A Reduction Ordering for Higher-Order Terms

Jürgen Avenhaus[1], Carlos Loría-Sáenz[2], and Joachim Steinbach[3]

[1] Universität Kaiserslautern
Fachbereich Informatik, Postfach 3049
67653 Kaiserslautern (Germany)

[2] Instituto Tecnologico de Costa Rica
Departamento de Computaction
Apartado 159-7050 Cartago (Costa Rica)

[3] Technische Universität München
Institut für Informatik
80290 München (Germany)

Abstract. We investigate one of the classical problems of the theory of term rewriting, namely termination. We present an ordering for comparing higher-order terms that can be utilized for testing termination and decreasingness of higher-order conditional term rewriting systems. The ordering relies on a first-order interpretation of higher-order terms and a suitable extension of the RPO.

1 Motivation

Term rewriting systems (TRSs) can be considered as a powerful theoretical model for reasoning about functional and logic programming in an abstract way, independently of a particular programming language. In such an approach to computer programming, logic and functional programs are represented by means of executable specifications essentially consisting of conditional equations. The operational semantics of these specifications is defined by term rewriting and equation solving, respectively. The extension of first-order logic to higher-order logic by means of (universally quantified) conditional equations enormously increases the expressive power of the specifications and permits an efficient operationalization.

Equations on terms specify (pure) term replacement criteria that can be performed on formulae or expressions. However, equations are ambiguous since they can be used in two different directions. Rewriting rules are directed equations which can be applied only in one direction by imposing some *orientation* on the terms in the equation. Such an orientation should imply that the repeated replacement of subterms in a given expression using the rules eventually stops yielding a simplest term or *normal form* unable to be further simplified. If equations represent, for instance, axioms of a theory (i.e. facts about some abstract entity) oriented replacement can be used to prove equality of formulae or expressions. In other words, to solve the so-called *word problem* of the equational theory we only need to *reduce* expressions to a common normal form. In the case that equations describe an equational (logic or functional) program, oriented replacement can be used for giving semantic values to expressions, i.e. for computing. The normal form of an expression represents its semantic value. Under these conditions, one of the fundamental problems associated with TRSs is the following one: Are the rules really simplifying, i.e. does rewriting always lead to a normal form after a finite sequence of rule applications?

The theory of first-order term rewriting offers different alternatives to solve this problem, i.e. to guarantee termination (see, for example, [Der87]). For introductions to TRSs see, for example, [DJ90, AM90]. See also [DO90] for a survey on conditional rewriting systems.

In [Lor93], criteria for testing confluence and termination of some classes of higher-order conditional TRSs (HCTRSs) have been developed (see also [AL94]). HCTRSs naturally extend (unconditional) HTRSs as defined in [Nip91]. [Lor93] follows the approach of [Nip91] by combining term rewriting and the λ-calculus. For the λ-calculus the reader is referred to [HS86].

Example 1. In order to illustrate how both computational paradigms – λ-calculus and TRSs – interact, let us consider an algebraic specification of the addition $(+)$ on natural numbers (nat):

$$\mathsf{0}+y \;\rightarrow\; y \tag{1}$$

$$\mathsf{s}(x)+y \;\rightarrow\; \mathsf{s}(x+y) \tag{2}$$

where as usual $\mathsf{s}\colon\ nat \rightarrow nat$ is the *successor function*, $\mathsf{0}\colon\ \rightarrow nat$ and $+\colon (nat, nat) \rightarrow nat$. A higher-order expression like

$$\lambda x.(x+x) \tag{3}$$

can be used for 'locally' specifying the *double* function and so we could compute, for instance,

$$\begin{array}{ll} & (\lambda x.(x+x))\mathsf{s}(\mathsf{0}) \\ \rightarrow_\beta & \mathsf{s}(\mathsf{0})+\mathsf{s}(\mathsf{0}) \\ \rightarrow_R & \mathsf{s}(\mathsf{0}+\mathsf{s}(\mathsf{0})) \\ \rightarrow_R & \mathsf{s}^2(\mathsf{0}) \end{array}$$

$\diamond$

Note that the R-reductions reduce the $+$ operator. The β-reduction represents the parameter-passing mechanism of programming languages.

In addition to local declarations like $\lambda x.(x+x)$ in Example 1, rewriting becomes more interesting when we permit specifying higher-order rules like the following ones

$$\mathsf{fold}([], X, z) \;\rightarrow\; z \tag{4}$$

$$\mathsf{fold}(x{::}L, Y, z) \;\rightarrow\; \mathsf{fold}(L, Y, Y(z, x)) \tag{5}$$

In the approach of [Nip91] to higher-order rewriting the dynamical parameter-passing (i.e. β-reduction) has priority over term rewriting using rules. In fact, one can consider β-reduction as part of the substitution operation. This represents the first characteristic of the combination of higher-order and term rewriting used in [Nip91]. The second characteristic is that terms can only be rewritten if they are in $\beta\eta$-long form. This decision does not represent any computational restriction and it simplifies much of the technical work. Thus, an expression like

$$\mathsf{fold}([1,2],\lambda xy.(x+y),0) \tag{6}$$

will be reduced by proceeding in the following way: The expression is in β-normal form, hence a rule can be applied. Using rule (5),

$$\mathsf{fold}([2],\lambda xy.(x+y),(\lambda xy.(x+y)\ 0\ 1)). \tag{7}$$

can be produced. Now we apply β-normalization yielding

$$\mathsf{fold}([2],\lambda xy.(x+y),(0+1)). \tag{8}$$

As in this example, β-normalization is already performed after having applied the term rewriting indicated by the selected rule. Repeating this process, first using rule (5) followed by rule (4), we yield the algebraic term

$$(0+1)+2 \tag{9}$$

which could further reduced to $\mathsf{s}^3(0)$ after translating 1 to $\mathsf{s}(0)$ and 2 to $\mathsf{s}(\mathsf{s}(0))$ and using rules (1) and (2). In other words, the term in (6) denotes the computation of the sum of all the elements contained in the list [1,2].

In order to verify termination of HCTRSs, higher-order terms must be compared, in some way. We develop a method for performing that task which is based on suitable extensions of first-order techniques. More specifically, we construct an ordering called HPO (for *higher-order path ordering*) by means of an extension of the recursive path ordering RPO ([Der82]) so that non-algebraic terms can also be compared. Termination and decreasingness of HCTRSs are then achieved as usual in the first-order case: the associated rewrite relation is required to be included in this ordering guaranteeing some kinds of monotonicity properties wrt. term structure and substitutions.

Our method is essentially based on a first-order interpretation of higher-order terms in $\beta\eta$-long form. Therefore, some of the principal properties and proof-techniques existing in the first-order case can also be used for the HPO. This is true, for instance, for well-foundedness. As we will see, the definition of the HPO looks like the standard definition of the RPO. In fact, they coincide on algebraic terms. Thus, the remaining properties making the HPO a quasi-ordering can be obtained by appropriately extending the corresponding proofs for simplification orderings. The HPO is based on a *precedence* $\succsim$ (i.e. a quasi-ordering over the set of operators $\mathcal{F}$) and the term interpretation makes use of an appropriate extension of it. This extension, called $\succsim_0$ includes new constants like binders $\lambda\overline{x_m}$ (which will be treated as unary operators). We also define a congruence $\sim_{\mathsf{HPO}}$ such that $\succsim_{\mathsf{HPO}}$ (i.e. $\sim_{\mathsf{HPO}} \cup \succ_{\mathsf{HPO}}$) becomes a quasi-ordering over $\mathcal{T}(\mathcal{F},\mathcal{V})$.

In order to illustrate the main ideas, let us consider the terms

$$s \equiv \mathsf{map}(\lambda x.\mathsf{g}(\mathsf{h}(x)),y{::}L) \quad \text{and} \tag{10}$$

$$t \equiv \mathsf{g}(\mathsf{h}(y)){::}\mathsf{map}(\lambda x.\mathsf{f}(\mathsf{h}(x)),L). \tag{11}$$

We explain how to prove $s \succ_{\mathsf{HPO}} t$ using the rules defining the HPO which will be exposed later. First of all, suppose a precedence $\succsim$ to be given satisfying

$$\mathsf{map} \succ :: \text{ and } \quad \mathsf{g} \sim \mathsf{f}. \tag{12}$$

Using this and proceeding as in a typical RPO-comparison, we have to prove

$$s \succ_{\mathsf{HPO}} \mathsf{g}(\mathsf{h}(y)) \quad \text{and} \tag{13}$$
$$s \succ_{\mathsf{HPO}} \mathsf{map}(\lambda x.\mathsf{f}(\mathsf{h}(x)), L) \tag{14}$$

in order to show $s \succ_{\mathsf{HPO}} t$. In the first case, we make use of an *extended subterm property* of the HPO: $\mathsf{g}(\mathsf{h}(y))$ is not exactly a subterm of s but it can be constructed using β-reduction and proper subterms of s:

$$(\lambda x.\mathsf{g}(\mathsf{h}(x))y)\downarrow_\beta \equiv \mathsf{g}(\mathsf{h}(y)). \tag{15}$$

This proves (13). For (14), the multisets of direct subterms must be compared:

$$\forall t \in \{\lambda x.\mathsf{f}(\mathsf{h}(x)), L\}\colon\ \exists s \in \{\lambda x.\mathsf{g}(\mathsf{h}(x)), y{::}L\}\colon s \succ_{\mathsf{HPO}} t \tag{16}$$

which is true since

$$\lambda x.\mathsf{g}(\mathsf{h}(x)) \sim_{\mathsf{HPO}} \lambda x.\mathsf{f}(\mathsf{h}(x)) \quad \text{and} \tag{17}$$
$$y{::}L \succ_{\mathsf{HPO}} L. \tag{18}$$

For (17), note that the λ-binders are the same and $\mathsf{g}(\mathsf{h}(x)) \sim_{\mathsf{HPO}} \mathsf{f}(\mathsf{h}(x))$ because $\mathsf{g} \sim \mathsf{f}$. The relation (18) is a consequence of the property that the HPO is an extension of the (modified) subterm ordering. Therefore, $s \succ_{\mathsf{HPO}} t$ holds.

The paper is organized as follows. In the following section, some necessary notations about terms, substitutions, reductions, patterns and orderings are presented. Subsequently, an extension of the signature (integrating λ-binders as unary operators) and the term algebra together with an appropriate precedence extension are introduced. Section 4 deals with the presentation of a generalized subterm ordering since the original subterm ordering is not sufficient for λ-expressions (see (13) and (15)). The definition as well as the main properties (e.g., the well-foundedness) of the new ordering HPO for higher-order terms are contained in Section 5. Although the HPO is not compatible with substitutions in general, we show that this property holds for a great class of HCTRSs occurring in practice (see Subsection 5.3). This provides a useful syntactic test for verifying termination and decreasingness as presented in Section 6.

This report is mainly a polished version of Chapter 6 in [Lor93], with some extensions (see Section 7). For related work, see Section 8. Proofs of all results can be found in the appendix.

2 Preliminaries

First of all, we give some elementary notations which are necessary for defining our ordering. Unless otherwise specified, we use the notations of [Nip91] and [AL94].

2.1 Terms

A *signature sig* is a triple $(\mathcal{B}, \mathcal{F}, \mathcal{V})$ where $\mathcal{B}$ is a set of basic types, $\mathcal{F}$ is a family of sets of *function symbols* (also called *operators*) and $\mathcal{V}$ is a family of sets of *variables*, both families based on $\mathcal{B}$. An *atom* is either a constant or a variable. The set $\mathcal{T}(\mathcal{F},\mathcal{V})$ of terms over *sig* is defined as usual. Additionally, $\lambda\overline{x_n}.t$ denotes $\lambda x_1.(\lambda x_2.(\cdots(\lambda x_n.t)\cdots))$ whereas $\mathsf{f}(\overline{t_n})$ denotes $\mathsf{f}(t_1,\ldots,t_n)$. The sets $\mathcal{FV}ar(t)$ and $\mathcal{BV}ar(t)$ stand for the set of *free variables* and the set of *bound variables* of a term t, respectively.

2.2 Substitutions

A (well-typed) *substitution* σ is a mapping from $\mathcal{V}$ into $\mathcal{T}(\mathcal{F},\mathcal{V})$ such that $x\sigma$ and x have the same type for each $x \in \mathcal{V}$ and only for a finite subset of $\mathcal{V}$ $x\sigma \not\equiv x$ holds. The *domain* of σ, denoted by $\mathcal{D}om(\sigma) \equiv \{x \in \mathcal{V} \mid x\sigma \not\equiv x\}$ is such a set. A substitution σ *has a property* $\mathcal{P}$ on terms if $x\sigma$ satisfies $\mathcal{P}$ for every $x \in \mathcal{D}om(\sigma)$.

2.3 Reductions

We recapitulate the standard reductions and the corresponding equalities (conversions) of the typed λ-calculus.

As well-known, the α-conversion is used to rename bound variables. Thus, terms only differing in bound variables are α-equivalent.

The β-reduction represents the parameter-passing mechanism of programming languages: bound variables in expressions (i.e. formal parameters of functions) are replaced within the scope by their binding values (i.e. actual parameters) permitting computation to take place:

$$s \rightarrow_\beta t \quad \text{if} \quad \begin{cases} \exists p \in \mathcal{P}os(s)\colon\ s|_p \equiv (\lambda x.s_1)s_2 \ \text{ and} \\ t \equiv s[\{x \leftarrow s_2\}s_1]_p \end{cases}$$

The position p used for the reduction is called a *redex*. One can show that $\rightarrow_\beta$ is confluent and terminating (see [HS86]). A term t is in β-normal form (β-*nf*) iff

$$t \equiv \lambda\overline{x_p}.\mathsf{F}t_1\ldots t_m \tag{19}$$

for some $p, m \geq 0$ such that m is less than or equal to the arity of F and such that the t_i are in β-*nf*. A term $\lambda\overline{x_p}.\mathsf{F}t_1\ldots t_m$ in β-*nf* is called *rigid* if F is either a bound variable or an operator. Otherwise, it is called *flexible*.

The η-reduction, finally, is used to represent equality of functions in the mathematical sense, i.e. $\mathsf{f} =_\eta \lambda x.\mathsf{f}(x)$.

Example 2.

$$\begin{aligned} \mathsf{f}((\lambda xy.\mathsf{g}(x,y))\mathsf{a}) &\rightarrow_\alpha \mathsf{f}((\lambda xz.\mathsf{g}(x,z))\mathsf{a}) \\ &\rightarrow_\beta \mathsf{f}(\lambda z.\mathsf{g}(\mathsf{a},z)) \\ &\equiv \mathsf{f}(\lambda z.\mathsf{ga}z) \\ &\rightarrow_\eta \mathsf{f}(\mathsf{ga}) \end{aligned}$$

$\diamond$

A very useful representation of terms in β-*nf* is the so-called $\beta\eta$-long form. To obtain it, terms in β-*nf* will recursively be completed by adding new variables and λ-abstractions using ${}_\eta\!\longleftarrow$ (i.e. the inverse of $\longrightarrow_\eta$) to preserve the type of the original term.

Definition 1 ($\beta\eta$-Long Form) *Let t be a term such that $t\!\downarrow_\beta \equiv \lambda\overline{x_p}.\mathsf{F}t_1 \ldots t_m$. The $\beta\eta$*-long form *$\widehat{t}$ of t is recursively defined by*

$$\begin{cases} t & \text{if } t \text{ is a first-order atom} \\ \lambda x_1 \ldots x_p x_{m+1} \ldots x_n.\mathsf{F}(\widehat{t_1}, \ldots, \widehat{t_m}, \widehat{x_{m+1}}, \ldots, \widehat{x_n}) & \text{otherwise} \end{cases}$$

where

- $\mathsf{F}\colon (\tau_1, \ldots, \tau_m, \tau_{m+1}, \ldots, \tau_n) \rightarrow \tau'$ *and*
- $x_{m+1}, \ldots, x_n$ *are new variables such that* $\forall k \in [1, n-m]$: $x_{m+k}\colon \rightarrow \tau_{m+k}$

A term t is in $\beta\eta$-long form if $\widehat{t} =_\alpha t$. $\diamond$

Let $\mathcal{T}_{\beta\eta}(\mathcal{F},\mathcal{V}) = \{t \in \mathcal{T}(\mathcal{F},\mathcal{V}) \mid t$ in $\beta\eta$-long form$\}$. A substitution σ is in $\beta\eta$-long form if $x\sigma \in \mathcal{T}_{\beta\eta}(\mathcal{F},\mathcal{V})$ for all $x \in \mathcal{D}om(\sigma)$.

For example, let t be $(\mathsf{map}\ X)$ where $\mathsf{map}\colon (\tau_1 \rightarrow \tau_2, list(\tau_1)) \rightarrow list(\tau_2)$ is an operator and $X\colon \tau_1 \rightarrow \tau_2$ a second order variable. Then $\widehat{t} \equiv \lambda L.\mathsf{map}(\lambda z.X(z), L)$ since $\widehat{X} \equiv \lambda z.X(z)$ and $\widehat{L} \equiv L$. In this case, $L\colon \rightarrow list(\tau_1)$ and $z\colon \rightarrow \tau_1$ are the newly generated variables. Note that $\widehat{t} \xrightarrow{*}_\eta t$. This example illustrates that terms in $\beta\eta$-long form possess a nearly algebraic structure where λ-binders are unary operators and bound variables are constants. Assuming that terms are in this special form will significantly simplify several technical problems associated with extending algebraic results to the higher-order case.

2.4 Patterns

A term t in β-*nf* is a (well-typed higher-order) *pattern* if every free occurrence of a variable X in t is a subterm of the form $X(\overline{x_n})$ such that $\overline{x_n}$ is η-equivalent to a list of distinct bound variables in $\mathcal{BV}ar(t)$.

For example, the terms $\lambda x.\mathsf{f}(\lambda y.X(y,x),\mathsf{a})$ and $\mathsf{map}(X,L)$ are patterns whereas $Y(\mathsf{a})$ and $\lambda xy.Z(x,x,y)$ are not patterns.

2.5 Orderings

The *proper subterm relation* $\rhd$ is defined as $s \rhd t$ iff $\exists p \neq \Lambda$: $s|_p \equiv t$. On terms in $\beta\eta$-long form we define $s \rhd_b t$ iff $s \rhd t$ and t is of basic type and not a bound variable in s. In the sequel, $\rhd_b$ will be used as the proper subterm relation, only. Thus, for simplicity, $\rhd$ will be used instead of $\rhd_b$.

Definition 2 (Recursive Path Ordering, [Der82]) *Let $\succsim$ be a quasi-precedence on $\mathcal{F}$ and s,t be two terms. Then the* recursive path ordering RPO *is defined as*

$$s \succ_{\mathsf{RPO}} t \quad \textit{iff} \quad \begin{array}{lll} (1) & \mathcal{A}rgs(s) \succsim^{\text{m}}_{\mathsf{RPO}} \{t\} & \textit{or} \\ (2) & \mathcal{H}ead(s) \succ \mathcal{H}ead(t) \ \wedge \ \{s\} \succ^{\text{m}}_{\mathsf{RPO}} \mathcal{A}rgs(t) & \textit{or} \\ (3) & \mathcal{H}ead(s) \sim \mathcal{H}ead(t) \ \wedge \ \mathcal{A}rgs(s) \succ^{\text{m}}_{\mathsf{RPO}} \mathcal{A}rgs(t) & \end{array}$$

where $\mathcal{H}ead(t)$ *and* $\mathcal{A}rgs(t)$ *denote the leading operator and the multiset of the arguments of* t*, respectively.* $\succ^{\text{m}}_{\mathsf{RPO}}$ *and* $\succsim^{\text{m}}_{\mathsf{RPO}}$ *stand for the multiset extension of* $\succ_{\mathsf{RPO}}$ *and* $\succsim_{\mathsf{RPO}}$*, respectively.* ◇

3 Relations on Extended Terms

We assume that in every context the bound variables occurring in any term are contained in the set $\mathcal{X}$ ($\mathcal{X} \subset \mathcal{V}$). More formally, $\mathcal{V} = \mathcal{X} \cup \mathcal{V}_0$ where

$$\mathcal{X} \cap \mathcal{V}_0 = \emptyset$$
$$\forall t \in \mathcal{T}(\mathcal{F},\mathcal{V}){:}\ \ \mathcal{BV}ar(t) \subset \mathcal{X} \quad \wedge \quad \mathcal{FV}ar(t) \cap \mathcal{X} \equiv \emptyset.$$

In addition, we assume by α-conversion that no $x \in \mathcal{X}$ is used more than once for λ-abstraction in any term t.

Definition 3 (Extended Symbols) *The set of* extended symbols $\mathcal{F}_0$ *is defined as* $\mathcal{F}_0 \ \equiv \ \mathcal{F} \cup \mathcal{X} \cup \mathcal{K}$ *where* $\mathcal{K} \equiv \bigcup_{m>0 \wedge x \in \mathcal{X}} \{\lambda \overline{x_m}\}$ *is the set of all valid* λ*-binders.* ◇

Note that $\mathcal{T}(\mathcal{F}_0,\mathcal{V}_0)$ is a set of first-order terms. Any $t \in \mathcal{T}_{\beta\eta}(\mathcal{F},\mathcal{V})$ can be identified with some term in $\mathcal{T}(\mathcal{F}_0,\mathcal{V}_0)$. By this identification $\mathcal{T}_{\beta\eta}(\mathcal{F},\mathcal{V}) \subseteq \mathcal{T}(\mathcal{F}_0,\mathcal{V}_0)$ holds.

Our intention is to establish a one-to-one correspondence between terms in $\beta\eta$-long form and 'algebraic' terms built using operators in $\mathcal{F}_0$ and variables in $\mathcal{V}$. Thus, for instance, we identify $\lambda x.\mathsf{g}(\mathsf{h}(x))$ with $\lambda x(\mathsf{g}(\mathsf{h}(x)))$ where λx is a unary operator in $\mathcal{K}$ and the bound variable x is considered as a constant. These terms are called *extended terms*.

First of all, an extension of quasi-orderings on $\mathcal{F}$ to $\mathcal{F}_0$ for constructing the HPO is needed.

Definition 4 ($\succsim_0$) *Given a quasi-ordering* $\succsim$ *over* $\mathcal{F}$*, we extend it to a relation* $\succsim_0$ *over* $\mathcal{F}_0$ *as follows:*

$$\mathsf{f} \succsim_0 \mathsf{g} \quad \textit{iff} \quad \begin{array}{l} - \ \mathsf{f} \succsim \mathsf{g} \ \textit{or} \\ - \ \mathsf{f} \in \mathcal{F} \ \textit{and} \ \mathsf{g} \in \mathcal{K} \cup \mathcal{X} \ \textit{or} \\ - \ \mathsf{f},\mathsf{g} \in \mathcal{X} \ \textit{and} \ \mathsf{f} =_\tau \mathsf{g} \ \textit{or} \\ - \ \mathsf{f} \equiv \lambda \overline{x_m},\ \mathsf{g} \equiv \lambda \overline{y_m} \ \textit{and} \ \overline{x_m} =_\tau \overline{y_m} \end{array}$$

where $\overline{x_m} =_\tau \overline{y_m}$ *iff* x_i *and* y_i *have the same type for every* $i \in [1, m]$*. We assume that precedences over* $\mathcal{F}$ respect arities and types of operators*, i.e.* $\mathsf{f} \sim \mathsf{g}$ *and* $\mathsf{f}{:}\ (\tau_1, \ldots, \tau_n) \to \tau$ *implies* $\mathsf{g}{:}\ (\tau_{\pi(1)}, \ldots, \tau_{\pi(n)}) \to \tau$ *where* π *is any permutation of* $\{1, \ldots, n\}$*. As usual, we also define the* strict part *of* $\succsim_0$ *as* $\mathsf{f} \succ_0 \mathsf{g}$ *iff* $(\mathsf{f} \succsim_0 \mathsf{g})$ *and* $(\mathsf{g} \not\succsim_0 \mathsf{f})$ *and the associated equivalence relation as* $\mathsf{f} \sim_0 \mathsf{g}$ *iff* $(\mathsf{f} \succsim_0 \mathsf{g})$ *and* $(\mathsf{g} \succsim_0 \mathsf{f})$*.* ◇

Note that the elements of $\mathcal{K}$ are minimal symbols wrt. $\succsim_0$ as well as λ-binders and higher-order variables will be comparable only if they are $=_\tau$-equal. Furthermore, $\mathsf{f} \sim_0 \mathsf{g}$ implies f and g to have the same arity.

Proposition 1 (Well-Foundedness of $\succ_0$) *Let $\succsim$ be a quasi-ordering over $\mathcal{F}$. Then*

1. *$\succsim_0$ is a quasi-ordering over $\mathcal{F}_0$.*
2. *$\succ_0$ is well-founded over $\mathcal{F}_0$ iff $\succ$ is well-founded over $\mathcal{F}$.* $\diamond$

Now, we construct the congruence $\sim_{\mathsf{HPO}}$ over terms built using symbols in $\mathcal{F}_0$ and generated by $\sim_0$. First, we need an auxiliary definition.

Definition 5 (Type-Respecting Permutation) *A permutation π of the set $\{1,\ldots,m\}$ is a* type-respecting permutation *of a list of m terms $\overline{t_m}$ if t_i and $t_{\pi(i)}$ have the same type for each $i \in [1,m]$.* $\diamond$

Definition 6 ($\sim_{\mathsf{HPO}}$) *Let $\sim_{\mathsf{HPO}}$ be the minimal congruence on extended terms in $\mathcal{T}(\mathcal{F}_0,\mathcal{V}_0)$ such that*

$$\mathsf{f}(\overline{s_m}) \sim_{\mathsf{HPO}} \mathsf{g}(\overline{t_m}) \quad \textit{iff} \quad \mathsf{f} \sim_0 \mathsf{g} \ \textit{and} \ s_i \sim_{\mathsf{HPO}} t_{\pi(i)}$$

for every $i \in [1,m]$, where π is any type-respecting permutation of $(\overline{t_m})$. $\diamond$

Note that $\lambda x(\mathsf{f}(\mathsf{h}(x))) \sim_{\mathsf{HPO}} \lambda y(\mathsf{g}(\mathsf{h}(y)))$ holds if $\mathsf{f} \sim \mathsf{g}$ (which implies $\mathsf{f} \sim_0 \mathsf{g}$) and $x =_\tau y$ (which implies $\lambda x \sim_{\mathsf{HPO}} \lambda y$). $\sim_{\mathsf{HPO}}$ has the following property.

Lemma 1 (Compatibility of $\sim_{\mathsf{HPO}}$). *If $\sigma \sim_{\mathsf{HPO}} \sigma'$ and $s \sim_{\mathsf{HPO}} t$ then $s\sigma \sim_{\mathsf{HPO}} t\sigma'$ for all substitutions σ and σ' of basic type[4] and all s,t in $\beta\eta$-long form.* $\diamond$

4 Generalized Subterm Ordering

For constructing the HPO a generalization of the subterm relation $\rhd$ is needed. In order to motivate the main ideas let us first discuss the definition of the RPO (see Definition 2). Only for the purpose of illustration let us define the relation $\rhd_{ex}$ as

$$\mathsf{f}(\overline{t_n}) \rhd_{ex} t_i$$

for every $i \in [1,n]$. Note that the RPO-schema 'decomposes' the problems using $\rhd_{ex}$. Our aim essentially consists of obtaining a suitable generalization of $\rhd_{ex}$ on terms in $\beta\eta$-long form (equivalently on extended terms). Thus, for instance, not only the comparison $\mathsf{f}(\lambda x.\mathsf{g}(x),\mathsf{a}) \rhd_{ex} \mathsf{a}$ should hold, but also $\mathsf{f}(\lambda x.\mathsf{g}(x),\mathsf{a}) \rhd_{ex} \mathsf{g}(\mathsf{a})$. In the latter case, the term $\mathsf{g}(\mathsf{a})$ is obtained by combining the direct subterms $\lambda x.\mathsf{g}(x)$ and a, and by applying (first-order) β-reduction. For constructing $\rhd_{ex}$ we will introduce two auxiliary relations ($\rhd_0$ and $\rhd_1$) which help us showing that $\rhd_{ex}$ is a well-founded ordering.

Once we have defined $\rhd_{ex}$, we will present the HPO following almost the same RPO-schema exposed above.

[4] That is, x is of basic type for every $x \in \mathcal{D}om(\sigma)$.

Definition 7 **($\rhd_{ex}$)** *Let $\rhd$ be the subterm relation on $\mathcal{T}(\mathcal{F}_0,\mathcal{V}_0)$. The relation $\rhd_0$ on extended terms is defined as*

$$\mathsf{f}(\overline{s_n}) \rhd_0 t$$

iff $\mathsf{f}\in\mathcal{F}$ *and* $\exists i\in[1,n]$: $s_i \equiv \lambda\overline{x_m}.s'$, $m>0$ *and* $t \equiv s'\tau$ *where* $\mathcal{D}om(\tau) \subseteq \{x_k \mid 1 \leq k \leq m,\ x_k \text{ first-order}\}$ *and for all* $x\in\mathcal{D}om(\tau)$: $x\tau \trianglelefteq s_j$ *for some* $j \neq i$, $x\tau$ *not below a free variable. Now,* $\rhd_1$ *is the union of* $\rhd_0 \cup \rhd$, *i.e.*

$$\rhd_1 \;=\; (\rhd \cup \rhd_0)^+ \quad \textit{and}$$

$$\rhd_{ex} \;=\; \rhd_1^+ \qquad \diamond$$

Example 3.

$$\begin{array}{lll} \mathsf{f}(\lambda xy.\mathsf{g}(x,\lambda z.\mathsf{s}(\mathsf{h}(y,z))),\mathsf{a},\mathsf{b}) & \rhd_1 & \mathsf{g}(\mathsf{a},\lambda z.\mathsf{s}(\mathsf{h}(\mathsf{b},z))) \\ & \rhd_1 & \mathsf{s}(\mathsf{h}(\mathsf{b},\mathsf{a})) \\ & \rhd & \mathsf{h}(\mathsf{b},\mathsf{a}) \end{array}$$

Thus, $\mathsf{f}(\lambda xy.\mathsf{g}(x,\lambda z.\mathsf{s}(\mathsf{h}(y,z))),\mathsf{a},\mathsf{b}) \rhd_{ex} \mathsf{h}(\mathsf{b},\mathsf{a})$. $\diamond$

Obviously, the definition of $\rhd_{ex}$ makes sense only if it is a well-founded ordering.

Lemma 2 (Basic Properties of $\rhd_{ex}$).

1. $\rhd \subset \rhd_1 \subset \rhd_{ex}$
2. $\rhd_{ex}$ *is globally finite.*
3. $s \rhd_{ex} t$ *implies* $s\sigma \rhd_{ex} t\sigma$ *for every* σ *of basic type.* $\diamond$

5 The Higher-Order Path Ordering

This section deals with the introduction of the ordering HPO for comparing higher-order terms. Although the definition is apparently different from the RPO, a careful observation shows that the HPO is essentially based on the RPO: λ-binders are treated as unary operators and higher-order as well as bound variables are considered as constants. Thus, in the definition of the HPO, we implicitly interpret terms in $\beta\eta$-long form in $\mathcal{T}(\mathcal{F},\mathcal{V})$ as first-order terms in $\mathcal{T}(\mathcal{F}_0,\mathcal{V}_0)$, i.e. any term of the form $\lambda\overline{x_m}.t$ will be considered as $\lambda\overline{x_m}(t)$ where $\lambda\overline{x_m}$ represents a unary operator in $\mathcal{K}$.

5.1 Definition

Definition 8 (HPO) *Let $\succsim$ be a quasi-precedence on $\mathcal{F}$. The* higher-order path ordering HPO *over terms s,t in $\mathcal{T}(\mathcal{F}_0,\mathcal{V}_0)$ is defined as*

$$\begin{array}{lll} s \succ_{\mathsf{HPO}} t & \textit{iff} & (1)\ \exists s'\colon s \rhd_{ex} s' \succsim_{\mathsf{HPO}} t \ \textit{ or} \\ & & (2)\ \mathcal{H}ead(s) \succsim_0 \mathcal{H}ead(t) \ \wedge\ \forall t'\colon t \rhd_1 t' \leadsto s \succ_{\mathsf{HPO}} t' \ \textit{ and} \\ & & \quad (2.1)\ \mathcal{H}ead(s) \succ_0 \mathcal{H}ead(t) \ \textit{ or} \\ & & \quad (2.2)\ \mathcal{H}ead(s) \sim_0 \mathcal{H}ead(t) \ \wedge\ \mathcal{A}rgs(s) \succ_{\mathsf{HPO}}^{\text{(m)}} \mathcal{A}rgs(t) \end{array} \qquad \diamond$$

Note that the restriction of the HPO to algebraic terms coincides with the RPO. In such a case, $\rhd_{ex}$ plays the role of the subterm property. Thus, (1) of Definition 8 is an extended subterm property as explained in Section 4. Note also that $\rhd \subseteq \rhd_{ex} \subseteq \succ_{\mathsf{HPO}}$.

Example 4. Consider the following HCTRS. We will prove its termination using the HPO based on the precedence $\mathsf{map} \succ ::$, $\mathsf{d} \succ \mathsf{s}$, $\mathsf{doublelist} \succ \mathsf{map}, \mathsf{d}$.

$$\begin{array}{lrcl}
1: & \mathsf{d}(0) & \rightarrow & 0 \\
2: & \mathsf{d}(\mathsf{s}(x)) & \rightarrow & \mathsf{s}(\mathsf{s}(\mathsf{d}(x))) \\
3: & \mathsf{map}(\lambda x.X(x),[]) & \rightarrow & [] \\
4: & \mathsf{map}(\lambda x.X(x),y::L) & \rightarrow & X(y)::\mathsf{map}(\lambda x.X(x),L) \\
5: & \mathsf{doublelist}(L) & \rightarrow & \mathsf{map}(\lambda x.\mathsf{d}(x),L)
\end{array}$$

The comparisons proceed as follows:

1: $\mathsf{d}(0) \succ_{\mathsf{HPO}} 0$ by (1) since $\mathsf{d}(0) \rhd_{ex} 0$.
2: $\mathsf{d}(\mathsf{s}(x)) \succ_{\mathsf{HPO}} \mathsf{s}(\mathsf{s}(\mathsf{d}(x)))$ using (2.1) first and then (2.2).
3: $\mathsf{map}(\lambda x.X(x),[]) \succ_{\mathsf{HPO}} []$ by (1).
4: First, $\mathsf{map}(\lambda x.X(x),y::L) \succ_{\mathsf{HPO}} X(y)$ by (1). Further, $\mathsf{map}(\lambda x.X(x),y::L) \succ_{\mathsf{HPO}} \mathsf{map}(\lambda x.X(x),L)$ by (2.2). So, $\mathsf{map}(\lambda x.X(x),y::L) \succ_{\mathsf{HPO}} X(y)::\mathsf{map}(\lambda x.X(x),L)$ by (2.2).
5: $\mathsf{doublelist}(L) \succ_{\mathsf{HPO}} \lambda x.\mathsf{d}(x)$ by (2.1) using $\mathsf{doublelist} \succ_0 \lambda x, \mathsf{d}, x$ and $\mathsf{doublelist}(\mathsf{L}) \succ_{\mathsf{HPO}} L$ by (1). By (2.1), we get $\mathsf{doublelist}(L) \succ_{\mathsf{HPO}} \mathsf{map}(\lambda x.\mathsf{d}(x),L)$. $\diamond$

5.2 Well-Foundedness

Now, we prove the main result of this paper, namely, that $\succsim_{\mathsf{HPO}}$ is a well-founded quasi-ordering. We presuppose that an arbitrary but fixed precedence $\succsim$ is used to generate the HPO. Furthermore, $\succsim$ is extended to a precedence $\succsim_0$ on $\mathcal{F}_0$, as indicated in Definition 4.

Theorem 1 (Compatibility of $\sim_{\mathsf{HPO}}$ and $\rhd_{ex}$) *If $s \sim_{\mathsf{HPO}} t \rhd_{ex} w$ then $s \rhd_{ex} v \sim_{\mathsf{HPO}} w$ for some v.* $\diamond$

Proposition 2 (Technical Property) *For every s,t and every σ of basic type:*
1. *If $t\sigma \rhd_{ex} t''$ then $\exists t'$: $t \rhd_{ex} t' \ \wedge \ t'\sigma \equiv t''$*
2. *$s \succ_{\mathsf{HPO}} t$ implies $s\sigma \succ_{\mathsf{HPO}} t\sigma$* $\diamond$

Theorem 2 (Quasi-Ordering Properties of the HPO)
1. *If $s \succ_{\mathsf{HPO}} t \sim_{\mathsf{HPO}} w$ then $s \succ_{\mathsf{HPO}} w$*
2. *If $s \sim_{\mathsf{HPO}} t \succ_{\mathsf{HPO}} w$ then $s \succ_{\mathsf{HPO}} w$*
3. *If $s \succ_{\mathsf{HPO}} t \succ_{\mathsf{HPO}} w$ then $s \succ_{\mathsf{HPO}} w$ (i.e. the HPO is transitive)* $\diamond$

By means of the next result we achieve well-foundedness and compatibility of the HPO wrt. the term structure.

Theorem 3 (Simplification Properties of the HPO) *Let s and $t \in \mathcal{T}_{\beta\eta}(\mathcal{F},\mathcal{V})$ such that s and t are of the same basic type and $s \succ_{\mathsf{HPO}} t$. Then*

1. $\mathsf{f}(t_1,\ldots,t_{i-1},t,t_{i+1},\ldots,t_n) \rhd_1 w$ *implies* $\mathsf{f}(t_1,\ldots,t_{i-1},s,t_{i+1},\ldots,t_n) \succ_{\mathsf{HPO}} w$ *and*
2. $C[s] \succ_{\mathsf{HPO}} C[t]$ *for every well-typed context* $C[.]$. ◇

Corollary 1 (Well-Foundedness of the HPO) *The relation $\succsim_{\mathsf{HPO}}$ is a quasi-ordering over terms in $\beta\eta$-long form in $\mathcal{T}(\mathcal{F},\mathcal{V})$ such that its strict part $\succ_{\mathsf{HPO}}$ is well-founded and properly contains $\rhd$.* ◇

5.3 Stability

Our next aim is to prove that the HPO is – in a restrictive form – compatible wrt. substitutions. For this purpose we first need some auxiliary results.

Proposition 3 ($\sim_{\mathsf{HPO}}$ and Flexible Patterns)

1. $X(\overline{s_n}) \unrhd_{ex} t$ *iff* $s_i \unrhd_{ex} t$ *for some i or* $t \equiv X(\overline{s_n})$
2. $X(\overline{w_k}) \succsim_{\mathsf{HPO}} t$ *iff* $X(\overline{w_k}) \sim_{\mathsf{HPO}} t$ *or* $w_j \sim_{\mathsf{HPO}} t$ *for some j whenever $\overline{w_k}$ are bound variables in $\mathcal{X}$.*
3. $X(\overline{v_k}) \sim_{\mathsf{HPO}} X(\overline{w_k})$ *implies* $X(\overline{v_k})\sigma{\downarrow}_\beta \sim_{\mathsf{HPO}} X(\overline{w_k})\sigma{\downarrow}_\beta$ *whenever $\overline{v_k}, \overline{w_k}$ are bound variables in $\mathcal{X}$ for every σ in $\beta\eta$-long form.* ◇

Lemma 3. *Let s be a basic pattern, $t \in \mathcal{T}_{\beta\eta}(\mathcal{F},\mathcal{V})$ and σ a substitution in $\beta\eta$-long form. If $s \rhd_{ex} t$ then $s\sigma{\downarrow} \rhd_{ex} t\sigma{\downarrow}$.* ◇

For the next lemma we need the homeomorphic embedding $\succ_{\mathsf{EMB}}$ on $\mathcal{T}(\mathcal{F}_0,\mathcal{V}_0)$. Note that $\succ_{\mathsf{EMB}} \subseteq \succ_{\mathsf{HPO}}$ on $\mathcal{T}_{\beta\eta}(\mathcal{F},\mathcal{V})$ since $\rhd \subseteq \succ_{\mathsf{HPO}}$ and the HPO is closed under contexts by Theorem 3.

Lemma 4. *Let $t \equiv \mathsf{f}(\overline{t_n})$, $\mathsf{f} \in \mathcal{F}$ and t, σ in $\beta\eta$-long form. If $t\sigma{\downarrow} \rhd_1 t'$ then $t \rhd_1 s$ and $s\sigma{\downarrow} \succ_{\mathsf{EMB}} t'$ for some s.* ◇

Example 5. Let $t \equiv \mathsf{f}(\lambda v.\mathsf{G}(v),\mathsf{H}(x))$.

1. Assume $\sigma = \{\mathsf{G} \leftarrow \lambda v.\mathsf{a},\ \mathsf{H} \leftarrow \lambda v.\mathsf{h}(x,\mathsf{b})\}$. Then $t\sigma{\downarrow} \equiv \mathsf{f}(\lambda v.\mathsf{a},\mathsf{h}(x,\mathsf{b})) \rhd_1 \mathsf{a} \equiv t'$. We may choose $s \equiv \mathsf{G}(v)$. Then we have $t \rhd_1 s$ and $s\sigma{\downarrow} \equiv \mathsf{a} \equiv t'$.
2. Assume $\sigma = \{\mathsf{H} \leftarrow \lambda v.\mathsf{h}(x,\mathsf{b})\}$. Then $t\sigma{\downarrow} \equiv \mathsf{f}(\lambda v.\mathsf{G}(v),\mathsf{h}(x,\mathsf{b})) \rhd_1 \mathsf{G}(\mathsf{b}) \equiv t'$. We may choose $s \equiv \mathsf{G}(\mathsf{H}(x))$. Then we have $t \rhd_1 s$ and $s\sigma{\downarrow} \equiv \mathsf{G}(\mathsf{h}(x,\mathsf{b})) \succ_{\mathsf{EMB}} \mathsf{G}(\mathsf{b}) \equiv t'$. ◇

By means of these results, we are able to prove

Theorem 4 (Compatibility wrt. Substitutions) *Let s be a basic rigid pattern, $\sigma, s, t \in \mathcal{T}_{\beta\eta}(\mathcal{F},\mathcal{V})$. Then*

1. $s \sim_{\mathsf{HPO}} t$ *implies* $s\sigma{\downarrow} \sim_{\mathsf{HPO}} t\sigma{\downarrow}$
2. $s \succ_{\mathsf{HPO}} t$ *implies* $s\sigma{\downarrow} \succ_{\mathsf{HPO}} t\sigma{\downarrow}$ ◇

In this form we have obtained the necessary properties which ensures that the HPO can be used as a reduction ordering. The next and final step consists of appropriately embedding the rewrite relation into this ordering.

6 A Test for Decreasingness

Using the results of the previous sections, we are able to postulate a sufficient test for termination and decreasingness of HCTRSs.

Theorem 5 (Termination and Decreasingness) *Let $\mathcal{R}$ be an HCTRS. Then*

1. *$\mathcal{R}$ is terminating if $l \succ_{\mathsf{HPO}} r$ for each rule $l \to r$ if C in $\mathcal{R}$.*
2. *$\mathcal{R}$ is decreasing if $l \succ_{\mathsf{HPO}} s$ for every s in $\{r\} \cup \{u, v \mid u = v \in C\}$ for each rule $l \to r$ if C in $\mathcal{R}$.* ◇

Example 4 of Section 5 exhibits an HCTRS whose decreasingness can be verified using this criterion.

Example 6. As another example, let $\mathcal{R}$ be given by

$$\begin{array}{rrcl}
1: & \mathsf{split}(P,[],L_1,L_2) & \to & \mathsf{pair}(L_1,L_2) \\
2: & \mathsf{split}(P,x{::}L,L_1,L_2) & \to & \mathsf{split}(P,L,x{::}L_1,L_2) \text{ if } P(x) \to \mathsf{true} \\
3: & \mathsf{split}(P,x{::}L,L_1,L_2) & \to & \mathsf{split}(P,L,L_1,x{::}L_2) \text{ if } P(x) \to \mathsf{false} \\
4: & \mathsf{partition}(x,L) & \to & \mathsf{split}(\lambda y.(x \leq y),L,[],[])
\end{array}$$

We can verify that $\mathcal{R}$ is decreasing wrt. the HPO based on the precedence $\mathsf{partition} \succ \leq, \mathsf{split}, []$ and $\mathsf{split} \succ \mathsf{pair}, \mathsf{true}, \mathsf{false}$. ◇

7 Extensions

The construction so far is based on the classical recursive path ordering and so shares the restricted power of the RPO. There are other insufficiencies concerning the arguments of higher-order variables in right-hand sides of rewrite rules. In this section, we will demonstrate these problems by examples and propose ideas for solutions.

Example 7. Let

$$\begin{array}{rrcl}
\mathcal{R}: & \mathsf{fold}(\mathsf{F},y,[]) & \to & y \\
& \mathsf{fold}(\mathsf{F},y,z :: l) & \to & \mathsf{fold}(\mathsf{F},\mathsf{F}(y,z),l)
\end{array}$$

Here the HPO fails to orient the second rule because in the right-hand side the second argument of F is greater than that in the left-hand side. But $\mathcal{R}$ is terminating due to the fact that the third argument becomes smaller in each rewrite step. The solution to this problem is to allow a status $stat(\mathsf{f})$ for all $\mathsf{f} \in \mathcal{F}$. This is well-known for reduction orderings on first-order terms, see [Der87, Ste94]. For $\mathcal{R}$, we may choose $stat(\mathsf{f})$ = right-to-left. ◇

Let HPOS denote the obvious modification of the HPO with status. Then we have

Theorem 6 *Each* HPOS *is a reduction ordering.* ◇

The next example shows that the extended subterm relation $\rhd_{ex}$ may be too weak.

Example 8. Let

$$\mathcal{R}: \quad \mathsf{dapply}(\mathsf{F},\mathsf{G},z) \;\rightarrow\; \mathsf{F}(\mathsf{G}(z))$$

The HPO is unable to orient this rule because, roughly speaking, $\rhd_0$ does not allow iterated application of evaluation of subterms: We have $\mathsf{dapply}(\mathsf{F},\mathsf{G},z) \rhd_0 \mathsf{F}(z)$ but not $\mathsf{dapply}(\mathsf{F},\mathsf{G},z) \rhd_0 \mathsf{F}(\mathsf{G}(z))$. We may extend $\rhd_0$ in this direction and so construct a more powerful HPO. This allows to prove the termination of $\mathcal{R}$. ◇

The next example shows that it is necessary to extend $\rhd_0$ even more.

Example 9. Let

$$\mathcal{R}: \quad \begin{array}{rcl} \mathsf{dmap}(\mathsf{F},\mathsf{G},[]) & \rightarrow & [] \\ \mathsf{dmap}(\mathsf{F},\mathsf{G},z :: l) & \rightarrow & \mathsf{G}(z) :: \mathsf{dmap}(\mathsf{F},\mathsf{F}(\mathsf{G}),l) \end{array}$$

The relation $\rhd_0$ is restricted in that $\mathsf{f}(\overline{s_n}) \rhd_0 t$, $t \equiv s_0\tau$, $s_i \equiv \lambda\overline{x_n}.s_0$ only if x is a variable of basic type for all $x \in \mathcal{D}om(\tau)$. In the second rule of $\mathcal{R}$ we have $\mathsf{F} =_\eta \lambda X v.\mathsf{F}(X,v)$, where $X =_\eta \lambda u.X(u)$. We need $\mathsf{dmap}(\mathsf{F},\mathsf{G},z :: l) \rhd_0 \mathsf{F}(\mathsf{G})$ for orienting this rule with the HPO. But, for this we need $X\tau \equiv \lambda u.\mathsf{G}(u)$ and X is not of basic type. We did not succeed up to now to extend the HPO in this direction. Notice that F has an argument of non-basic type, so $\mathsf{dmap}(\mathsf{F},\mathsf{G},z :: l)$ is not 'simple' in the terminology of [LP95]. The approach in [LP95] is unable to deal with this case, too. ◇

8 Related Work

To our knowledge, there are two approaches to prove termination of higher-order rewriting systems, [LP95] and [vdP94]. Our approach dates back to [Lor93].

The approach of [LP95] is similar to ours, the main difference being that it uses the more general construction of [DH93] to construct reduction orderings based on the RPO-schema. In that paper it is proposed to replace comparing f and g in $s \equiv \mathsf{f}(\overline{s_n})$ and $t \equiv \mathsf{g}(\overline{t_m})$ of (2) in the definition of the HPO by comparing weights of s and t, the weight of a term being defined by weights of the function symbols $\mathsf{f} \in \mathcal{F}$. This seems to be more flexible in handling arguments of higher-order variables F. On the other side, the approach needs to show sometimes that $\mathsf{g}(\overline{t_m})$ dominates F. For that the sufficient condition $t_i \equiv \mathsf{F}$, for some i, is used. So $\mathsf{f}(\lambda u.\mathsf{g}(\mathsf{F}(u)),z) \rightarrow \mathsf{F}(z)$ causes some problems. This causes no problem to our approach. So none of the two approaches seems to subsume the other one.

The paper [vdP94] uses semantic arguments to construct reduction orderings in that the $\mathsf{f} \in \mathcal{F}$ are interpreted as strict monotone functions in a *sig*-algebra equipped with a well-founded ordering. So it can be seen as an extension of the polynomial orderings on first-order terms. This approach is quite different from the syntactic orderings used in this paper. So we do not comment in detail on that.

References

[AL94] Jürgen Avenhaus and Carlos Loría-Sáenz. Higher-order conditional rewriting and narrowing. In J.-P. Jouannaud, editor, *Proc. First International Conference on Constraints in Computational Logics*, volume 845 of *LNCS*, pages 269–284, München (Germany), 1994.

[AM90] Jürgen Avenhaus and Klaus E. Madlener. Term rewriting and equational reasoning. In R.B. Banerji, editor, *Formal Techniques in Artificial Intelligence – A Sourcebook*, Studies in Computer Science and Artificial Intelligence 6, pages 1–43. Elsevier Science Publishers B.V. (North-Holland), Amsterdam, 1990.

[Der82] Nachum Dershowitz. Orderings for term rewriting systems. *JTCS*, 17(3):279–301, March 1982.

[Der87] Nachum Dershowitz. Termination of rewriting. *JSC*, 3:69–116, February/April 1987. see also Corrigendum - Termination of rewriting (*JSC*, 4:409-410, 1987) and *1st RTA*, volume 202 of LNCS, pages 180-224, Dijon (France), May 1985.

[DH93] Nachum Dershowitz and Charles Hoot. Topics in termination. In C. Kirchner, editor, *5th RTA*, volume 690 of *LNCS*, pages 198–212, Montreal (Canada), June 1993.

[DJ90] Nachum Dershowitz and Jean-Pierre Jouannaud. *Rewrite systems*, volume B of *Handbook of Theoretical Computer Science*, chapter 6, pages 243–320. Elsevier Science Publisher B.V., 1990.

[DO90] Nachum Dershowitz and Mitsohiro Okada. A rationale for conditional equational programming. *Theoretical Computer Science*, 75:111–138, 1990.

[HS86] J. Roger Hindley and Jonathan P. Seldin. Introduction to combinators and λ-calculus. *Cambridge University Press*, 1986.

[Lor93] Carlos Loría-Sáenz. *A theoretical framework for reasoning about program construction based on extensions of rewrite systems.* PhD Thesis, Fachbereich Informatik, Universität Kaiserslautern (Germany), December 1993.

[LP95] Olav Lysne and Javier Piris. A termination ordering for higher order rewrite systems. *6th RTA*, Kaiserslautern Germany, April 1995, to appear.

[LS92] Carlos Loría-Sáenz and Joachim Steinbach. Termination of combined (rewrite and λ-calculus) systems. In M. Rusinowitch and J.L. Rémy, editors, *3rd CTRS*, volume 656 of *LNCS*, pages 143–147, Pont-à-Mousson (France), July 1992.

[Nip91] Tobias Nipkow. Higher-order critical pairs. In *Proc. 6th LICS*, pages 342–349, Amsterdam (The Netherlands), 1991. IEEE Computer Society Press.

[Ste94] Joachim Steinbach. *Termination of Rewriting – Extensions, Comparison and Automatic Generation of Simplification Orderings.* PhD Thesis, Fachbereich Informatik, Universität Kaiserslautern (Germany), January 1994.

[vdP94] Jaco van de Pol. Termination proof for higher-order rewrite systems. In *Proc. 1st International Workshop on Higher-Order Algebra*, pages 305–325, Amsterdam (The Netherlands), 1994.

A Proofs

Proposition 1 (Well-Foundedness of $\succ_0$) *Let $\succsim$ be a quasi-ordering over $\mathcal{F}$. Then*

1. *$\succsim_0$ is a quasi-ordering over $\mathcal{F}_0$.*
2. *$\succ_0$ is well-founded over $\mathcal{F}_0$ iff $\succ$ is well-founded over $\mathcal{F}$.* $\diamond$

Proof. straightforward by using the corresponding definitions $\square$

Lemma 1 (Compatibility of $\sim_{\mathsf{HPO}}$) *If $\sigma \sim_{\mathsf{HPO}} \sigma'$ and $s \sim_{\mathsf{HPO}} t$ then $s\sigma \sim_{\mathsf{HPO}} t\sigma'$ for all substitutions σ and σ' of basic type and all s, t in $\beta\eta$-long form.* $\diamond$

Proof. By induction on (s, t) using $(\rhd, \rhd)_{lex}$. If $s \sim_{\mathsf{HPO}} t$ and $s \equiv \mathsf{f}(\overline{s_m})$ and $t \equiv \mathsf{g}(\overline{t_m})$ then $\mathsf{f} \sim_0 \mathsf{g}$ and $\forall i \in [1, m]$: $s_i \sim_{\mathsf{HPO}} t_{\pi(i)}$ for a type-respecting permutation π of $(\overline{t_m})$. By induction, we get $s_i\sigma \sim_{\mathsf{HPO}} t_{\pi(i)}\sigma'$. Since σ and σ' are of basic type, $s\sigma \equiv \mathsf{f}(\overline{s_m\sigma})$ and $t\sigma' \equiv \mathsf{g}(\overline{t_m\sigma'})$ hold. By definition of $\sim_{\mathsf{HPO}}$, we get $s\sigma \sim_{\mathsf{HPO}} t\sigma'$. $\square$

Lemma 2 (Basic Properties of $\rhd_{ex}$)

1. *$\rhd \subset \rhd_1 \subset \rhd_{ex}$*
2. *$\rhd_{ex}$ is globally finite.*
3. *$s \rhd_{ex} t$ implies $s\sigma \rhd_{ex} t\sigma$ for every σ of basic type.* $\diamond$

Proof. 1. It follows directly from the corresponding definitions.
2. $\rhd_1$ is finitely branching, i.e. for any s there are only finitely many t with $s \rhd_1 t$. So, $\rhd_{ex}$ is globally finite by König's lemma.
3. obvious by using the corresponding definitions $\square$

Theorem 1 (Compatibility of $\sim_{\mathsf{HPO}}$ and $\rhd_{ex}$) *If $s \sim_{\mathsf{HPO}} t \rhd_{ex} w$ then $s \rhd_{ex} v \sim_{\mathsf{HPO}} w$ for some v.* $\diamond$

Proof. 1. We first prove: If $s \sim_{\mathsf{HPO}} t \rhd_1 w$ then $s \rhd_1 v \sim_{\mathsf{HPO}} w$. Assume $s \equiv \mathsf{f}(\overline{s_n})$ and $t \equiv \mathsf{g}(\overline{t_n})$, so $\mathsf{f} \sim_0 \mathsf{g}$ and $s_{\pi(i)} \sim_{\mathsf{HPO}} t_i$ for some permutation π.
If $w \equiv t_i$ for some i, then we may choose $v \equiv s_{\pi(i)}$. So assume $w \equiv t_0\tau$, $t_i \equiv \lambda\overline{x_m}.t_0$ and $x_j\tau \unlhd t_k$. Then $s_{\pi(i)} \equiv \lambda\overline{y_m}.s_0$ and $s_0 \sim_{\mathsf{HPO}} t_0$. Define σ by $y_j\sigma \equiv s_k|_p$ if $x_j\tau \equiv t_k|_p$ and choose $v \equiv s_0\sigma$. Then $s \rhd_1 v \sim_{\mathsf{HPO}} w$.
2. We now prove the claim of the theorem by induction on $\rhd_{ex}$. Assume $s \sim_{\mathsf{HPO}} t \unrhd_{ex} w' \rhd_1 w$. By induction hypothesis we have $s \rhd_{ex} v' \sim_{\mathsf{HPO}} w' \rhd_1 w$, and by part 1. we have $v' \rhd_1 v \sim_{\mathsf{HPO}} w$. This gives $s \rhd_{ex} v \sim_{\mathsf{HPO}} w$. $\square$

Proposition 2 (Technical Property) *For every s, t and every σ of basic type:*

1. *If $t\sigma \rhd_{ex} t''$ then $\exists t'$: $t \rhd_{ex} t' \;\wedge\; t'\sigma \equiv t''$*
2. *$s \succ_{\mathsf{HPO}} t$ implies $s\sigma \succ_{\mathsf{HPO}} t\sigma$* $\diamond$

Proof. 1. It is simple from the definition of $\rhd_{ex}$.

2. By induction on (s,t) using $(\rhd_{ex},\rhd_{ex})$. Consider the cases on $s \succ_{\mathsf{HPO}} t$ according to the definition of the HPO.
   - If $s \rhd_{ex} s' \sim_{\mathsf{HPO}} t$, then $s\sigma \rhd_{ex} s'\sigma$ by Lemma 1 and Lemma 2. Thus, $s\sigma \succ_{\mathsf{HPO}} t\sigma$.
   - If $s \equiv \mathsf{f}(\overline{s_m})$, $t \equiv \mathsf{g}(\overline{t_n})$, $\mathsf{f} \succsim_0 \mathsf{g}$ and $s \succ_{\mathsf{HPO}} t'$ for each t' such that $t \rhd_{ex} t'$ and either
     (a) $\mathsf{f} \succ_0 \mathsf{g}$ or
     (b) $\mathsf{f} \sim_0 \mathsf{g}$ and $\{s_i\}_{i=1}^n \succ_{\mathsf{HPO}}^{ⓜ} \{t_i\}_{i=1}^n$
     From $s \succ_{\mathsf{HPO}} t'$, we get $s\sigma \succ_{\mathsf{HPO}} t'\sigma$ for every t' such that $t \rhd_{ex} t'$. And if $t\sigma \rhd_{ex} t''$ then we can find t'' such that $t \rhd_{ex} t'$ and $t'\sigma \equiv t''$, by 1. It follows
     $$t\sigma \rhd_{ex} t'' \text{ implies } s\sigma \succ_{\mathsf{HPO}} t'' \tag{20}$$
     That directly implies, $s\sigma \succ_{\mathsf{HPO}} t\sigma$, in case $\mathsf{f} \succ_0 \mathsf{g}$. Hence, suppose $\mathsf{f} \sim_0 \mathsf{g}$. Hence, we have $\{s_i\}_{i=1}^n \succ_{\mathsf{HPO}}^{ⓜ} \{t_i\}_{i=1}^n$. By induction, we get $\{s_i\sigma\}_{i=1}^n \succ_{\mathsf{HPO}}^{ⓜ} \{t_i\sigma\}_{i=1}^n$. That and (20) yield $s\sigma \succ_{\mathsf{HPO}} t\sigma$. □

Theorem 2 (Quasi-Ordering Properties of the HPO)

1. *If* $s \succ_{\mathsf{HPO}} t \sim_{\mathsf{HPO}} w$ *then* $s \succ_{\mathsf{HPO}} w$
2. *If* $s \sim_{\mathsf{HPO}} t \succ_{\mathsf{HPO}} w$ *then* $s \succ_{\mathsf{HPO}} w$
3. *If* $s \succ_{\mathsf{HPO}} t \succ_{\mathsf{HPO}} w$ *then* $s \succ_{\mathsf{HPO}} w$ *(i.e. the* HPO *is transitive)* ◇

Proof. We only prove part 1. The proofs of the remaining cases can analogously be performed. We proceed by induction on (s,t,w) using $(\rhd_{ex},\rhd_{ex},\rhd_{ex})_{lex}$. Let us consider the cases on $s \succ_{\mathsf{HPO}} t$ according to the definition of the HPO. In each case, we assume $s \equiv \mathsf{f}(\overline{s_m})$, $t \equiv \mathsf{g}(\overline{t_n})$, $w \equiv \mathsf{h}(\overline{w_n})$ and $\mathsf{g} \sim_0 \mathsf{h}$ and for each $i \in [1,n]$
$$t_i \sim_{\mathsf{HPO}} w_{\pi(i)} \tag{21}$$
for an arbitrary type-respecting permutation π of $(\overline{t_n})$. From (21)
$$\{t_i\}_{i=1}^n \sim_{\mathsf{HPO}} \{w_i\}_{i=1}^n \tag{22}$$
compared as multisets. Now, to the cases:

- $s \rhd_{ex} s' \sim_{\mathsf{HPO}} t \sim_{\mathsf{HPO}} w$. Then $s \rhd_{ex} s' \sim_{\mathsf{HPO}} w$ and hence $s \succ_{\mathsf{HPO}} w$.
- $\mathsf{f} \succsim_0 \mathsf{g}$ and
  $$\forall\, t'\colon\ t \rhd_{ex} t' \ \text{ implies } \ s \succ_{\mathsf{HPO}} t'. \tag{23}$$
  We first have $\mathsf{f} \succsim_0 \mathsf{h}$. Also, if $t \sim_{\mathsf{HPO}} w \rhd_{ex} w'$ then $t \rhd_{ex} v \sim_{\mathsf{HPO}} w'$ by Theorem 1. Hence, by induction
  $$\forall\, w'\colon\ w \rhd_{ex} w' \text{ implies } \ s \succ_{\mathsf{HPO}} w'. \tag{24}$$
  And, if $\mathsf{f} \succ_0 \mathsf{h}$ then
  $$\{s_i\}_{i=1}^m \succ_{\mathsf{HPO}}^{ⓜ} \{t_i\}_{i=1}^n \sim_{\mathsf{HPO}} \{w_i\}_{i=1}^n \tag{25}$$

Again by induction $\{s_i\}_{i=1}^m \succ_{\mathsf{HPO}}^{\text{(m)}} \{w_i\}_{i=1}^n$. From that and (24), we get $s \succ_{\mathsf{HPO}} w$ as required. □

Theorem 3 (Simplification Properties of the HPO) *Let s and $t \in \mathcal{T}_{\beta\eta}(\mathcal{F},\mathcal{V})$ such that s and t are of the same basic type and $s \succ_{\mathsf{HPO}} t$. Then*

1. $\mathsf{f}(t_1,...,t_{i-1},t,t_{i+1},...,t_n) \rhd_1 w$ *implies* $\mathsf{f}(t_1,...,t_{i-1},s,\ldots,t_{i+1},t_n) \succ_{\mathsf{HPO}} w$ *and*
2. $C[s] \succ_{\mathsf{HPO}} C[t]$ *for every well-typed context* $C[.]$. ◇

Proof. 1. Let s, w be given. We prove the claim by induction on t using $\rhd_{ex}$. There are three cases:

(a) $w \equiv t_j$: We have $\mathsf{f}(t_1,\ldots,t_{i-1},s,t_{i+1},\ldots,t_n) \rhd_1 w$ and the claim holds by $\rhd_1 \subseteq \succ_{\mathsf{HPO}}$.

(b) $w \equiv t$: We have $\mathsf{f}(t_1,\ldots,t_{i-1},s,t_{i+1},\ldots,t_n) \rhd s \succ_{\mathsf{HPO}} t \rhd_1 w$, therefore $\mathsf{f}(t_1,\ldots,t_{i-1},s,t_{i+1},\ldots,t_n) \succ_{\mathsf{HPO}} w$.

(c) $t_j \equiv \lambda\overline{x_m}t'$ and $w \equiv t'\tau$, where $x_k\tau \unlhd t_{j_k}$ or $x_k\tau \unlhd t$: We define σ by $x_k\sigma \equiv x_k\tau$ if not $t \unrhd x_k\tau$ and $x_k\sigma \equiv s$ otherwise. If $t \unrhd x_k\tau$ then $x_k\sigma \succ_{\mathsf{HPO}} x_k\tau$ and the induction hypothesis gives $t'\sigma \succ_{\mathsf{HPO}} t'\tau$. We have $\mathsf{f}(t_1,\ldots,t_{i-1},s,t_{i+1},\ldots,t_n) \succ_{\mathsf{HPO}} w$.

2. By 1. we have $\mathsf{f}(t_1,\ldots,t_{i-1},s,t_{i+1},\ldots,t_n) \succ_{\mathsf{HPO}} v$ whenever $\mathsf{f}(t_1,\ldots,t_{i-1},t,t_{i+1},\ldots,t_n) \rhd_1 v$. This gives $C[s] \succ_{\mathsf{HPO}} v$ whenever $C[t] \rhd_1 v$. Now by (2.2) of Definition 8 we get $C[s] \succ_{\mathsf{HPO}} C[t]$. □

Corollary 1 (Well-Foundedness of the HPO) *The relation $\succsim_{\mathsf{HPO}}$ is a quasi-ordering over terms in $\beta\eta$-long form in $\mathcal{T}(\mathcal{F},\mathcal{V})$ such that its strict part $\succ_{\mathsf{HPO}}$ is well-founded and properly contains $\rhd$.* ◇

Proof. By the previous results, the HPO is a simplification ordering on $\mathcal{T}(\mathcal{F}_0,\mathcal{V}_0)$ and hence terminating. □

Proposition 3 ($\sim_{\mathsf{HPO}}$ and Flexible Patterns)

1. $X(\overline{s_n}) \unrhd_{ex} t$ *iff* $s_i \unrhd_{ex} t$ *for some i or* $t \equiv X(\overline{s_n})$
2. $X(\overline{w_k}) \succsim_{\mathsf{HPO}} t$ *iff* $X(\overline{w_k}) \sim_{\mathsf{HPO}} t$ *or* $w_j \sim_{\mathsf{HPO}} t$ *for some j whenever $\overline{w_k}$ are bound variables in $\mathcal{X}$.*
3. $X(\overline{v_k}) \sim_{\mathsf{HPO}} X(\overline{w_k})$ *implies* $X(\overline{v_k})\sigma{\downarrow_\beta} \sim_{\mathsf{HPO}} X(\overline{w_k})\sigma{\downarrow_\beta}$ *whenever $\overline{v_k}, \overline{w_k}$ are bound variables in $\mathcal{X}$ for every σ in $\beta\eta$-long form.* ◇

Proof. Use the corresponding definitions. □

Definition 9 *Let s be a basic rigid pattern.*

1. *$\Delta_i(s)$ is defined as follows:*

$$\Delta_0(s) \equiv \{t \mid s \rhd t\}$$
$$\Delta_{i+1}(s) \equiv \{t \mid u \in \Delta_i(s), u \rhd_0 v \rhd t\} \cup \Delta_i(s)$$

2. $\sigma \in \Delta_i(s)$ *if* $\forall x \in \mathcal{D}om(\sigma)$: $x \in \mathcal{BV}ar(s)$, *x first-order and* $x\sigma \in \Delta_i(s)$. $\diamond$

Fact 1 *Let s be a basic rigid pattern,* $t \in \mathcal{T}_{\beta\eta}(\mathcal{F},\mathcal{V})$. *If* $t \in \Delta_i(s)$, $i > 0$, *then there are* u, τ *such that* $s \rhd u$, $t \equiv u\tau$ *and* $\tau \in \Delta_{i-1}(s)$. *Furthermore, if* $t \equiv \mathsf{f}(\overline{t_n})$ *then* $u \equiv \mathsf{f}(\overline{u_n})$, $\mathsf{f} \in \mathcal{F}_0 \cup \mathcal{V}_0$.

Proof. Let s be fixed. We perform induction on $\rhd_{ex}$.

1. $s \trianglerighteq_{ex} v \equiv \mathsf{g}(\overline{v_m}) \rhd v_j \equiv t$, $\mathsf{g} \in \mathcal{F}_0 \cup \mathcal{V}_0$.
   By induction hypothesis we have $w \equiv \mathsf{g}(\overline{w_m})$, $s \rhd w$, $\tau' \in \Delta_{i-1}(s)$ and $v \equiv w\tau'$. So $t \equiv v_j \equiv w_j\tau'$.
   (a) If $w_j \equiv x \in \mathcal{X}$ then $x\tau' \equiv u_0 \equiv \mathsf{f}(\overline{u_n}) \in \Delta_{i-1}(s)$. By induction hypothesis there are $u \equiv \mathsf{f}(\overline{u_n})$, $\tau'' \in \Delta_{i-2}(s)$ such that $u_0 \equiv u\tau''$ and $s \rhd u$. We have $t \equiv u\tau$ for $\tau = \tau' \cup \tau$ and $\tau \in \Delta_{i-1}(s)$.
   (b) If $w_j \notin \mathcal{X}$ then $w_j \equiv \mathsf{f}(\overline{u_m})$. Choosing $u \equiv \mathsf{f}(\overline{u_m})$ and $\tau = \tau'$ we have $s \trianglerighteq w \rhd u$ and $t \equiv u\tau$.
2. $s \trianglerighteq_{ex} v \equiv \mathsf{g}(\overline{v_m}) \rhd_1 t$, $v_j \equiv \lambda\overline{x_m}.v_0$, $t \equiv v_0\tau_1$. So $v \in \Delta_{i-1}(s)$. By induction hypothesis we have $s \rhd w \equiv \mathsf{g}(\overline{w_m})$ and $v_0 \equiv w\tau'$, $\tau' \in \Delta_{i-2}(s)$. Choose $u \equiv w$ and $\tau = \tau' \cup \tau_1$. Then $s \rhd w \equiv u$ and $t \equiv v_0\tau_1 \equiv w\tau'\tau_1 \equiv w\tau \equiv u\tau$, so $s \rhd u$ and $t \equiv u\tau$. Furthermore, we have $\tau \in \Delta_{i-1}(s)$. $\square$

Lemma 3 *Let s be a basic pattern,* $t \in \mathcal{T}_{\beta\eta}(\mathcal{F},\mathcal{V})$ *and* σ *a substitution in* $\beta\eta$*-long form. If* $s \rhd_{ex} t$ *then* $s\sigma\!\downarrow \rhd_{ex} t\sigma\!\downarrow$. $\diamond$

Proof. Let s be fixed and $s \trianglerighteq_{ex} t \equiv \mathsf{f}(\overline{t_n})$, $\mathsf{f} \in \mathcal{F}_0 \cup \mathcal{V}_0$ and $t \in \Delta_i(s)$. By Fact 1 we have u and τ such that $s \rhd u$, $t \equiv u\tau$ and $\tau \in \Delta_{i-1}(s)$. We perform induction on $(i, \rhd)$.
Let $s \trianglerighteq_{ex} v \rhd_1 t$ and $v \equiv \mathsf{f}(\overline{v_n})$, $v \in \Delta_i(s)$.

1. $i = 0$: Since s is a pattern, $t \notin \mathcal{X}$, v cannot be a subterm of a flexible term. So $s\sigma\!\downarrow \rhd v\sigma\!\downarrow$ and $\mathsf{f} \in \mathcal{F}_0$. If $t \equiv v_j$ then $v\sigma\!\downarrow \equiv \mathsf{f}(\overline{v_n}\sigma\!\downarrow) \rhd t\sigma\!\downarrow$ and hence $s\sigma\!\downarrow \rhd_{ex} t\sigma\!\downarrow$.
2. $i > 0$: We have $s \rhd w$ and $\tau \in \Delta_{i-1}(s)$ such that $v \equiv w\tau$. If $\mathsf{f} \in \mathcal{F}_0$ and $t \equiv v_j$ then $s\sigma\!\downarrow \trianglerighteq_{ex} v\sigma\!\downarrow \rhd t\sigma\!\downarrow$ as above. Also, if $\mathsf{f} \in \mathcal{F}_0$ and $v_j \equiv \lambda\overline{x_m}.v_0$, $v \equiv v_0\tau_0$, then $s\sigma\!\downarrow \trianglerighteq_{ex} v\sigma\!\downarrow \rhd t\sigma\!\downarrow$ as above. So assume $\mathsf{f} = \mathsf{F} \in \mathcal{V}_0$ and $t \equiv v_j$. Then $w \equiv \mathsf{F}(\overline{w_n})$ and $t \equiv v_j \equiv w_j\tau$. As in the proof of Fact 1, there is u and $\tau' \in \Delta_{i-2}(s)$ such that $t \equiv u\tau'$ and $s \rhd u$. By induction hypothesis we now have $s\sigma\!\downarrow \rhd_{ex} t\sigma\!\downarrow$. $\square$

Example 10. Let $s \equiv \mathsf{f}(\lambda x_1.\mathsf{f}(\lambda x_2.\mathsf{G}(x_1,x_2),x_1),t)$ where $t \equiv \mathsf{h}(\lambda x_3.\mathsf{H}(x_3),\mathsf{a})$. Then $s \rhd_1 t_1 \equiv \mathsf{f}(\lambda x_2.\mathsf{G}(t,x_2),t) \rhd_1 t_2 \equiv \mathsf{G}(t,t) \rhd_1 t_3 \equiv t \rhd_1 t_4 \equiv \mathsf{H}(\mathsf{a})$. We may choose

$$\begin{array}{ll} u_1 \equiv \mathsf{f}(\lambda x_2.\mathsf{G}(x_1,x_2),x_1) & \tau_1 = \{x_1 \leftarrow t\} \\ u_2 \equiv \mathsf{G}(x_1,x_2) & \tau_2 = \{x_1 \leftarrow t,\ x_2 \leftarrow t\} \\ u_3 \equiv \mathsf{h}(\lambda x_3.\mathsf{H}(x_3),\mathsf{a}) & \tau_3 = \emptyset \\ u_4 \equiv \mathsf{H}(x_3) & \tau_4 = \{x_3 \leftarrow \mathsf{a}\} \end{array}$$

The reader may verify that $s \rhd u_i$ and $t_i \equiv u_i\tau_i$, $\tau_i \in \Delta_0(s)$ and $s\sigma\!\downarrow \rhd_{ex} t_i\sigma\!\downarrow$ for $i \in [1,4]$ hold. $\diamond$

Lemma 4 *Let $t \equiv \mathsf{f}(\overline{t_n})$, $\mathsf{f} \in \mathcal{F}$ and t, σ in $\beta\eta$-long form. If $t\sigma\!\downarrow \rhd_1 t'$ then $t \rhd_1 s$ and $s\sigma\!\downarrow \succ_{\mathsf{EMB}} t'$ for some s.* $\diamond$

Proof. Let $u \equiv t\sigma\!\downarrow$, so $u \equiv \mathsf{f}(\overline{u_n})$ and $u_i \equiv t_i\sigma\!\downarrow$.

1. $t' \equiv u_i$ for some i: Choose $s \equiv t_i$. Then $t \rhd_1 s$ and $s\sigma\!\downarrow \equiv t_i\sigma\!\downarrow \equiv u_i \equiv t'$.
2. $u_i \equiv \lambda\overline{x_m}.u_0$, $t' \equiv u_0\tau$ for some i where $\mathcal{D}om(\tau) \subseteq \{\overline{x_m}\}$ and $x\tau \unlhd u_j$ for some $j \neq i$ if $x \in \mathcal{D}om(\tau)$. Then $t \equiv \lambda\overline{x_m}.t_0$ and $u_0 \equiv t_0\sigma\!\downarrow$, so $t' \equiv t_0\tau\!\downarrow$.
   (a) $u_0 \equiv t_0\sigma\!\downarrow$ contains no x_k: Then $t_0\sigma\!\downarrow \equiv t_0\sigma\tau\!\downarrow \equiv u_0\tau \equiv t'$. Choose $s \equiv t_0$. Then $t \rhd_1 s$ and $s\sigma\!\downarrow \equiv t'$
   (b) Assume $x_k\tau' \equiv t_j$, so $x_k\tau'\sigma\!\downarrow \equiv u_j \unrhd x_k\tau$. Choose $s \equiv t_0\tau'$. We have $u_0(\tau' \circ \sigma\!\downarrow) \succ_{\mathsf{EMB}} u_0\tau$. This gives $s\sigma\!\downarrow \equiv (t_0\tau'\sigma)\!\downarrow \equiv u_0(\tau' \circ \sigma\!\downarrow) \succ_{\mathsf{EMB}} u_0\tau \equiv t'$. $\square$

Theorem 4 (Compatibility wrt. Substitutions) *Let s be a basic rigid pattern, $\sigma, s, t \in \mathcal{T}_{\beta\eta}(\mathcal{F},\mathcal{V})$. Then*
1. *$s \sim_{\mathsf{HPO}} t$ implies $s\sigma\!\downarrow \sim_{\mathsf{HPO}} t\sigma\!\downarrow$*
2. *$s \succ_{\mathsf{HPO}} t$ implies $s\sigma\!\downarrow \succ_{\mathsf{HPO}} t\sigma\!\downarrow$* $\diamond$

Proof. Assertion 1. is easily proved by induction on $\rhd$. We prove assertion 2. by induction using $(\succ_{\mathsf{HPO}}, \succ_{\mathsf{HPO}})_{lex}$.

1. $s \succ_{\mathsf{HPO}} t$ according (1) in Definition 8. Then $s \rhd_{ex} s' \sim_{\mathsf{HPO}} t$, so $s\sigma\!\downarrow \rhd_{ex} s'\sigma\!\downarrow$ by Lemma 3 and $s'\sigma\!\downarrow \succ_{\mathsf{HPO}} t\sigma\!\downarrow$ by induction hypothesis. So $s\sigma\!\downarrow \succ_{\mathsf{HPO}} t\sigma\!\downarrow$.
2. $s \succ_{\mathsf{HPO}} t$ according to (2) in Definition 8. Then $\mathsf{f} \succsim_0 \mathsf{g}$, where $s \equiv \mathsf{f}(\overline{s_n})$, $t \equiv \mathsf{g}(\overline{t_m})$ and $s \succ_{\mathsf{HPO}} t'$ whenever $t \rhd_{ex} t'$.
   By induction hypothesis we have $s\sigma\!\downarrow \succ_{\mathsf{HPO}} t'\sigma\!\downarrow$ whenever $t \rhd_1 t'$. Now Lemma 3 gives $s\sigma\!\downarrow \succ_{\mathsf{HPO}} t_0$ whenever $t\sigma\!\downarrow \rhd_1 t_0$.
   If $\mathsf{f} \succ_0 \mathsf{g}$, then $s\sigma\!\downarrow \succ_{\mathsf{HPO}} t\sigma\!\downarrow$ by (2.1) in Definition 8.
   If $\mathsf{f} \sim_0 \mathsf{g}$ then $\{\overline{s_n}\} \overset{\text{ⓜ}}{\succ}_{\mathsf{HPO}} \{\overline{t_m}\}$. So $s_i \succsim_{\mathsf{HPO}} t_{j_i}$ for all i and $s_i \succ_{\mathsf{HPO}} t_{j_i}$ for one $i \in [1,n]$.
   If s_i is flexible, then $s_i\sigma\!\downarrow \succsim_{\mathsf{HPO}} t_{j_i}\sigma\!\downarrow$ by Proposition 1. If s_i is rigid, then $s_i\sigma\!\downarrow \succsim_{\mathsf{HPO}} t_{j_i}\sigma\!\downarrow$ by induction hypothesis.
   So $\{\overline{s_n}\sigma\!\downarrow\} \overset{\text{ⓜ}}{\succ}_{\mathsf{HPO}} \{\overline{t_m}\sigma\!\downarrow\}$ and therefore $s\sigma\!\downarrow \succ_{\mathsf{HPO}} t\sigma\!\downarrow$ by (2.2) in Definition 8. $\square$

Theorem 5 (Termination and Decreasingness) *Let $\mathcal{R}$ be an HCTRS. Then*
1. *$\mathcal{R}$ is terminating if $l \succ_{\mathsf{HPO}} r$ for each rule $l \rightarrow r$ if C in $\mathcal{R}$.*
2. *$\mathcal{R}$ is decreasing if $l \succ_{\mathsf{HPO}} s$ for every s in $\{r\} \cup \{u, v \mid u = v \in C\}$ for each rule $l \rightarrow r$ if C in $\mathcal{R}$.* $\diamond$

Proof. A direct consequence of the properties of the HPO and the restrictions on rule systems which guarantees that $s \succ_{\mathsf{HPO}} t$ whenever $s \rightarrow_R t$ for every s and t in $\beta\eta$-long form. $\square$