s \rightarrow_{\mathcal{R},head} q \rightarrow^*_{\mathcal{R}} t$.*

Proof. Using Lemma 4, write $s = C[\lambda x. u]$, $w = C[\lambda x. v]$. Let y be a fresh variable of the same type as $\lambda x. u$. Then $s = C[y\uparrow][y := \lambda x. u]$ and $t = C[y\uparrow][y := \lambda x. v]$ (Lemma 5; $\lambda x. u$ does not freely contain variables bound by C). As $w \rightarrow_{head} t$, write $w = l\gamma, t = r\gamma$ for a rule $l \rightarrow r$. Since l is a weak pattern it will also match on $C[y\uparrow]$: according to Lemma 6 there is a substitution δ with $C[y\uparrow] = l\delta$. Now observe that $s = C[y\uparrow][y := \lambda x. u] = l\delta[y := \lambda x. u] \rightarrow_{head} r\delta[y := \lambda x. u] \rightarrow^* r\delta[y := \lambda x. v] = C[y\uparrow][y := \lambda x. v] = t$; hence take $q := r\delta[y := \lambda x. u]$. □

Lemma 8. *Let $\mathcal{R}$ be a set of weak rules. If $s \rightarrow_{\mathcal{R},\mathsf{strong}} w \rightarrow_{\mathcal{R},\mathsf{weak},in} t$, then there exists q such that $s \rightarrow_{\mathcal{R},\mathsf{weak},in} q \rightarrow^*_{\mathcal{R},in} t$.*

Proof. By induction on the structure of s. First observe that w has base type (since $w \rightarrow_{\mathsf{weak}} t$), and shares type with s and t. Hence, $s = a(s_1, \dots, s_n)$ with $a \in \mathcal{F} \cup \mathcal{V}$. Since a strong step cannot occur at the head of a base type term, $w = a(s_1, \dots, s_i', \dots, s_n)$ and as the $w \rightarrow t$ step is internal $w = a(w_1, \dots, w_j, \dots, w_n)$ and $t = a(w_1, \dots, w_j', \dots, w_n)$. If $j \neq i$ the two steps are independent; $s \rightarrow_{\mathsf{weak},in} a(s_1, \dots, w_j', \dots, s_n) \rightarrow_{\mathsf{strong}} t$. If $j = i$ and $w_j \rightarrow_{head} w_j'$ we apply Lemma 7 to find $s_i \rightarrow_{head} v \rightarrow^* w_i'$ and therefore $s \rightarrow_{\mathsf{weak},in} a(s_1, \dots, v, \dots, s_n) = f(w_1, \dots, v, \dots, w_n) \rightarrow^*_{in} f(w_1, \dots, w_n)$. Finally, if $j = i$ and $w_j \rightarrow_{\mathsf{weak},in} w_j'$ we can apply the induction hypothesis for a similar result. □

Lemmas 7 and 8 together prove Lemma 1.

.2 Proofs relating to Section 5

To demonstrate Theorem 1 we need a number of properties of the subterm relation $\rhd$. These are straightforward and some of them have been proved elsewhere for a real subterm relation. We give both a short proof and a complete version with all the details.

Lemma 9 (properties of $\rhd$). *For all terms s, t and weak rules $\mathcal{R}$:*

1. *$s \rhd t$ iff $s = C[t']$ with $C \neq \Box$ and $t'[\boldsymbol{x} := \boldsymbol{c}] = t$ where $\{\boldsymbol{x}\} = FV(t') \setminus FV(s)$.*
2. *$s[\boldsymbol{x} := \boldsymbol{c}] \rightarrow_{\mathcal{R}} t$ iff there is some t' such that $t = t'[\boldsymbol{x} := \boldsymbol{c}]$ and $s \rightarrow_{\mathcal{R}} t'$.*
3. *If $s \unrhd \cdot \rightarrow_{\mathcal{R}} t$ then also $s \rightarrow_{\mathcal{R}} \cdot \unrhd t$ (where $\cdot$ denotes composition).*
4. *If $\rightarrow_{\mathcal{R}}$ is terminating then so is $\rightarrow_{\mathcal{R}} \cup \rhd$.*

Proof (short proof of lemma 9). (1) is proved by a simple induction, on the size of C for the $\Leftarrow$ implication and on the number of direct subterm steps ($f(\dots, s, \dots) \rhd s, \lambda x. s \rhd s[x := c]$) for $\Rightarrow$. For (2), perform induction on the definition of $\rightarrow_{\mathcal{R}}$; the induction step reduces easily to the induction hypothesis, and the base step follows from left-linearity of $\mathcal{R}$. (3) is a combination of (1) and (2) (if $s = C[r'] \rhd r'[\boldsymbol{x} := \boldsymbol{c}] \rightarrow_{\mathcal{R}} t$ then $s \rightarrow_{\mathcal{R}} C[t'] \rhd t$). (4) is a consequence of (3), since by shuffling $\rightarrow_{\mathcal{R}}$ steps forward every infinite $\rightarrow_{\mathcal{R}} \cup \rhd$ sequence would lead to either an infinite $\rightarrow_{\mathcal{R}}$ reduction or an infinite $\rhd$ chain (which is impossible, since $\rhd$ reduces to a smaller term). $\square$

Proof (long proof of lemma 9(1)). We have two directions to show.

For the $\Leftarrow$ direction, suppose $s = C[t']$, $FV(t') \setminus FV(s) = \{\boldsymbol{x}\}$ and $t'[\boldsymbol{x} := \boldsymbol{c}] = t$. We prove $s \unrhd t$ by induction on the size of C, and $s \rhd t$ if C is not the empty substitution follows immediately (since $C[t']$ has more symbols than t). Certainly, if C is the empty substitution, $s \unrhd t$ because $s = t$. Otherwise, write either $C[] = \lambda y. D[]$ or $C[] = a(\dots, D[], \dots)$ with $a \in \mathcal{F}_c \cup \mathcal{V}$. In the first case let $\{\boldsymbol{z}\} = FV(D[t']) = \{\boldsymbol{x}\} \setminus \{y\}$. We have $s \rhd t$ if $D[t'][y := c] \unrhd t = t'[y := c][\boldsymbol{z} := \boldsymbol{c}]$, which holds by the induction hypothesis. In the second case note that $FV(t') \setminus FV(D[t']) = FV(t') \setminus FV(C[t'])$, so $D[t'] \unrhd t'[\boldsymbol{y} := \boldsymbol{c}] = t$ by induction and therefore $C[t'] \rhd t$.

For the $\Rightarrow$ direction, let !! denote the direct subterm relation, so $a(\dots, q, \dots)!!q$ and $(\lambda x. q)!!q[x := c]$ ($\rhd$ is the transitive closure of !!). If $s \rhd t$ then $s = s_0!!s_1!! \dots !!s_n$. We perform induction on n. Certainly, if $n = 0$ then $s = t$ and we can complete with $t' = t$ and C the empty substitution. If $n > 0$, let $q = s_{n-1}$. We may use the induction hypothesis to find a term q', context C and variables $\{\boldsymbol{x}\} = FV(q') \setminus FV(s)$ such that $s = C[q']$ and $q'[\boldsymbol{x} := \boldsymbol{c}] = q$. Now, if $q = a(q_1, \dots, t, \dots, q_n)$, then it can only be that $q' = a'(q_1', \dots, t', \dots, q_n')[\boldsymbol{x} := \boldsymbol{c}]$; since $FV(t') \setminus FV(s) \subseteq FV(q') \setminus FV(s)$ this suffices (choose environment $D[] = C[a'(q_1', \dots, \square, \dots, q_n')]$). If $q = \lambda y. t$ we can write $q' = \lambda y. t'$ with $t'[\boldsymbol{x} := \boldsymbol{c}, y := c] = t$; then $FV(t') \setminus FV(s) \subseteq \{\boldsymbol{x}\} \cup \{y\}$ and we can choose context $D = \lambda y. \square$. $\square$

Proof (long proof of lemma 9(2)). Given $s[\boldsymbol{x} := \boldsymbol{c}] \rightarrow_{\mathcal{R}} t$ we must find t' such that $s \rightarrow_{\mathcal{R}} t'$ and $t'[\boldsymbol{x} := \boldsymbol{c}] = t$.

To start, we prove by induction on l: if $s[\boldsymbol{x} := \boldsymbol{c}] = l\gamma$ with l a weak pattern and $\mathrm{dom}(\gamma) = FV(l)$, there is δ such that $s = l\delta$ and $\gamma(z) = \delta[\boldsymbol{x} := \boldsymbol{c}](z)$ for all $z \in FV(l)$. First, if $l = y{\uparrow}$ for a variable y, take $\delta(y) := s$; evidently $\delta[\boldsymbol{x} := \boldsymbol{c}](y) = s[\boldsymbol{x} := \boldsymbol{c}] = \gamma(y)$. Second, if $l = f(l_1, \dots, l_n)$, then $s = f(s_1, \dots, s_n)$ with $l_i\gamma = s_i[\boldsymbol{x} := \boldsymbol{c}]$ for all i. By linearity of l we can split γ in substitutions $\gamma_1, \dots, \gamma_n$ on the disjunct domains $\mathrm{dom}(\gamma_1), \dots, \mathrm{dom}(\gamma_n)$. Use the induction hypothesis on each l_i; we find δ_i such that $s_i = l_i\delta_i$ and $\gamma_i(z) = \delta[\boldsymbol{x} := \boldsymbol{c}](z)$ for all $z \in FV(l)$. All δ_i can be assumed to have the same domain as γ_i, so taking $\delta := \delta_1 \cup \dots \cup \delta_n$ we have a well-defined substitution satisfying all requirements.

Now, given s, t such that $s[\boldsymbol{x} := \boldsymbol{c}] \rightarrow_{\mathcal{R}} t$. We find t' with induction on the definition of $\rightarrow_{\mathcal{R}}$. In the base case, $s[\boldsymbol{x} := \boldsymbol{c}] \rightarrow_{\mathcal{R},head} t$, there are $l \rightarrow r \in \mathcal{R}$ and substitution γ such that $s[\boldsymbol{x} := \boldsymbol{c}] = l\gamma$, $r\gamma = t$; we can assume $\mathrm{dom}(\gamma) = FV(l)$ and $FV(l)$ is disjunct from $\{\boldsymbol{x}\}$. As we saw above, there is δ such that $s = l\delta$ and $\delta[\boldsymbol{x} := \boldsymbol{c}] = \gamma$; take $t' = r\delta$ and we have $s \rightarrow_{head} t'$ and $t'[\boldsymbol{x} := \boldsymbol{c}] = r\delta[\boldsymbol{x} := \boldsymbol{c}] = r(\delta[\boldsymbol{x} := \boldsymbol{c}]) = t$, the last step because no x_i occurs in r (being free variables which don't occur in l). For the induction step, either $s[\boldsymbol{x} := \boldsymbol{c}] = \lambda y. u \rightarrow \lambda y. v = t$ with $y \notin \{\boldsymbol{x}\}$ or $s[\boldsymbol{x} := \boldsymbol{c}] = a(q_1, \dots, u, \dots, q_n) \rightarrow a(q_1, \dots, v, \dots, q_n)$ with $a \in \mathcal{F}_c \cup \mathcal{V}$. We can write $s = \lambda x. u'$ or $s = a'(q_1', \dots, u', \dots, q_n')$ (a'

might be one of the x_i) and use the induction hypothesis to find v' such that $u' \to v'$ and $v'[\boldsymbol{x} := \boldsymbol{c}] = v$. If $s = \lambda y.\, u'$ take $t' := \lambda y.\, v'$, otherwise choose $t' := a'(q'_1, \dots, v', \dots, q'_n)$. In either case, the requirements are met. □

Proof (long proof of lemma 9(3)). By (1) and (2) $s = C[r']$, $r' \to_{\mathcal{R}} t'$, $r = r'[\boldsymbol{x} := \boldsymbol{c}]$ and $t = t'[\boldsymbol{x} := \boldsymbol{c}]$ (with $\{\boldsymbol{x}\} = FV(r') \setminus FV(s) \supseteq FV(t') \setminus FV(s)$). Postponing the $\rhd$ step, $s = C[r'] \to_{\mathcal{R}} C[t'] \rhd t$. □

Proof (long proof of lemma 9(4)). Let t_1 be an arbitrary term that is non-terminating under $\to_{\mathcal{R}} \cup \rhd$. Given t_i, let s be a minimal term such that $t_i \rhd s$ and an infinite such sequence starts in s; such s always exists. Since s is minimal the first step in an infinite sequence starting in s would be $s \to_{\mathcal{R}} q$; then $t_i \unrhd \cdot \to_{\mathcal{R}} q$ so by (3) we can find t_{i+1} such that $t_i \to_{\mathcal{R}} t_{i+1} \unrhd q$. But then also an infinite $\to_{\mathcal{R}} \cup \rhd$ reduction starts from t_{i+1}, and we can proceed with this term. This infinite procedure generates a sequence $t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} \dots$. □

Having Lemma 9, and defining $|s|$ as the term s with any marks removed, we can prove Theorem 1 in detail:

Proof (proof of Theorem 1). For the $\Leftarrow$ direction, we assume $\to_{\mathcal{R}}$ admits an infinite reduction and create an infinite dependency chain. To that end, find a minimal non-terminating term t and let $t_0 := t^{\#}$. For all $i \in \mathbb{N}$, given a term $t_i = f(s_1, \dots, s_n)^{\#}$ such that $|t_i| = f(s_1, \dots, s_n)$ is minimal non-terminating, consider an infinite reduction starting in $|t_i|$. Since all s_j are terminating not all steps in this reduction can be strong (eventually a headmost step will have to be done); using Lemma 1 pull weak steps to the left until we have a reduction $|t_i| \to_{\mathsf{weak},in} u_1 \to_{\mathsf{weak},in} \dots \to_{\mathsf{weak},in} w \to_{head} w'$ (a head step will be reached eventually, again because $|t_i|$ is minimal). w' is still non-terminating and $w = l\gamma$, $w' = r\gamma$ for some rule $l \to r$ and substitution γ, which we can safely assume has domain $FV(l)$. Since $\to_{\mathsf{weak},in}$ steps do not affect the root symbol, l has the form $f(q_1, \dots, q_n)$, and thus f is a defined symbol. Define $l_{i+1} := l^{\#}$ and $\gamma_{i+1} := \gamma$ and let p be the smallest candidate term of r such that $p\gamma$ is non-terminating. Let $p_{i+1} := p^{\#}$ and $t_{i+1} := q^{\#}$ where q is the smallest non-terminating $\unrhd$-subterm of $p\gamma$ (so $|t_{i+1}| = q$ is also minimal non-terminating).

During this process, it is evident that $l_{i+1} \leadsto p_{i+1} \in \mathsf{DP}(\mathcal{R})$ and that $t_i \to_{\mathsf{weak}}^{*} l_{i+1}\gamma_{i+1}$ (since for $\to_{in}$ reductions it does not matter if the head symbol is changed from f to $f^{\#}$). If $p = f(p_1, \dots, p_m)$ with $f \in \mathcal{D}$ then $p\gamma = f(p_1\gamma, \dots, p_m\gamma)$; by minimality of p not $p_i\gamma \unrhd q$, so $p^{\#}\gamma = t$. If $p = x(p_1, \dots, p_m)$ there is no such guarantee, but evidently not $q \subseteq \gamma(y)$ for any y, as these are strict subterms of l. As such, the chain $[(l_i, p_i, \gamma_i, t_i) | i \in \mathbb{N}]$ created in this process is an infinite dependency chain.

For the $\Rightarrow$ direction, let an infinite dependency chain be given and define $s_i := |l_i|\gamma_i$ for all $i \in \mathbb{N}$. Then always $s_i \to_{\mathcal{R}}^{+} \cdot \unrhd \cdot \to_{\mathcal{R}}^{*} s_{i+1}$, which by Lemma 9(4) implies non-termination of $\to_{\mathcal{R}}$. □

Theorem 2 follows without much difficulty.

Proof (long proof of Theorem 2). Given a reduction pair $(>, \geq)$ satisfying the requirements in the theorem, and infinite dependency chain $[(l_i, p_i, \gamma_i, t_i) | i \in \mathbb{N}]$ we have for each i: $l_i\gamma_i > t_i$ either by requirement 1 and 7(3) (if $\mathrm{head}(p_i) \in \mathcal{F}_c^{\#}$), or by 3 (if $\mathrm{head}(p_i) \in \mathcal{V}$). We have $t_i \geq l_{i+1}\gamma_{i+1}$ by requirements 2, 7(3) and 7(2). By 7(1) then $l_i\gamma_i > l_{i+1}\gamma_{i+1}$ for all i, contradicting well-foundedness.

For the other direction, given a terminating rewrite relation $\rightarrow_{\mathcal{R}}$, let $s > t$ if $|s| \rightarrow_{\mathcal{R}}^{+} \cdot \trianglerighteq |t|$ and $s \geq t$ if $|s| \rightarrow_{\mathcal{R}}^{*} |t|$. Evidently $>$ is an ordering and $\geq$ a quasi-ordering; compatibility (7(1)) holds by Lemma 9(3). $\geq$ is fully monotonic, and as for weak stability, note that $\rightarrow_{\mathcal{R}}$ is weakly stable, as can be seen by induction over the derivation $s \rightarrow_{\mathcal{R}} t$ (if $s = l\gamma$ and $t = r\gamma$ for $l \rightarrow r \in \mathcal{R}$, then $s\delta = l\gamma\delta \rightarrow_{\mathcal{R}} r\gamma\delta = t\delta$; if $s = f(s_1, \ldots, s_i, \ldots, s_n) \rightarrow_{\mathcal{R}} f(s_1, \ldots, s_i', \ldots, s_n)$ because $s_i \rightarrow_{\mathcal{R}} s_i'$, then by induction $s_i\delta \rightarrow_{\mathcal{R}} s_i'\delta$ so $s\delta = f(s_1\delta, \ldots, s_i\delta, \ldots, s_n\delta) \rightarrow_{\mathcal{R}} f(s_1\delta, \ldots, s_i'\delta, \ldots, s_n\delta) = s\delta$; these are the only forms s might have, because $x\uparrow$ with x a variable does not reduce). We can also see that $\triangleright$ is weakly stable, with induction over the derivation of $s \trianglerighteq t$ (if $s = t$ then $s\gamma = t\gamma$ and if $s = f(s_1, \ldots, s_n) \triangleright t$ because $s_i \trianglerighteq t$, then by induction $s\gamma \triangleright t\gamma$ because $s_i\gamma \trianglerighteq t\gamma$). Combining this, both $>$ and $\geq$ are weakly stable, and thus $(>, \geq)$ is a reduction pair. It is also clear that $l > p$ for all dependency pairs $l \rightsquigarrow p$ and that $l \geq r$ for all rules $l \rightarrow r$. Requirement 3 is similarly satisfied. □

.3 Proofs relating to Section 6.1

Proof (long proof of Theorem 3). We show that the reduction pair as described in Theorem 3 satisfies the requirements of Theorem 2. Requirements 1 and 2 are passed immediately by the corresponding requirements of Theorem 3. For the last requirement let $l \rightsquigarrow x(p_1, \ldots, p_m) \in \mathsf{DP}(\mathcal{R})$ with $x \in \mathcal{V}$, γ a substitution with domain $FV(l)$ and $t \trianglelefteq p\gamma$ such that not $\gamma(y) \trianglerighteq t$ for any y. If not $p_i\gamma \trianglerighteq t$ for any of the p_i we are done by condition 3 of Theorem 3. Otherwise, consider the smallest q such that some $p_i \trianglerighteq q$ and $q\gamma \trianglerighteq t$. Since t has base type we can assume that q does, too, so write $q = a(q_1, \ldots, q_n)$ and note that no $q_i\gamma \trianglerighteq t$ by minimality of q. If $a \in \mathcal{V}$ and $n = 0$, then $\gamma(a) \trianglerighteq t$, contradiction. If $a \in \mathcal{V}$ and $n > 0$ then $l \rightsquigarrow q^{\#} \in \mathsf{DP}^{\beta o}(\mathcal{R})$ and by Thm.3(3) it follows that $l\gamma > t^{\#}$. If $a \in \mathcal{F}_c$ then $q\gamma = a(q_1\gamma, \ldots, q_n\gamma)$; since none of the $q_i\gamma \trianglerighteq t$ we must have $q\gamma = t$, and therefore $a \in \mathcal{D}$. So in this case too, $l \rightsquigarrow q^{\#} \in \mathsf{DP}^{\beta o}(\mathcal{R})$, and by Thm.3(1) and weak stability $l\gamma > q^{\#}\gamma = t^{\#}$. □

.4 Proofs relating to Section 6.2

Individual proofs of Lemma 2

Proof (long proof of Lemma 2(1)). By induction on the type of x. It is obvious for base type x, and if $x\uparrow = \lambda y_1 \ldots y_n . x(y_1\uparrow, \ldots, y_n\uparrow)$ then $\pi_S(x\uparrow) = \lambda y_1 \ldots y_n . x(\pi_{S \cup \{y_1, \ldots, y_n\}}(y_1\uparrow), \ldots, \pi_{S \cup \{y_1, \ldots, y_n\}}(y_n\uparrow))$, whether $x \in S$ or $x \notin S$. By the induction hypothesis on all y_i, this $= \lambda y_1 \ldots y_n . x(y_1\uparrow, \ldots, y_n\uparrow) = x\uparrow$. □

Proof (long proof of Lemma 2(2)). By induction on the definition of π_X. Note that always $FV(s) \cap X = FV(s) \cap (X \cup Y)$. Note also that if $FV(s) \cap Y = \emptyset$ and q is a direct subterm of s, also $FV(q) \cap Y = \emptyset$ Even when $s = \lambda y.\, q$, as y can be chosen fresh. Thus, $\pi_{X \cup Y}(s) = \lambda y.\, \pi_{(X \cup \{y\}) \cup Y}(q) =$ (IH) $\lambda y.\, \pi_{X \cup \{y\}}(q) = \pi_X(s)$. When $s = a(s_1, \dots, s_n)$ and either $a \notin \mathrm{dom}(\pi)$ or $X \cap FV(s) \neq \emptyset$, then $\pi_{X \cup Y}(s) = a(\pi_{X \cup Y}(s_1), \dots, \pi_{X \cup Y}(s_n)) =$ (IH) $a(\pi_X(s_1), \dots, \pi_X(s_n)) = \pi_X(s)$. If $s = f(s_1, \dots, s_n)$ with $f \in \mathrm{dom}(\pi)$ and $X \cap FV(s) = \emptyset$, then also $(X \cup Y) \cap FV(s) = \emptyset$, so either $\pi_{X \cup Y}(s) = f_\pi(\pi_{X \cup Y}(s_{i_1}), \dots, \pi_{X \cup Y}(s_{i_k})) =$ (IH) $f_\pi(\pi_X(s_{i_1}), \dots, \pi_X(s_{i_k})) = \pi_X(s)$, or $\pi_{X \cup Y}(s) = \pi_{X \cup Y}(s_i) =$ (IH) $\pi_X(s_i) = \pi_X(s)$. $\square$

Proof (long proof of Lemma 2(3)). By induction on l. If $l = x\uparrow$, $\pi(l) = l$ (by (1)) is a weak pattern. Otherwise $l = f(l_1, \dots, l_n)$ and by induction hypothesis all l_i are weak patterns. This proves the lemma whether $\pi(l) = f(\pi(l_1), \dots, \pi(l_n))$ or $\pi(l) = f_\pi(\pi(l_{i_1}), \dots, \pi(l_{i_k}))$ or $\pi(l) = \pi(l_i)$. $\square$

Proof (long proof of Lemma 2(4)). Given term s and set X, we need to see that $s \sqsupseteq_{AF} \pi_X(s)$. Perform induction on the size of s. If $s = \lambda x.\, s'$, then $\pi_X(s) = \lambda x.\, \pi_{X \cup \{x\}}(s')$. By the induction hypothesis $s' \sqsupseteq_{AF} \pi_{X \cup \{x\}}(s')$, so $s \sqsupseteq_{AF} \pi_X(s)$ by monotonicity of $\sqsupseteq_{AF}$. If $s = f(s_1, \dots, s_n)$ with $f \in F_c^\# \cup \mathcal{V}$, then by the induction hypothesis each $s_i \sqsupseteq_{AF} \pi_X(s_i)$, so $s \sqsupseteq_{AF} s' := f(\pi_X(s_1), \dots, \pi_X(s_n))$. Now $\pi_X(s)$ is either $f(\pi_X(s_1), \dots, \pi_X(s_n))$ or $f_\pi(\pi_X(s_{i_1}), \dots, \pi_X(s_{i_k}))$ or $\pi_X(s_i)$. In the first case $s' = \pi_X(s)$, in the second two $s' \sqsupseteq_{AF} \pi_X(s)$ in a single step. $\square$

Proof (long proof of Lemma 2(5)). Define (only in this proof) $a(s_1, \dots, s_n)!!t$ if either $t = a_\pi(s_{i_1}, \dots, s_{i_k})$ and $\pi(a) = [i_1, \dots, i_k]$, or $t = s_i$ if $\pi(a) = i$. By definition $\sqsupseteq_{AF}$ is the smallest monotonic and transitive relation containing !!. Consider, however, the alternative relation $\geq$:

1. $\lambda x.\, s \geq \lambda x.\, t$ if $s \geq t$
2. $a(s_1, \dots, s_n) \geq t$ if one of:
   (a) $t = a(t_1, \dots, t_n)$ with $s_1 \geq t_1, \dots, s_n \geq t_n$ for $a \in \mathcal{F}_\pi \cup \mathcal{V}$
   (b) $t = a_\pi(t_{i_1}, \dots, t_{i_k})$ with $s_{i_j} \geq t_{i_j}$ for all j, and $\pi(a) = [i_1, \dots, i_k]$
   (c) $s_i \geq t$ and $\pi(a) = i$

$\geq$ is evidently contained in $\sqsupseteq_{AF}$, by induction on its definition. If $\sqsupseteq_{AF}$ is also contained in $\geq$ the two relations are equal, and the statement follows. We know that $\sqsupseteq_{AF}$ is contained in $\geq$ if we can prove that $\geq$ is reflexive, transitive and monotonic, and contains !!. Reflexivity and monotonocity are evident with induction on the size of the term, using points 1 and 2a of the definition of $\geq$ (in the base case, $a \geq a$ for base type constants and variables by rule 2a), and using reflexivity it is also clear that $\geq$ contains !!. Transitivity, $\forall s, t, r[s \geq t \geq r \Rightarrow s \geq r]$, follows by induction over the size of s, as we see below:

Let $s \geq t \geq r$. If $s \geq t$ by rule 1 then $t \geq r$ by the same rule, and $s \geq r$ by the induction hypothesis. Otherwise write $s = a(s_1, \dots, s_n)$. If $s \geq t$ by rule 2c then it holds because some $s_i \geq t$; by the induction hypothesis $s_i \geq r$ and therefore $s \geq r$ by the same rule. If $s \geq t$ by rule 2b then $t = a_\pi(t_{i_1}, \dots, t_{i_k})$; the $t \geq r$

step can only be by rule 2a, so $r = a_\pi(r_{i_1}, \dots, r_{i_k})$ with each $s_{i_j} \geq t_{i_j} \geq r_{i_j}$ and therefore by induction $s_{i_j} \geq r_{i_j}$; $s \geq r$ by rule 2b. Finally, if $s \geq t$ by rule 2a, write $t = a(t_1, \dots, t_n)$ with each $s_i \geq t_i$. Consider why $t \geq r$. If this is also by rule 2a, $r = a(r_1, \dots, r_n)$ with each $s_i \geq t_i \geq r_i$, so by the induction hypothesis $s_i \geq r_i$ and thus $s \geq r$ by 2a. If it is by 2b $r = a_\pi(r_{i_1}, \dots, r_{i_k})$ and by the induction hypothesis always $s_{i_j} \geq r_{i_j}$; $s \geq r$ by that same rule 2b. Finally, if $t \geq r$ by 2c there is i such that $\pi(a) = i$ and $s_i \sqsupseteq_{AF} t_i \sqsupseteq_{AF} r$; by the induction hypothesis $s_i \sqsupseteq_{AF} r$ and thus $t \geq r$ by 2c. □

Proof (long proof of Lemma 2(6)). Perform induction on the pre-term $s\gamma$, ordered by the union of $\rightarrow_\beta$ with the proper subterm relation (this is not $\rhd$; s is a subterm of $\lambda x.\, s$). It has been demonstrated in other sources that this union is well-founded. On pre-terms, the subterm relation is closed under substitution.

First, consider $s = \lambda x.\, s'$. By (5) $t = \lambda x.\, t'$ and $s' \sqsupseteq_{AF} t'$; by induction $s'\gamma \sqsupseteq_{AF} t'\delta$, and therefore $s'\gamma \sqsupseteq_{AF} t'\delta$. Otherwise, write $s = a(s_1, \dots, s_n)$. Once more (5) provides three possible forms for t. If $\pi(a) = [i_1, \dots, i_k]$ and $t = a_\pi(t_{i_1}, \dots, t_{i_k})$ with each $s_{i_j} \sqsupseteq_{AF} t_{i_j}$, then by induction each $s_{i_j}\gamma \sqsupseteq_{AF} t_{i_j}\delta$, so similarly $s\gamma = a_\pi(s_{i_1}\gamma, \dots, s_{i_k}\gamma) \sqsupseteq_{AF} a_\pi(t_{i_1}\delta, \dots, t_{i_k}\delta) = t\delta$. If $\pi(a) = i$ and $s_i \sqsupseteq_{AF} t$, then by the induction hypothesis $s_i\gamma \sqsupseteq_{AF} t\delta$. So assume $t = a(t_1, \dots, t_n)$ with each $s_i \sqsupseteq_{AF} t_i$. If $a \notin \mathrm{dom}(\gamma)$ (so also $a \notin \mathrm{dom}(\delta)$), $s\gamma = a(s_1\gamma, \dots, s_n\gamma) \sqsupseteq_{AF} a(t_1\delta, \dots, t_n\delta) = t\delta$ by the induction hypothesis on the $s_i\gamma$. If $a \in \mathrm{dom}(\gamma)$ and $n = 0$, $s\gamma = \gamma(a) \sqsupseteq_{AF} \delta(a) = t\delta$. If $a \in \mathrm{dom}(\gamma)$ and $n > 0$, write $\gamma(a) = \lambda y_1 \dots y_n.\, q$ and (using (5)) $\delta(a) = \lambda y_1 \dots y_n.\, q'$ with $q \sqsupseteq_{AF} q'$. By the $\rightarrow_\beta$ part of the induction hypothesis $q[y_1 := s_1\gamma, \dots, y_n := s_n\gamma] \sqsupseteq_{AF} q'[y_1 := t_1\delta, \dots, y_n := t_n\delta]$ if each $s_i\gamma \sqsupseteq_{AF} t_i\delta$, which is the case by the subterm part of the induction hypothesis. Since (as terms) $s\gamma = q[\boldsymbol{y} := \boldsymbol{s}\gamma]$ and $t\delta = q'[\boldsymbol{y} := \boldsymbol{t}\delta]$, the lemma holds. □

Proof (long proof of Lemma 2(7)). By induction on the pre-term $s\gamma$, ordered by the union of $\rightarrow_\beta$ and the subterm relation. If s is an abstraction we use (2) to rewrite $\pi_{X \cup Y}(\lambda x.\, q) = \lambda x.\, \pi_{X \cup Y \cup \{x\}}(q)\pi_X(\gamma)$ to $\lambda x.\, \pi_{X \cup Y \cup \{x\}}(q)\pi_{X \cup \{x\}}(\gamma)$, which by induction hypothesis $\sqsupseteq_{AF}$ $\lambda x.\, \pi_{X \cup \{x\}}(q\gamma) = \pi_X(\lambda x.\, q\gamma)$. Otherwise let $s = a(s_1, \dots, s_n)$ with $a \in \mathcal{F}_c^\# \cup \mathcal{V}$. If $FV(s) \cap Y = \emptyset$ then $FV(\pi_{X \cup Y}(s)) \cap Y = \emptyset$ as well, so $\pi_{X \cup Y}(s)\pi_X(\gamma) = \pi_{X \cup Y}(s) = \pi_X(s)$ (by (2)), $= \pi_X(s\gamma)$. If $a \in FV(s)$ and $n = 0$, $\pi_{X \cup Y}(s)\pi_X(\gamma) = \pi_X(\gamma)(a) = \pi_X(\gamma(a)) = \pi_X(s\gamma)$. So assume $FV(s) \cap Y \neq \emptyset$ and $n > 0$; write $\pi_{X \cup Y}(s) = a(\pi_{X \cup Y}(s_1), \dots, \pi_{X \cup Y}(s_n))$. If $a \notin Y$ we have $\pi_{X \cup Y}(s)\pi_X(\gamma) = a(\pi_{X \cup Y}(s_1)\pi_X(\gamma), \dots, \pi_{X \cup Y}(s_n)\pi_X(\gamma))$, which by the induction hypothesis $\sqsupseteq_{AF}$-reduces to $a(\pi_X(s_1\gamma), \dots, \pi_X(s_n\gamma)) = \pi_X(s\gamma)$. If $a \in Y$ let $\gamma(a) = \lambda y_1 \dots y_n.\, q$ and define $\delta := [y_1 := s_1\gamma, \dots, y_n := s_n\gamma]$. Since $\pi_{X \cup Y}(s_i)\pi_X(\gamma) \sqsupseteq_{AF} \pi_X(s_i\gamma)$ for all i by the induction hypothesis, $\pi_{X \cup Y}(s)\pi_X(\gamma) = \pi_{X \cup \{\boldsymbol{y}\}}(q)[y_1 := \pi_{X \cup Y}(s_1)\pi_X(\gamma), \dots, y_n := \pi_{X \cup Y}(s_n)\pi_X(\gamma)]$ $\sqsupseteq_{AF}$ $\pi_{X \cup \{\boldsymbol{y}\}}(q)\pi_X(\delta)$. By the $\rightarrow_\beta$ part of the induction hypothesis, this term $\sqsupseteq_{AF}$-reduces to $\pi_X(q\delta) = \pi_X(s\gamma)$. □

Proof (long proof of Lemma 2(8)). Perform induction on the size of s. For an abstraction, $\pi_X(\lambda x.\, q)\pi(\gamma) = \lambda x.\, \pi_{X \cup \{x\}}(q)\pi(\gamma) \sqsupseteq_{AF} \lambda x.\, \pi_{X \cup \{x\}}(t\gamma)$ by induction hypothesis, $= \pi_X((\lambda x.\, q)\gamma)$. Otherwise, write $s = a(s_1, \dots, s_n)$ with $a \in \mathcal{F}_c^\# \cup \mathcal{V}$.

If $a \in \mathrm{Dom}(\gamma)$, let $\gamma(a) = \lambda \boldsymbol{y}.\, q$ and define $\delta = [y_1 := s_1\gamma, \ldots, y_n := s_n\gamma]$. By induction hypothesis $\pi_X(s_i)\pi(\gamma) \sqsupseteq_{AF} \pi_X(s_i\gamma)$ for all i, so $\pi_X(s)\pi(\gamma) \sqsupseteq_{AF} \pi_{\{\boldsymbol{y}\}}(q)\pi_X(\delta) = \pi_{X\cup\{\boldsymbol{y}\}}(q)\pi_X(\delta)$ by (2), $\sqsupseteq_{AF} \pi_X(q\delta) = \pi_X(s\gamma)$ by (7). Alternatively, if $a \in \mathrm{dom}(\pi)$ and $X \cap FV(s) = \emptyset$, either $\pi_X(s)\pi(\gamma) = a_\pi(\pi_X(s_{i_1})\pi(\gamma), \ldots, \pi_X(s_{i_k})\pi(\gamma))$ or $\pi_X(s)\pi(\gamma) = \pi_X(s_i)\pi(\gamma)$. Since X is disjunct from the range of γ, also $X \cap FV(s\gamma) = \emptyset$ so $\pi_X(s\gamma) = a_\pi(\pi_X(s_{i_1}\gamma), \ldots, \pi_X(s_{i_k}\gamma))$ or $\pi_X(s\gamma) = \pi_X(s_i\gamma)$ respectively. In both cases, the induction hypothesis supplies $\pi_X(s)\pi(\gamma) \sqsupseteq_{AF} \pi_X(s\gamma)$. Finally, if $a \notin \mathrm{Dom}(\gamma) \cup \mathrm{Dom}(\pi)$, or $X \cap FV(s) \neq \emptyset$ then $\pi_X(s)\pi(\gamma) = a(\pi_X(s_1)\pi(\gamma), \ldots, \pi_X(s_n)\pi(\gamma)) \sqsupseteq_{AF} a(\pi_X(s_1\gamma), \ldots, \pi_X(s_n\gamma))$ by induction hypothesis, which either $= \pi_X(s\gamma)$ (if $a \notin \mathrm{dom}(\pi)$ or $X \cap FV(s\gamma)$ is also non-empty), or reduces to it with a single $\sqsupseteq_{AF}$ step on the top term. □

Proof (long proof of Lemma 2(9)). By induction on the structure of s. If s is (the eta-long form of) a variable x, $\pi(s)\pi(\gamma) = s\pi(\gamma)$ by (1), $= \pi(\gamma)(x) = \pi(\gamma(x)) = \pi(s\gamma)$. Alternatively, if $s = f(s_1, \ldots, s_n)$ with $f \in \mathcal{F}_c^\#$, by the induction hypothesis each $\pi(s_i)\pi(\gamma) = \pi(s_i\gamma)$; this proves the lemma whether $f \notin \mathrm{Dom}(\pi)$, $\pi(f) = [i_1, \ldots, i_k]$, or if $\pi(f) = i$ with s_i of base type (for example in the last case, $\pi(s)\pi(\gamma) = \pi(s_i)\pi(\gamma) = \pi(s_i\gamma) = \pi(s\gamma)$). □

Proof (long proof of Lemma 2(10a)). As stated in the proof of Lemma 2(10), we will first prove: *if $s \rhd t$ and s has base type and $FV(s) \cap X \neq \emptyset$, then not $t \sqsupseteq_{AF} \pi_X(s)$.*

First we prove that always $FV(s) \cap X = FV(\pi_X(s)) \cap X$, by induction on the form of s. If $X \cap FV(s) = \emptyset$, then no element of X can occur in $FV(\pi_X(s))$ either (π does not create new variables), so $FV(s) \cap X = FV(\pi_X(s)) \cap X = \emptyset$. If $s = a(s_1, \ldots, s_n)$ with $X \cap FV(s) \neq \emptyset$, then $\pi_X(s) = a(\pi_X(s_1), \ldots, \pi_X(s_n))$. Therefore, $FV(\pi_X(s)) \cap X = \{a | a \in X\} \cup \bigcup_{1 \leq i \leq n} FV(\pi_X(s_i)) \cap X =$ (IH) $\{a | a \in X\} \cup \bigcup_{1 \leq i \leq n} FV(s_i) \cap X = FV(s) \cap X$. Finally, if $s = \lambda y.\, q$ then $FV(\pi_X(s)) \cap X = FV(\lambda y.\, \pi_{X\cup\{y\}}(q)) \cap X = (FV(\pi_{X\cup\{y\}}(q)) \setminus \{y\}) \cap X = (FV(\pi_{X\cup\{y\}}(q)) \cap (X \cup \{y\})) \setminus \{y\} =$ (IH) $(FV(q) \cap (X \cup \{y\}) \setminus \{y\} = (FV(q) \setminus \{y\}) \cap X = FV(s) \cap X$.

To prove the statement, let s be a base type term $a(s_1, \ldots, s_n)$ and assume that $FV(s) \cap X \neq \emptyset$ and $s \unrhd t \sqsupseteq_{AF} \pi_X(s)$; we will see that $s = t$, by induction on the size of s. We can safely assume that t is the smallest $\rhd$-subterm of s with $t \sqsupseteq_{AF} \pi_X(s)$; if $t = s$ we are done, so assume towards a contradiction that some $s_i \unrhd t$. Write $\pi_X(s) = a(\pi_X(s_1), \ldots, \pi_X(s_n))$. By (5), $t = a(t_1, \ldots, t_n)$ with each $t_j \sqsupseteq_{AF} \pi_X(s_j)$ (the other two forms are impossible since a is unfiltered and t minimal). But some $s_i \unrhd t$, so $s_i(\boldsymbol{c}) \unrhd t \rhd t_i(\boldsymbol{c}) \sqsupseteq_{AF} \pi_X(s_i)(\boldsymbol{c})$, which by (7) $\sqsupseteq_{AF} \pi_X(s_i(\boldsymbol{c}))$. But $FV(s_i) \cap X \supseteq FV(t) \cap X \supseteq FV(\pi_X(s)) \cap X = FV(s) \cap X \neq \emptyset$, and $s_i(\boldsymbol{c})$ has as many symbols as s_i; we can apply the induction hypothesis to find that $s_i = t_i$, which leads to a contradiction because $s_i \unrhd t$! □

Proof (long proof of Lemma 2(10)). Since $s \sqsupseteq_{AF} q$, q might have one of three forms by (5). It cannot have the third form, since $s_i \sqsupseteq_{AF} q \sqsupseteq_{AF} \pi_X(s_i)$ would contradicting (10a). Nor can it have the second form, as then q would be headed by f_π, which $\pi_X(s)$ is not, and therefore still some $s_{i_j} \sqsupseteq_{AF} \pi_X(s)$. Thus, $q = f(q_1, \ldots, q_n)$ with each $s_i \sqsupseteq_{AF} q_i$. Again we cannot have some $s_i \sqsupseteq_{AF} q_i \sqsupseteq_{AF} \pi_X(s)$, and $\pi_X(s)$ is headed by f, so by point 5 each $q_i \sqsupseteq_{AF} \pi_X(s_i)$. □

Proofs relating to Lemma 3

Lemma 3 is probably the most fundamental Lemma of this paper, as it demonstrates how we get rid of the rather awkward requirement 3 of Theorem 2 (or the corresponding requirement in Theorem 3). We separate the definition given in the short proof in the text, and prove two lemmas about it, which combine to give Lemma 3.

Definition 12. *Given terms v, w over $\mathcal{F}_c^\#$ and u over $\mathcal{F}_\pi$, we say u* preserves v in w *if:*

– *if $w = v$, then $u \sqsupseteq_{AF} h\pi(v)$*
– *if $w \rhd v$, then:*
  - *if $w = \lambda x. w'$, then $u = \lambda x. u'$ and $u'[x := c]$ preserves v in $w'[x := c]$*
  - *if $w = a(w_1, \dots, w_n)$ for $a \in \mathcal{F}_c^\# \cup \mathcal{V}$, then $u = a(u_1, \dots, u_n)$ and u_i preserves v in w_i for each $i \leq n$.*

A statement "u preserves v in w" implies that if $w \unrhd v$, then also $u \unrhd \cdot \sqsupseteq_{AF} h\pi(v)$, and if the first $\unrhd$ is strict, so is the second. Moreover, if $w \unrhd v$ then w and u have roughly the same form, at least in those subterms containing v. Thus, we aim to see that $\pi(p)\pi(\gamma)$ preserves t in $p\gamma$.

Lemma 10. *Suppose u preserves v in w. Then $u \unrhd \cdot \sqsupseteq_{AF} h\pi(v)$ if $w = v$ and $u \rhd \cdot \sqsupseteq_{AF} h\pi(v)$ if $w \rhd v$.*

Proof. By induction on the size of w. If $w = v$, then $u \sqsupseteq_{AF} h\pi(v)$ by definition. Otherwise, assume $w \rhd v$ (as there is nothing to prove if this is not the case either). If w is an abstraction $\lambda x. w'$ then $u = \lambda x. u'$ with $u'[x := c]$ preserving v in $w'[x := c]$. By the induction hypothesis $u \rhd u'[x := c] \unrhd \cdot h\pi(v)$. Alternatively, if $w = a(w_1, \dots, w_n)$ with $f \in \mathcal{F}_c \cup \mathcal{V}$, then $u = a(u_1, \dots, u_n)$ with each u_i preserving v in w_i. Since $w \rhd v$ there must be some i with $w_i \unrhd v$; by the induction hypothesis $u \rhd u_i \unrhd \cdot \sqsupseteq_{AF} h\pi(v)$. □

Lemma 11. *Given terms s, s', base type term t and substitutions γ, γ' on the same domain X such that:*

1. *s, t and each $\gamma(x)$ are terms over $\mathcal{T}(\mathcal{F}_c^\#)$; s' and each $\gamma'(x)$ are in $\mathcal{T}(\mathcal{F}_\pi)$;*
2. *$s \sqsupseteq_{AF} s' \sqsupseteq_{AF} \pi_X(s)$ and all $\gamma(x) \sqsupseteq_{AF} \gamma'(x) \sqsupseteq_{AF} \pi(\gamma(x))$;*
3. *s' preserves t in s and each $\gamma'(x)$ preserves t in $\gamma(x)$;*

Then $s'\gamma'$ preserves t in $s\gamma$.

Proof. Perform induction on the pre-term $s\gamma$, ordered with the union of $\rightarrow_\beta$ and the subterm relation, and assume that $s\gamma \unrhd t$ (if not, we are done anyway). Assume also $FV(s) \cap X \neq \emptyset$ (if not, $s'\gamma = s'$ preserves t in $s = s\gamma$ by requirement 3). If s is an abstraction $\lambda x. w$ then by Lemma 2(5), $s' = \lambda x. w'$ with $w \sqsupseteq_{AF} w' \sqsupseteq_{AF} \pi_{X \cup \{x\}}(w)$, and noting that $s\gamma \neq t$ (different types) we conclude with the induction hypothesis. Otherwise $s = a(w_1, \dots, w_n)$ with $a \in \mathcal{F}_c^\# \cup \mathcal{V}$

and by Lemma 2(10) $s' = a(w'_1, \dots, w'_n)$ with each $w_i \sqsupseteq_{AF} w'_i \sqsupseteq_{AF} \pi_X(w_i)$. Applying the induction hypothesis on all $w_i, w'_i, \gamma, \gamma'$, we find that $w'_i\gamma'$ always preserves t in $w_i\gamma$. If $a \in X$ let $\gamma(x) = \lambda \boldsymbol{y}.\, r$, and $\gamma'(a) = \lambda \boldsymbol{y}.\, r'$ (by Lemma 2(5), $r \sqsupseteq_{AF} r' \sqsupseteq_{AF} \pi_{\{\boldsymbol{y}\}}(r)$). The Lemma follows by the $\rightarrow_\beta$ part of the induction hypothesis on r, r', δ, δ', where $\delta(y_i) = w_i\gamma$ and $\delta'(y_i) = w'_i\gamma$ (by Lemma 2(6) $\delta(y_i) \sqsupseteq_{AF} \delta'(y_i) \sqsupseteq_{AF} \pi_X(w_i)\pi(\gamma)$, which by Lemma 2(7) $\sqsupseteq_{AF}$ $\pi(w_i\gamma) = \pi(\delta(y_i))$). If $a \notin X$ then $s'\gamma' = a(w'_1\gamma', \dots, w'_n\gamma')$. Since each $w'_i\gamma'$ preserves t in $w_i\gamma$ the second requirement of preserving is met. The first, $s'\gamma' \sqsupseteq_{AF}$ $h\pi(t)$ if $s\gamma = t$, holds because $s'\gamma' \sqsupseteq_{AF} a(\pi_X(w_1)\pi(\gamma), \dots, \pi_X(w_n)\pi(\gamma)) \sqsupseteq_{AF}$ $a(\pi(w_1\gamma), \dots, \pi(w_n\gamma)) = h\pi(s\gamma)$, the first step by assumption 2, and the second by Lemma 2(7). □

Lemma 3 follows from these two Lemmas:

Proof (proof of Lemma 3). Apply Lemma 11 on $s, \pi_X(s), t, \gamma, \pi(\gamma)$; due to Lemma 2(4) indeed $s \sqsupseteq_{AF} \pi_X(s)$ and $\gamma \sqsupseteq_{AF} \pi(\gamma)$, and $\pi_X(s)$ preserves t in s because t does not occur in s; each $\pi(\gamma(x))$ preserves t in $\gamma(x)$ for the same reason. We derive that $\pi_X(s)\pi(\gamma)$ preserves t in $s\gamma$, so by Lemma 10 $\pi_X(s)\pi(\gamma) \trianglerighteq \cdot \sqsupseteq_{AF} h\pi(t)$. □

Proofs relating to Theorem 4

In Theorem 4 we must show that the ordering associated with a reduction pair over the filtered rules is also a reduction pair, and fits the requirements of Theorem 3. So, given a reduction pair $(\succ, \succeq)$ over $\mathcal{T}(\mathcal{F}_\pi)$ such that $\succeq$ contains $\sqsupseteq_{AF}$. Define its associated pair $(>, \geq)$ as a pair of relations over $\mathcal{T}(\mathcal{F}_c^\#)$ as follows: $s > t$ if $\pi(s) \succ \pi(t)$, and $s \geq t$ if $\pi(s) \succeq \pi(t)$.

Lemma 12. *$>$ and $\geq$ are compatible.*

Proof. Let $s > t \geq r$. That is, $\pi(s) \succ \pi(t) \succeq \pi(r)$. Then $\pi(s) \succ \pi(r)$ because $\succ$ and $\succeq$ are compatible, and therefore $s > r$. □

Lemma 13. *$\geq$ is base-monotonic*

Proof. Let s_j have base type, and $s_j \geq s'_j$, so $\pi(s_j) \succeq \pi(s'_j)$. For $a \in \mathcal{F}_c^\# \cup \mathcal{V}$, either $a \notin \mathrm{dom}(\pi)$ or $\pi(a) = [i_1, \dots, i_n]$ or $\pi(a) = i$. In the first two cases $\pi(a(s_1, \dots, s_i, \dots, s_n)) \succeq \pi(a(s_1, \dots, s'_i, \dots, s_n))$ by (base-)monotonicity of $\succeq$, or by reflexivity if $j \notin \{i_1, \dots, i_n\}$. In the last case, if $j = i$ then $\pi(f(s_1, \dots, s_n)) =$ $\pi(s_i) \succeq \pi(s'_i) = \pi(f(s_1, \dots, s'_i, \dots, s_n))$ by assumption, if $j \neq i$ by reflexivity. □

Lemma 14. *For weak patterns s and subtitutions γ on $\mathrm{dom}(s)$, if sRt then $s\gamma R t\gamma$ ($R \in \{>, \geq\}$).*

Proof. If $s > t$, then $\pi(s) \succ \pi(t)$, so $\pi(s\gamma) = \pi(s)\pi(\gamma)$ (Lemma 2(9)), $\succ \pi(t)\pi(\gamma)$ (Lemma 2(3) and because $\succ$ is weakly stable), $\sqsupseteq_{AF} \pi(t\gamma)$ (Lemma 2(8)), which proves $s\gamma > t\gamma$. The same reasoning works for $\geq$. □

Proof (long proof of Theorem 4). Given a reduction pair $(\succ, \succeq)$ on $\mathcal{T}(\mathcal{F}_\pi)$ which satisfies the requirements of the theorem, Lemmas 12, 13 and 14 show that the associated pair is a reduction pair on $\mathcal{T}(\mathcal{F}_c^\#)$ (because $\sqsupseteq_{AF}$ is contained in $\succeq$). Moreover, requirements 1 and 2 of Theorem 3 are clearly met; the only difficulty is condition 3. Consider suitable l, p, γ, t and write $p = y(p_1, \dots, p_n)$ and $\gamma(y) = \lambda x_1 \dots x_n . q$. Using Lemmas 2(9) and 2(3), $\pi(l\gamma) = \pi(l)\pi(\gamma) \succ \pi(p)\pi(\gamma) = y(\pi(p_1), \dots, \pi(p_n))\pi(\gamma) = \pi_{\{\boldsymbol{x}\}}(q)[\boldsymbol{x} := \pi(\boldsymbol{p})\pi(\gamma)] =: M$. Since t neither occurs in q nor in any $p_i\gamma$ we can apply Lemma 3 and find that $M \rhd \cdot \sqsupseteq_{AF} h\pi(t)$, which $\succeq \pi(t^\#)$ by the last two requirements of Theorem 4. Thus, either by transitivity of $\succ$ or compatibility of $\succ$ and $\succeq$ we have $\pi(l\gamma) \succ \pi(t^\#)$. □