A Lambda-Free Higher-Order Recursive Path Order

Jasmin Christian Blanchette[1,2,3], Uwe Waldmann[3], and Daniel Wand[3,4]

[1] Vrije Universiteit Amsterdam, Netherlands
[2] Inria Nancy – Grand Est, Villers-lès-Nancy, France
[3] Max-Planck-Institut für Informatik, Saarbrücken, Germany
[4] Institut für Informatik, Technische Universität München, Germany

Abstract. We generalize the recursive path order (RPO) to higher-order terms without λ-abstraction. This new order fully coincides with the standard RPO on first-order terms also in the presence of currying, distinguishing it from previous work. It has many useful properties, including well-foundedness, transitivity, stability under substitution, and the subterm property. It appears promising as the basis of a higher-order superposition calculus.

1 Introduction

Most automatic reasoning tools are restricted to first-order formalisms, even though many proof assistants and specification languages are higher-order. Translations bridge the gap, but they usually have a cost. Thus, a recurrent question in our field is,

Which first-order methods can be gracefully extended to a higher-order setting?

By "gracefully," we mean that the higher-order extension of the method is as powerful as its first-order counterpart on the first-order portions of the input.

The distinguishing features of higher-order terms are that (1) they support currying, meaning that an n-ary function may be applied to fewer than n arguments, (2) variables can be applied, and (3) λ-abstractions, written $\lambda x.\ t_x$, can be used to specify anonymous functions $x \mapsto t_x$. Iterated applications are written without parentheses or commas, as in $\mathsf{f}\,\mathsf{a}\,\mathsf{b}$. Many first-order proof calculi have been extended to higher-order logic, including resolution [4, 20], tableaux [13], and connections [30], but so far there exists no sound and complete higher-order version of superposition [31], where completeness is considered with respect to Henkin semantics [5, 17]. Together with CDCL(T) [16], superposition is one of the leading proof calculi for classical first-order logic with equality.

To prune the search space, superposition depends on a term order, which is fixed in advance of the proof attempt. For example, from $\mathsf{p}(\mathsf{a})$ and $\neg\, \mathsf{p}(x) \vee \mathsf{p}(\mathsf{f}(x))$, resolution helplessly derives infinitely many clauses of the form $\mathsf{p}(\mathsf{f}^i(\mathsf{a}))$, whereas for superposition the literal $\mathsf{p}(\mathsf{f}(x))$ is maximal in its clause and blocks all inferences. To work with superposition, the order must fulfill many requirements, including compatibility with contexts, stability under substitution, and totality on ground (variable-free) terms. The lexicographic path order (LPO) and the Knuth–Bendix order (KBO) [3] both fulfill the

requirements. LPO is a special case of the recursive path order (RPO), which also subsumes the multiset path order [39]. Suitable generalizations of LPO and KBO appear to be crucial ingredients of a future higher-order superposition prover.

A simple technique to support currying and applied variables is to make all symbols nullary and to represent application by a distinguished binary symbol @. Thus, the higher-order term $\mathsf{f}\,(x\,\mathsf{f})$ is translated to $@(\mathsf{f},@(x,\mathsf{f}))$, which can be processed by first-order methods. We call this the *applicative encoding*. As for λ-abstractions, in many settings they can be avoided using λ-lifting [21] or SK combinators [37]. A drawback of the applicative encoding is that argument tuples cannot be compared using different methods for different function symbols. The use of an application symbol also weakens the order in other ways [26, Sect. 2.3.1]. Hybrid schemes have been proposed to strengthen the encoding: If a function f always occurs with at least k arguments, these can be passed directly in an uncurried style—e.g., $@(\mathsf{f}(\mathsf{a},\mathsf{b}),x)$. However, this relies on a closed-world assumption—namely, that all terms that will ever be compared arise in the input problem. This is at odds with the need for complete higher-order proof calculi to synthesize arbitrary terms during proof search [5], in which a symbol f may be applied to fewer arguments than anywhere in the problem. A scheme by Hirokawa et al. [18] circumvents this issue but requires additional symbols and rewrite rules.

Versions of RPO tailored for higher-order terms are described in the literature, including Lifantsev and Bachmair's LPO on λ-free higher-order terms [28], Jouannaud and Rubio's higher-order RPO (HORPO) [24], Kop and van Raamsdonk's iterative HORPO [27], the HORPO extension with polynomial interpretation orders by Bofill et al. [11], and the computability path order by Blanqui et al. [10], also a variant of HORPO. All of these combine uncurrying and currying: They distinguish between *functional* arguments, which are passed directly as a tuple to a function, and *applicative* arguments, which are optional. Coincidence with the standard RPO on first-order terms is achieved only for uncurried functions. Techniques to automatically curry or uncurry functions have been developed, but they rely on the closed-world assumption. Moreover, the orders all lack totality on ground terms; the HORPO variants also lack the subterm property, and only their (noncomputable) transitive closure is transitive.

We introduce a new "graceful" order $>_{\mathsf{ho}}$ for untyped λ-free higher-order terms (Sect. 3). It generalizes the first-order RPO along two main axes: (1) It relies on a higher-order notion of subterm; (2) it supports terms with applied variables—e.g., $x\,\mathsf{b} >_{\mathsf{ho}} x\,\mathsf{a}$ if $\mathsf{b} \succ \mathsf{a}$ according to the underlying precedence $\succ$ on symbols. The order is parameterized by a family of abstractly specified extension operators indexed by function symbols, allowing lexicographic, multiset, and other extension operators. An optimized variant, $>_{\mathsf{oh}}$, coincides with $>_{\mathsf{ho}}$ under a reasonable assumption on the extension operator. For comparison, we also present the first-order RPO $>_{\mathsf{fo}}$ and its composition $>_{\mathsf{ap}}$ with the applicative encoding, both recast to our abstract framework.

The λ-free fragment is useful in its own right and constitutes a stepping stone towards full higher order. Our new order operates exclusively on curried functions while coinciding with the standard RPO on first-order terms. This was considered impossible by Lifantsev and Bachmair [28]:

> Pairs, or more generally tuples, allow one to compare the arguments of different functions with greater flexibility. For instance, the arguments of one function

> may be compared lexicographically, whereas in other cases comparision may be based on the multisets of arguments. … But since function symbols are much more decoupled from their arguments in a higher-order setting than in a first-order setting, the information needed for different argument-comparision methods would be lost if one, say, just curried all functions.

The order $>_{\mathsf{ho}}$ enjoys many useful properties (Sect. 4). One property that is missing is compatibility with a specific type of higher-order context: If $s' >_{\mathsf{ho}} s$, it is still possible that $s'\ t \not>_{\mathsf{ho}} s\ t$. For example, if $\mathsf{g} \succ \mathsf{f} \succ \mathsf{b} \succ \mathsf{a}$, then $\mathsf{f}\ (\mathsf{g}\ \mathsf{a}) >_{\mathsf{ho}} \mathsf{g}$ by the subterm property, but $\mathsf{f}\ (\mathsf{g}\ \mathsf{a})\ \mathsf{b} <_{\mathsf{ho}} \mathsf{g}\ \mathsf{b}$ by coincidence with the first-order RPO [36]. We can get a counterexample even if we do not assume the subterm property: With the same precedence $\succ$ as above, we have $\mathsf{f}\ (\mathsf{g}\ \mathsf{a})\ \mathsf{a} >_{\mathsf{ho}} \mathsf{g}\ \mathsf{a}$ and $\mathsf{f}\ (\mathsf{g}\ \mathsf{a})\ \mathsf{b} <_{\mathsf{ho}} \mathsf{g}\ \mathsf{b}$. Regardless of how we orient $\mathsf{f}\ (\mathsf{g}\ \mathsf{a})$ and g, we must violate compatibility with higher-order contexts. Nonetheless, we expect the order to be usable for λ-free higher-order superposition, at the cost of some complications [12]. The proofs of the properties were carried out in a proof assistant, Isabelle/HOL [33], and are publicly available [8] (Sect. 5). Detailed informal proofs are included in this report.

Beyond superposition, the order can also be employed to prove termination of higher-order term rewriting systems. Because it treats all functions as curried, it differs from the other higher-order RPOs on many examples (Sect. 6), thereby enriching the portfolio of methods available to termination provers.

Conventions. We fix a set $\mathcal{V}$ of *variables* with typical elements x, y. A higher-order signature consists of a nonempty set Σ of (function) *symbols* $\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{f}, \mathsf{g}, \mathsf{h}, \ldots$. Untyped λ-free higher-order (Σ-)*terms* $s, t, u \in \mathcal{T}_\Sigma\ (= \mathcal{T})$ are defined inductively by the grammar $s ::= x \mid \mathsf{f} \mid t\ u$. These are isomorphic to applicative terms [25].

A term of the form $t\ u$ is called an *application*. Non-application terms $\zeta, \xi \in \Sigma \uplus \mathcal{V}$ are called *heads*. Terms can be decomposed in a unique way as a head applied to zero or more arguments: $\zeta\ s_1 \ldots s_m$. This view corresponds to the first-order, uncurried syntax $\zeta(s_1, \ldots, s_m)$, except that ζ may always be a variable.

The *size* $|s|$ of a term is the number of grammar rule applications needed to construct it. The set of variables occurring in s is written $\mathit{vars}(s)$. The set of *subterms* of a term s always contains s; for applications $t\ u$, it also includes all the subterms of t and u.

A *first-order* signature Σ extends a higher-order signature by associating an *arity* with each symbol belonging to Σ. A *first-order* term is a term in which variables are unapplied and symbols are applied to the number of arguments specified by their arity. For consistency, we will use a curried syntax for first-order terms.

2 Extension Orders

Orders such as RPO depend on extension operators to recurse through tuples of arguments. The literature is mostly concerned with the lexicographic and multiset orders [3, 39]. We favor an abstract treatment that formulates requirements on the extension operators. Beyond its generality, this approach emphasizes the complications arising from the higher-order setting.

Let $A^* = \bigcup_{i=0}^{\infty} A^i$ be the set of tuples (or finite lists) of arbitrary length whose components are drawn from a set A. We write its elements as $(a_1, \ldots, a_m)$, where $m \geq 0$, or simply $\bar{a}$. The empty tuple is written $()$. Singleton tuples are identified with elements of A. The number of components of a tuple $\bar{a}$ is written $|\bar{a}|$. Given an m-tuple $\bar{a}$ and an n-tuple $\bar{b}$, we denote by $\bar{a} \cdot \bar{b}$ the $(m+n)$-tuple consisting of the concatenation of $\bar{a}$ and $\bar{b}$.

Given a function $h : A \to A$, we let $h(\bar{a})$ stand for the componentwise application of h to $\bar{a}$. Abusing notation, we sometimes use a tuple where a set or multiset is expected, ignoring the extraneous structure. Moreover, since all our functions are curried, we write $\zeta\ \bar{s}$ for a curried application $\zeta\ s_1\ \ldots\ s_m$, without risk of ambiguity.

Given a relation $>$, we write $<$ for its inverse (i.e., $a < b \Leftrightarrow b > a$) and $\geq$ for its reflexive closure (i.e., $b \geq a \Leftrightarrow b > a \vee b = a$). A (strict) partial order is a relation that is irreflexive (i.e., $a \not> a$) and transitive (i.e., $c > b \wedge b > a \Longrightarrow c > a$). A (strict) total order is a partial order that satisfies totality (i.e., $b \geq a \vee a > b$). A relation $>$ is well founded if and only if there exists no infinite chain of the form $a_0 > a_1 > \cdots$.

Let $\gg \subseteq (A^*)^2$ be a family of relations indexed by a relation $> \subseteq A^2$. For example, $\gg$ could be the lexicographic or multiset extension of $>$. We assume throughout that if $B \subseteq A$, then the extension $\gg_B$ of the restriction $>_B$ of $>$ to elements from B coincides with $\gg$ on $(B^*)^2$. Moreover, the following properties are essential for all the orders defined later, whether first- or higher-order:

X1. *Monotonicity*: $\bar{b} \gg_1 \bar{a}$ implies $\bar{b} \gg_2 \bar{a}$ if $b >_1 a$ implies $b >_2 a$ for all a, b;
X2. *Preservation of stability*:
$\bar{b} \gg \bar{a}$ implies $h(\bar{b}) \gg h(\bar{a})$ if $b > a$ implies $h(b) > h(a)$ for all a, b;
X3. *Preservation of transitivity*: $\gg$ is transitive if $>$ is transitive;
X4. *Preservation of irreflexivity*: $\gg$ is irreflexive if $>$ is irreflexive and transitive;
X5. *Preservation of well-foundedness*: $\gg$ is well founded if $>$ is well founded;
X6. *Compatibility with tuple contexts*: $b > a$ implies $\bar{c} \cdot b \cdot \bar{d} \gg \bar{c} \cdot a \cdot \bar{d}$.

Because the relation $>$ will depend on $\gg$ for its definition, we cannot assume outright that it is a partial order, a fine point that is sometimes overlooked [39, Sect. 6.4.2].

The remaining properties of $\gg$ will be required only by some of the orders or for some optional properties of $>$:

X7. *Preservation of totality*: $\gg$ is total if $>$ is total;
X8. *Compatibility with prepending*: $\bar{b} \gg \bar{a}$ implies $a \cdot \bar{b} \gg a \cdot \bar{a}$;
X9. *Compatibility with appending*: $\bar{b} \gg \bar{a}$ implies $\bar{b} \cdot a \gg \bar{a} \cdot a$;
X10. *Minimality of empty tuple*: $a \gg ()$.

We now define the extension operators and study their properties. All of them are also defined for tuples of different lengths.

Definition 1. The *lexicographic extension* $\gg^{\mathsf{lex}}$ of the relation $>$ is defined recursively by $() \not\gg^{\mathsf{lex}} \bar{a}$, $b \cdot \bar{b} \gg^{\mathsf{lex}} ()$, and $b \cdot \bar{b} \gg^{\mathsf{lex}} a \cdot \bar{a} \Leftrightarrow b > a \vee b = a \wedge \bar{b} \gg^{\mathsf{lex}} \bar{a}$.

The reverse, or right-to-left, lexicographic extension is defined analogously. Both operators lack the essential property X5. In addition, the left-to-right version lacks X9; a counterexample is $\bar{b} = \mathsf{c}$, $\bar{a} = ()$, and $a = \mathsf{d}$, where $\mathsf{d} > \mathsf{c}$. Correspondingly, the right-to-left version lacks X8. The other properties are straightforward to prove.

Definition 2. The *length-lexicographic extension* $\gg^{\mathsf{llex}}$ of the relation $>$ is defined by $\bar{b} \gg^{\mathsf{llex}} \bar{a} \Leftrightarrow |\bar{b}| > |\bar{a}| \vee |\bar{b}| = |\bar{a}| \wedge \bar{b} \gg^{\mathsf{lex}} \bar{a}$.

The length-lexicographic extension and its right-to-left counterpart satisfy all of the properties listed above. We can also apply arbitrary permutations on same-length tuples before comparing them lexicographically; however, the resulting operators generally fail to satisfy properties X8 and X9.

Definition 3. The *multiset extension* $\gg^{\mathsf{ms}}$ of the relation $>$ is defined by $\bar{b} \gg^{\mathsf{ms}} \bar{a} \Leftrightarrow \exists Y, X.\ \emptyset \neq Y \subseteq \bar{b} \wedge \bar{a} = (\bar{b} - Y) \uplus X \wedge \forall x \in X.\ \exists y \in Y.\ y > x$, where X, Y range over multisets, the tuples $\bar{a}, \bar{b}$ are implicitly converted to multisets, and $\uplus$ denotes multiset sum (the sum of the multiplicity functions).

The multiset extension, due to Dershowitz and Manna [15], satisfies all properties except X7. Huet and Oppen [19] give an alternative formulation that is equivalent for partial orders $>$ but exhibits subtle differences if $>$ is an arbitrary relation. In particular, the Huet–Oppen order does not satisfy property X3, making it unsuitable for establishing that RPO variants are partial orders.

Finally, we consider the componentwise extension of relations to pairs of tuples of the same length. For partial orders $>$, this order underapproximates any extension that satisfies properties X3 and X6. It also satisfies all properties except X7.

Definition 4. The *componentwise extension* $\gg^{\mathsf{cw}}$ of the relation $>$ is defined so that $(b_1, \ldots, b_n) \gg^{\mathsf{cw}} (a_1, \ldots, a_m)$ if and only if $m = n$, $b_1 \geq a_1$, …, $b_m \geq a_m$, and $b_i > a_i$ for some $i \in \{1, \ldots, m\}$.

3 Term Orders

This section presents four orders: the standard first-order RPO (Sect. 3.1), the applicative RPO (Sect. 3.2), our new λ-free higher-order RPO (Sect. 3.3), and an optimized variant of our new RPO (Sect. 3.4).

3.1 The Standard First-Order RPO

The following definition is close to Zantema's formulation [39, Definition 6.4.4] but adapted to our setting. With three rules instead of four, it is more concise than Baader and Nipkow's formulation of LPO [3, Definition 5.4.12] and lends itself better to a higher-order generalization.

Definition 5. Let $\succ$ be a well-founded total order on Σ, and let $\gg^{\mathsf{f}} \subseteq (\mathcal{T}^*)^2$ be a family of relations indexed by $> \subseteq \mathcal{T}^2$ and by $\mathsf{f} \in \Sigma$ and satisfying properties X1–X6. The induced *recursive path order* $>_{\mathsf{fo}}$ on first-order Σ-terms is defined inductively so that $t >_{\mathsf{fo}} s$ if any of the following conditions is met, where $t = \mathsf{g}\ \bar{t}$:

F1. $t' \geq_{\mathsf{fo}} s$ for some term $t' \in \bar{t}$;
F2. $s = \mathsf{f}\ \bar{s}$, $\mathsf{g} \succ \mathsf{f}$, and $\mathit{chkargs}(t, \bar{s})$;
F3. $s = \mathsf{f}\ \bar{s}$, $\mathsf{f} = \mathsf{g}$, $\bar{t} \gg^{\mathsf{f}}_{\mathsf{fo}} \bar{s}$, and $\mathit{chkargs}(t, \bar{s})$.

The auxiliary predicate $\mathit{chkargs}(t, \bar{s})$ is true if and only if $t >_{\mathsf{fo}} s'$ for all terms $s' \in \bar{s}$. The inductive definition is legitimate by the Knaster–Tarski theorem owing to the monotonicity of $\gg^{\mathsf{f}}$ (property X1).

RPO is a compromise between two design goals. On the one hand, rules F2 and F3, which form the core of the order, attempt to perform a comparison of two terms by first looking at their heads, proceeding recursively to break ties. On the other hand, rule F1 ensures that terms are larger than their proper subterms and, transitively, larger than terms smaller than these. The *chkargs* predicate prevents the application of F2 and F3 when F1 is applicable in the other direction, ensuring irreflexivity.

The more recent literature defines RPO somewhat differently: Precision is improved by replacing recursive calls to $\geq_{\mathsf{fo}}$ with a nonstrict quasiorder $\gtrsim_{\mathsf{fo}}$ and by exploiting a generalized multiset extension [14,35]. These extensions are useful but require substantial duplication in the definitions and the proofs, without yielding much new insight into orders for higher-order terms.

3.2 The Applicative RPO

Applicative orders are built by encoding applications using a binary symbol @ and by employing a first-order term order. For RPO, the precedence $\succ$ must be extended to consider @. A natural choice is to make @ the least element of $\succ$. Because @ is the only symbol that may be applied, $\gg^{@}$ is the only member of the $\gg$ family that is relevant. This means that it is impossible to use the lexicographic extension for some functions and the multiset extension for others.

Definition 6. Let Σ be a higher-order signature, and let $\Sigma' = \Sigma \uplus \{@\}$ be a first-order signature in which all symbols belonging to Σ are assigned arity 0 and @ is assigned arity 2. The *applicative encoding* $[\![\ ]\!] : \mathcal{T}_\Sigma \to \mathcal{T}_{\Sigma'}$ is defined recursively by the equations $[\![\zeta]\!] = \zeta$ and $[\![s\ t]\!] = @\ [\![s]\!]\ [\![t]\!]$.

Assuming that @ has the lowest precedence, the composition of the first-order RPO with the encoding $[\![\ ]\!]$ can be formulated directly as follows.

Definition 7. Let $\succ$ be a well-founded total order on Σ, and let $\gg \subseteq (\mathcal{T}^*)^2$ be a family of relations indexed by $> \subseteq \mathcal{T}^2$ and satisfying properties X1–X6. The induced *applicative recursive path order* $>_{\mathsf{ap}}$ on higher-order Σ-terms is defined inductively so that $t >_{\mathsf{ap}} s$ if any of the following conditions is met:

A1. $t = t_1\ t_2$ and either $t_1 \geq_{\mathsf{ap}} s$ or $t_2 \geq_{\mathsf{ap}} s$ (or both);
A2. $t = \mathsf{g} \succ \mathsf{f} = s$;
A3. $t = \mathsf{g}$, $s = s_1\ s_2$, and $\mathit{chkargs}(t, s_1, s_2)$;
A4. $t = t_1\ t_2$, $s = s_1\ s_2$, $(t_1, t_2) \gg_{\mathsf{ap}} (s_1, s_2)$, and $\mathit{chkargs}(t, s_1, s_2)$.

The predicate $\mathit{chkargs}(t, s_1, s_2)$ is true if and only if $t >_{\mathsf{ap}} s_1$ and $t >_{\mathsf{ap}} s_2$.

3.3 A Graceful Higher-Order RPO

Our new "graceful" higher-order RPO is much closer to the first-order RPO than the applicative RPO. It reintroduces the symbol-indexed family of extension operators and consists of three rules H1–H3 corresponding to F1–F3.

The order relies on a mapping ghd from variables to nonempty sets of possible ground heads that may arise when instantiating the variables. This mapping is extended to symbols f by taking $\mathit{ghd}(\mathsf{f}) = \{\mathsf{f}\}$. A substitution $\sigma : \mathcal{V} \to \mathcal{T}$ is said to *respect* the ghd mapping if for all variables x, we have $\mathit{ghd}(\zeta) \subseteq \mathit{ghd}(x)$ whenever $x\sigma = \zeta\ \bar{s}$. This mapping allows us to restrict instantiations, typically based on a typing discipline, and thereby increase the applicability of rules H2 and especially H3. Precedences $\succ$ are extended to variables by taking $y \succ x \Leftrightarrow \forall \mathsf{g} \in \mathit{ghd}(y), \mathsf{f} \in \mathit{ghd}(x).\ \mathsf{g} \succ \mathsf{f}$.

Definition 8. Let $\succ$ be a well-founded total order on Σ, let $\gg^{\mathsf{f}} \subseteq (\mathcal{T}^*)^2$ be a family of relations indexed by $> \subseteq \mathcal{T}^2$ and by $\mathsf{f} \in \Sigma$ and satisfying properties X1–X6 and X8, and let $\mathit{ghd} : \mathcal{V} \to \mathcal{P}(\Sigma) - \{\emptyset\}$. The induced *graceful recursive path order* $>_{\mathsf{ho}}$ on higher-order Σ-terms is defined inductively so that $t >_{\mathsf{ho}} s$ if any of the following conditions is met, where $s = \zeta\ \bar{s}$ and $t = \xi\ \bar{t}$:

H1. $t = t_1\ t_2$ and either $t_1 \geq_{\mathsf{ho}} s$ or $t_2 \geq_{\mathsf{ho}} s$ (or both);
H2. $\xi \succ \zeta$, $\mathit{vars}(t) \supseteq \mathit{vars}(\zeta)$, and $\mathit{chksubs}(t, s)$;
H3. $\xi = \zeta$, $\bar{t} \gg^{\mathsf{f}}_{\mathsf{ho}} \bar{s}$ for all symbols $\mathsf{f} \in \mathit{ghd}(\zeta)$, and $\mathit{chksubs}(t, s)$.

The predicate $\mathit{chksubs}(t, s)$ is true if and only if term s is a head or an application of the form $s_1\ s_2$ with $t >_{\mathsf{ho}} s_1$ and $t >_{\mathsf{ho}} s_2$.

There are two main novelties compared with $>_{\mathsf{fo}}$. First, rule H1 and the $\mathit{chksubs}$ predicate traverse subterms in a genuinely higher-order fashion. Second, rules H2 and H3 can compare terms with variable heads.

Property X8, compatibility with prepending, is necessary to ensure stability under substitution: If $x\ \mathsf{b} >_{\mathsf{ho}} x\ \mathsf{a}$, we want $\mathsf{f}\ \bar{s}\ \mathsf{b} >_{\mathsf{ho}} \mathsf{f}\ \bar{s}\ \mathsf{a}$ to hold as well.

Example 9. It is instructive to contrast our new order with the applicative order by studying a few small examples. Let $\mathsf{h} \succ \mathsf{g} \succ \mathsf{f} \succ \mathsf{b} \succ \mathsf{a}$, let $\gg$ be the length-lexicographic extension (which degenerates to the plain lexicographic extension for $>_{\mathsf{ap}}$), and let $\mathit{ghd}(x) = \Sigma$ for all variables x. Section 1 already presented a case where $>_{\mathsf{ho}}$ and $>_{\mathsf{ap}}$ disagree: $\mathsf{g}\ \mathsf{b} >_{\mathsf{ho}} \mathsf{f}\ (\mathsf{g}\ \mathsf{a})\ \mathsf{b}$ but $\mathsf{g}\ \mathsf{b} <_{\mathsf{ap}} \mathsf{f}\ (\mathsf{g}\ \mathsf{a})\ \mathsf{b}$. Other disagreements include

$$\mathsf{g}\ \mathsf{f} >_{\mathsf{ho}} \mathsf{f}\ \mathsf{g}\ \mathsf{f} \qquad \mathsf{g}\ \mathsf{f} >_{\mathsf{ho}} \mathsf{f}\ \mathsf{g}\ (\mathsf{f}\ \mathsf{g}) \qquad \mathsf{g}\ \mathsf{g} >_{\mathsf{ho}} \mathsf{f}\ \mathsf{g}\ \mathsf{g} \qquad \mathsf{g}\ (\mathsf{f}\ \mathsf{h}) >_{\mathsf{ho}} \mathsf{f}\ \mathsf{h}\ (\mathsf{f}\ \mathsf{h})$$

and $\mathsf{g}\ \mathsf{g}\ \mathsf{g}\ (\mathsf{f}\ (\mathsf{g}\ (\mathsf{g}\ \mathsf{g}\ \mathsf{g}))) >_{\mathsf{ho}} \mathsf{g}\ (\mathsf{g}\ \mathsf{g}\ \mathsf{g})\ (\mathsf{g}\ \mathsf{g}\ \mathsf{g})$. For all of these, the core rules H2 and H3 are given room for maneuver, whereas $>_{\mathsf{ap}}$ must consider subterms using A1. In the presence of variables, some terms are comparable only with $>_{\mathsf{ho}}$ or only with $>_{\mathsf{ap}}$:

$$\mathsf{g}\ x >_{\mathsf{ho}} \mathsf{f}\ x\ x \qquad \mathsf{g}\ x >_{\mathsf{ho}} \mathsf{f}\ x\ \mathsf{g} \qquad \mathsf{f}\ x\ y >_{\mathsf{ap}} x\ y \qquad x\ \mathsf{f}\ (x\ \mathsf{f}) >_{\mathsf{ap}} \mathsf{f}\ x$$

To apply rule A4 on the first example, we would need $(\mathsf{g}, x) \gg^{\mathsf{lex}}_{\mathsf{ap}} (\mathsf{f}\ x, x)$, but the term g cannot be larger than $\mathsf{f}\ x$ since it does not contain x. The last two examples reveal that the applicative order tends to be stronger when either side is a variable applied to some arguments—at least when ghd is not restricting the variable instantiations.

3.4 An Optimized Variant of the Graceful Higher-Order RPO

The higher-order term $\mathsf{f}\,\mathsf{a}\,\mathsf{b}$ has four proper subterms: a, b, f, and $\mathsf{f}\,\mathsf{a}$. In contrast, the corresponding first-order term, traditionally written $\mathsf{f}(\mathsf{a},\mathsf{b})$, has only the arguments a and b as proper subterms. In general, a term of size k has up to $k-1$ distinct proper subterms in a higher-order sense but only half as many in a first-order sense. By adding a reasonable requirement on the extension operator, we can avoid this factor-of-2 penalty when computing the order.

Definition 10. Let $\succ$ be a well-founded total order on Σ, let $\gg^{\mathsf{f}} \subseteq (\mathcal{T}^*)^2$ be a family of relations indexed by $> \subseteq \mathcal{T}^2$ and by $\mathsf{f} \in \Sigma$ and satisfying properties X1–X6, X8, and X10, and let $\mathit{ghd} : \mathcal{V} \to \mathcal{P}(\Sigma) - \{\emptyset\}$. The induced *optimized graceful recursive path order* $>_{\mathsf{oh}}$ on higher-order Σ-terms is defined inductively so that $t >_{\mathsf{oh}} s$ if any of the following conditions is met, where $s = \zeta\ \bar{s}$ and $t = \xi\ \bar{t}$:

O1. $t' \geq_{\mathsf{oh}} s$ for some term $t' \in \bar{t}$;
O2. $\xi \succ \zeta$, $\mathit{vars}(t) \supseteq \mathit{vars}(\zeta)$, and $\mathit{chkargs}(t, \bar{s})$;
O3. $\xi = \zeta$, $\bar{t} \gg^{\mathsf{f}}_{\mathsf{oh}} \bar{s}$ for all symbols $\mathsf{f} \in \mathit{ghd}(\zeta)$, and $\mathit{chkargs}(t, \bar{s})$.

The predicate $\mathit{chkargs}(t, \bar{s})$ is true if and only if $t >_{\mathsf{oh}} s'$ for all terms $s' \in \bar{s}$.

The optimized $>_{\mathsf{oh}}$ depends on the same parameters as $>_{\mathsf{ho}}$ except that it additionally requires minimality of the empty tuple (property X10). In conjunction with compatibility with prepending (X8), this property ensures that $\bar{a} \cdot a \gg^{\mathsf{f}} \bar{a}$. As a result, $\mathsf{f}\ \bar{s}\ s$ is greater than its subterm $\mathsf{f}\ \bar{s}$, relieving rule O1 from having to consider such subterms.

Syntactically, the definition of $>_{\mathsf{oh}}$ generalizes that of the first-order $>_{\mathsf{fo}}$. Semantically, the restriction of $>_{\mathsf{oh}}$ to first-order terms coincides with $>_{\mathsf{fo}}$. The requirements X8 and X10 on $\gg^{\mathsf{f}}$ can be made without loss of generality in a first-order setting.

The quantification over $\mathsf{f} \in \mathit{ghd}(\zeta)$ in rule O3 can be inefficient in an implementation, when different symbols in $\mathit{ghd}(\zeta)$ disagree on which $\gg$ to use. We could generalize the definition of $>_{\mathsf{oh}}$ further to allow underapproximation, but some care would be needed to ensure transitivity. A simple alternative is to enrich all sets $\mathit{ghd}(\zeta)$ that disagree on $\gg$ with a distinguished symbol for which the componentwise extension is used. Since this extension operator is more restrictive than any other ones, whenever it is present in a set $\mathit{ghd}(\zeta)$ there is no need to compute the other ones.

4 Properties

We now state and prove the main properties of our RPO. We focus on the general variant $>_{\mathsf{ho}}$ and show that it is equivalent to the optimized variant $>_{\mathsf{oh}}$ (assuming property X10). Many of the proofs are adapted from Baader and Nipkow [3] and Zantema [39].

Lemma 11. *If $t >_{\mathsf{ho}} s$, then $\mathit{vars}(t) \supseteq \mathit{vars}(s)$.*

Proof. By strong induction on $|s| + |t|$. If $t >_{\mathsf{ho}} s$ was derived by rule H1, the desired result follows immediately from the induction hypothesis. If $t >_{\mathsf{ho}} s$ was derived by rule H2, we distinguish two cases. If $s = \zeta$, the result follows from the condition

$\mathit{vars}(t) \supseteq \mathit{vars}(\zeta)$ on H2. Otherwise, $s = s_1\ s_2$; by $\mathit{chksubs}(t,s)$ and the induction hypothesis, $\mathit{vars}(t) \supseteq \mathit{vars}(s_1)$ and $\mathit{vars}(t) \supseteq \mathit{vars}(s_2)$, yielding the desired result. If $t >_{\mathsf{ho}} s$ was derived by rule H3, we distinguish two cases. If $s = \zeta$, then t must also have ζ as its head. The $s = s_1\ s_2$ case is as for rule H2.

As a consequence of Lemma 11, the condition $\mathit{vars}(t) \supseteq \mathit{vars}(\zeta)$ of rule H2 could be written equivalently (but less efficiently) as $\mathit{vars}(t) \supseteq \mathit{vars}(s)$.

Theorem 12 (Transitivity). *If $u >_{\mathsf{ho}} t$ and $t >_{\mathsf{ho}} s$, then $u >_{\mathsf{ho}} s$.*

Proof. By well-founded induction on the multiset $\{|s|, |t|, |u|\}$ with respect to the multiset extension of $>$ on $\mathbb{N}$.

If $u >_{\mathsf{ho}} t$ was derived by rule H1, we have $u = u_1\ u_2$ and $u_k \geq_{\mathsf{ho}} t$ for some k. Since $t >_{\mathsf{ho}} s$ by hypothesis, $u_k >_{\mathsf{ho}} s$ follows either immediately (if $u_k = t$) or by the induction hypothesis (if $u_k >_{\mathsf{ho}} t$). We get $u >_{\mathsf{ho}} s$ by rule H1.

Otherwise, $u >_{\mathsf{ho}} t$ was derived by rule H2 or H3. The $\mathit{chksubs}$ condition ensures that u is greater than any immediate subterms of t. We proceed by case analysis on the rule that derived $t >_{\mathsf{ho}} s$.

If $t >_{\mathsf{ho}} s$ was derived by H1, we have $t = t_1\ t_2$ and $t_j \geq_{\mathsf{ho}} s$ for some j. We already noted that $u >_{\mathsf{ho}} t_j$ thanks to $\mathit{chksubs}(u,t)$. In conjunction with $t_j \geq_{\mathsf{ho}} s$, we derive $u >_{\mathsf{ho}} s$ either immediately or by the induction hypothesis.

Otherwise, $t >_{\mathsf{ho}} s$ was derived by rule H2 or H3. The $\mathit{chksubs}$ condition ensures that t is greater than any immediate subterms of s. We derive $u >_{\mathsf{ho}} s$ by applying H2 or H3. We first prove $\mathit{chksubs}(u,s)$. The only nontrivial case is $s = s_1\ s_2$. Using $u >_{\mathsf{ho}} t$, we get $u >_{\mathsf{ho}} s_1$ and $u >_{\mathsf{ho}} s_2$ by the induction hypothesis.

If both $u >_{\mathsf{ho}} t$ and $t >_{\mathsf{ho}} s$ were derived by rule H3, we apply H3 to derive $u >_{\mathsf{ho}} s$. This relies on the preservation by $\gg^{\mathsf{f}}_{\mathsf{ho}}$ of transitivity (property X3) on the set consisting of the argument tuples of s,t,u. Transitivity of $>_{\mathsf{ho}}$ on these tuples follows from the induction hypothesis. Finally, if either $u >_{\mathsf{ho}} t$ or $t >_{\mathsf{ho}} s$ was derived by rule H2, we apply H2, relying on the transitivity of $\succ$ and on Lemma 11. □

Theorem 13 (Irreflexivity). *$s \not>_{\mathsf{ho}} s$.*

Proof. By strong induction on $|s|$. We assume $s >_{\mathsf{ho}} s$ and show that this leads to a contradiction. If $s >_{\mathsf{ho}} s$ was derived by rule H1, we have $s = s_1\ s_2$ with $s_i \geq_{\mathsf{ho}} s$ for some i. Since a term cannot be equal to one of its proper subterms, the comparison is strict. Moreover, we have $s >_{\mathsf{ho}} s_i$ by rule H1. Transitivity yields $s_i >_{\mathsf{ho}} s_i$, contradicting the induction hypothesis. If $s >_{\mathsf{ho}} s$ was derived by rule H2, the contradiction follows immediately from the irreflexivity of $\succ$. Otherwise, $s >_{\mathsf{ho}} s$ was derived by rule H3. Let $s = \zeta\ \bar{s}$. We have $\bar{s} \gg^{\mathsf{f}}_{\mathsf{ho}} \bar{s}$ for all $\mathsf{f} \in \mathit{ghd}(\zeta) \neq \emptyset$. Since $\gg^{\mathsf{f}}$ preserves irreflexivity for transitive relations (property X4) and $>_{\mathsf{ho}}$ is transitive (Theorem 12), there must exist a term $s' \in \bar{s}$ such that $s' >_{\mathsf{ho}} s'$. However, this contradicts the induction hypothesis. □

By Theorems 12 and 13, $>_{\mathsf{ho}}$ is a partial order. In the remaining proofs, we will often leave applications of these theorems (and of antisymmetry) implicit.

Theorem 14 (Subterm Property). *If s is a proper subterm of t, then $t >_{\mathsf{ho}} s$.*

Proof. By structural induction on t, exploiting rule H1 and transitivity of $>_{\mathsf{ho}}$. □

The first-order RPO satisfies compatibility with Σ-operations. A slightly more general property holds for $>_{\mathsf{ho}}$:

Theorem 15 (Compatibility with Functions). *If $t' >_{\mathsf{ho}} t$, then $s\ t'\ \bar{u} >_{\mathsf{ho}} s\ t\ \bar{u}$.*

Proof. By induction on the length of $\bar{u}$. The base case, $\bar{u} = ()$, follows from rule H3, compatibility of $\gg^{\mathsf{f}}$ with tuple contexts (property X6), and the subterm property (Theorem 14). The step case, $\bar{u} = \bar{u}' \cdot u$, also follows from rule H3 and compatibility of $\gg^{\mathsf{f}}$ with contexts. The $\mathit{chksubs}(s\ t'\ \bar{u}'\ u, s\ t\ \bar{u}'\ u)$ condition follows from the induction hypothesis and the subterm property. □

A related property, compatibility with arguments, is useful to rewrite subterms such as $\mathsf{f}\,\mathsf{a}$ in $\mathsf{f}\,\mathsf{a}\,\mathsf{b}$ using a rewrite rule $\mathsf{f}\,x \to t_x$. Unfortunately, $>_{\mathsf{ho}}$ does not enjoy this property: $s' >_{\mathsf{ho}} s$ does not imply $s'\ t >_{\mathsf{ho}} s\ t$. Two counterexamples follow:

1. Given $\mathsf{g} \succ \mathsf{f}$, we have $\mathsf{f}\,\mathsf{g} >_{\mathsf{ho}} \mathsf{g}$ by rule H1, but $\mathsf{f}\,\mathsf{g}\,\mathsf{f} <_{\mathsf{ho}} \mathsf{g}\,\mathsf{f}$ by rule H2.
2. Let $\mathsf{f} \succ \mathsf{b} \succ \mathsf{a}$, and let $\gg^{\mathsf{f}}$ be the lexicographic extension. Then $\mathsf{f}\,\mathsf{a} >_{\mathsf{ho}} \mathsf{f}$ by rule H3, but $\mathsf{f}\,\mathsf{a}\,\mathsf{b} <_{\mathsf{ho}} \mathsf{f}\,\mathsf{b}$ also by rule H3.

The second counterexample and similar ones involving rule H3 can be excluded by requiring that $\gg^{\mathsf{f}}$ is compatible with appending (property X9), which holds for the length-lexicographic and multiset extensions. But there is no way to rule out the first counterexample without losing coincidence with the first-order RPO.

Theorem 16 (Compatibility with Arguments). *Assume that $\gg^{\mathsf{f}}$ is compatible with appending (property X9) for every symbol $\mathsf{f} \in \Sigma$. If $s' >_{\mathsf{ho}} s$ is derivable by rule H2 or H3, then $s'\ t >_{\mathsf{ho}} s\ t$.*

Proof. If $s' >_{\mathsf{ho}} s$ is derivable by rule H2, we apply H2 to derive $s'\ t >_{\mathsf{ho}} s\ t$. To show $\mathit{chksubs}(s'\ t, s\ t)$, we must show that $s'\ t >_{\mathsf{ho}} s$ and $s'\ t >_{\mathsf{ho}} t$. Both are consequences of the subterm property (Theorem 14), together with $s' >_{\mathsf{ho}} s$.

If $s' >_{\mathsf{ho}} s$ is derivable by rule H3, we apply H3 to derive $s'\ t >_{\mathsf{ho}} s\ t$. The condition on the variables of the head of $s'\ t$ can be shown by exploiting the condition on the variables of the head of s'. The *chksubs* condition is shown as above. The condition on the argument tuples follows by property X9. □

Theorem 17 (Stability under Substitution). *If $t >_{\mathsf{ho}} s$, then $t\sigma >_{\mathsf{ho}} s\sigma$ for any substitution σ that respects the mapping ghd.*

Proof. By well-founded induction on the multiset $\{|s|, |t|\}$ with respect to the multiset extension of $>$ on $\mathbb{N}$.

If $t >_{\mathsf{ho}} s$ was derived by rule H1, we have $t = t_1\ t_2$ and $t_j \geq_{\mathsf{ho}} s$ for some j. By the induction hypothesis, $t_j\sigma \geq_{\mathsf{ho}} s\sigma$. Hence, $t\sigma >_{\mathsf{ho}} s\sigma$ by rule H1.

If $t >_{\mathsf{ho}} s$ was derived by rule H2, we have $s = \zeta\ \bar{s}$, $t = \xi\ \bar{t}$, $\xi \succ \zeta$, and $\mathit{chksubs}(t, s)$. We derive $t\sigma >_{\mathsf{ho}} s\sigma$ by applying H2. Since σ respects ghd, we have $\xi\sigma \succ \zeta\sigma$. From $t >_{\mathsf{ho}} s$, we have $\mathit{vars}(t) \supseteq \mathit{vars}(s)$ by Lemma 11 and hence $\mathit{vars}(t\sigma) \supseteq \mathit{vars}(s\sigma) \supseteq \mathit{vars}(\xi\sigma)$. To show $\mathit{chksubs}(t\sigma, s\sigma)$, the nontrivial cases are $s = x$ and is $s = s_1\ s_2$. If $s = x$, then

s must be a subterm of t by Lemma 11, and therefore $s\sigma$ is a subterm of $t\sigma$. Thus, we have $t\sigma >_{\mathsf{ho}} s\sigma$ by the subterm property (Theorem 14), from which it is easy to derive $\mathit{chksubs}(t\sigma, s\sigma)$, as desired. If $s = s_1\ s_2$, we get $t >_{\mathsf{ho}} s_1$ and $t >_{\mathsf{ho}} s_2$ from $\mathit{chksubs}(t,s)$. By the induction hypothesis, $t\sigma >_{\mathsf{ho}} s_1\sigma$ and $t >_{\mathsf{ho}} s_2\sigma$, as desired.

If $t >_{\mathsf{ho}} s$ was derived by rule H3, we have $s = \zeta\ \bar{s}$, $t = \zeta\ \bar{t}$, $\bar{t} \gg^{\mathsf{f}}_{\mathsf{ho}} \bar{s}$ for all $\mathsf{f} \in \mathit{ghd}(\zeta)$, and $\mathit{chksubs}(t,s)$. We derive $t\sigma >_{\mathsf{ho}} s\sigma$ by applying H3. Clearly, $s\sigma$ and $t\sigma$ have the same head. The $\mathit{chksubs}(t\sigma, s\sigma)$ condition is proved as for rule H2 above. Finally, we must show that $\bar{t}\sigma \gg^{\mathsf{f}}_{\mathsf{ho}} \bar{s}\sigma$ for all $\mathsf{f} \in \mathit{ghd}(\zeta')$, where $\zeta\sigma = \zeta'\ \bar{u}$ for some $\bar{u}$. Since σ respects ghd, we have $\mathit{ghd}(\zeta') \subseteq \mathit{ghd}(\zeta)$; hence, $\bar{t} \gg^{\mathsf{f}}_{\mathsf{ho}} \bar{s}$ for all $\mathsf{f} \in \mathit{ghd}(\zeta')$. By the induction hypothesis, $t' >_{\mathsf{ho}} s'$ implies $t'\sigma >_{\mathsf{ho}} s'\sigma$ for all $s', t' \in \bar{s} \cup \bar{t}$. By preservation of stability (property X2), we have $\bar{t}\sigma \gg^{\mathsf{f}}_{\mathsf{ho}} \bar{s}\sigma$. By compatibility with prepending (property X8), we get $\bar{u} \cdot \bar{t}\sigma \gg^{\mathsf{f}}_{\mathsf{ho}} \bar{u} \cdot \bar{s}\sigma$, as required to apply H3. □

Theorem 18 (Well-foundedness). *There exists no infinite descending chain* $s_0 >_{\mathsf{ho}} s_1 >_{\mathsf{ho}} \cdots$.

Proof. We assume that there exists a chain $s_0 >_{\mathsf{ho}} s_1 >_{\mathsf{ho}} \cdots$ and show that this leads to a contradiction. If the chain contains nonground terms, we can instantiate all variables by arbitrary terms respecting ghd and exploit stability under substitution (Theorem 17). Thus, we may assume without loss of generality that the terms $s_0, s_1, \ldots$ are ground.

We call a ground term *bad* if it belongs to an infinite descending $>_{\mathsf{ho}}$-chain. Without loss of generality, we assume that s_0 has minimal size among all bad terms and that s_{i+1} has minimal size among all bad terms t such that $s_i >_{\mathsf{ho}} t$.

For each index i, the term s_i must be of the form $\mathsf{f}\ u_1 \ldots u_n$ for some symbol f and ground terms $u_1, \ldots, u_n$. Let

$$U_i = \begin{cases} \emptyset & \text{if } n = 0 \\ \{u_1, \ldots, u_n, \mathsf{f}\ u_1 \cdots u_{n-1}\} & \text{otherwise} \end{cases}$$

Now let $U = \bigcup_{i=0}^{\infty} U_i$. All terms belonging to U are good: A term from U_0's badness would contradict the minimality of s_0; and if a term $u \in U_{i+1}$ were bad, we would have $s_{i+1} >_{\mathsf{ho}} u$ by rule H1 and $s_i >_{\mathsf{ho}} u$ by transitivity, contradicting the minimality of s_{i+1}.

Next, we show that the only rules that can be used to derive $s_i >_{\mathsf{ho}} s_{i+1}$ are H2 and H3. Suppose H1 were used. Then there would exist a good term $u \in U_i$ such that $u \geq_{\mathsf{ho}} s_{i+1} >_{\mathsf{ho}} s_{i+2}$. This would imply the existence of an infinite chain $u >_{\mathsf{ho}} s_{i+2} >_{\mathsf{ho}} s_{i+3} >_{\mathsf{ho}} \cdots$, contradicting the goodness of u.

Because $\succ$ is well founded and H3 preserves the head symbol, rule H2 can be applied only a finite number of times in the chain. Hence, there must exist an index k such that $s_i >_{\mathsf{ho}} s_{i+1}$ is derived using H3 for all $i \geq k$. Consequently, all terms s_i for $i \geq k$ share the same head symbol f.

Let $s_i = \mathsf{f}\ \bar{u}_i$ for all $i \geq k$. Since H3 is used consistently from index k, we have an infinite $\gg^{\mathsf{f}}_{\mathsf{ho}}$-chain: $\bar{u}_k \gg^{\mathsf{f}}_{\mathsf{ho}} \bar{u}_{k+1} \gg^{\mathsf{f}}_{\mathsf{ho}} \bar{u}_{k+2} \gg^{\mathsf{f}}_{\mathsf{ho}} \cdots$. But since U contains only good terms and comprises all terms occurring in some argument tuple $\bar{u}_i$, $>_{\mathsf{ho}}$ is well founded on U. By preservation of well-foundedness (property X5), $\gg^{\mathsf{f}}_{\mathsf{ho}}$ is well founded. This contradicts the existence of the above $\gg^{\mathsf{f}}_{\mathsf{ho}}$-chain. □

Theorem 19 (Ground Totality). *Assume* $\gg^{\mathsf{f}}$ *preserves totality (property X7) for every symbol* $\mathsf{f} \in \Sigma$*, and let* s,t *be ground terms. Then either* $t \geq_{\mathsf{ho}} s$ *or* $t <_{\mathsf{ho}} s$.

Proof. By strong induction on $|s| + |t|$. If not $\mathit{chksubs}(t, s)$, then $t \not>_{\mathsf{ho}} s_1$ and $t \not>_{\mathsf{ho}} s_2$ for $s = s_1\ s_2$. By the induction hypothesis, $s_1 \geq_{\mathsf{ho}} t$ and $s_2 \geq_{\mathsf{ho}} t$. Thus, $s >_{\mathsf{ho}} t$ by rule H1. Analogously, if not $\mathit{chksubs}(s,t)$, then $t >_{\mathsf{ho}} s$. Hence, we may assume $\mathit{chksubs}(t, s)$ and $\mathit{chksubs}(s,t)$. Let $s = \mathsf{f}\ \bar{s}$ and $t = \mathsf{g}\ \bar{t}$. If $\mathsf{g} \succ \mathsf{f}$ or $\mathsf{g} \prec \mathsf{f}$, we have $t >_{\mathsf{ho}} s$ or $s >_{\mathsf{ho}} t$ by rule H2. Otherwise, $\mathsf{f} = \mathsf{g}$. By preservation of totality (property X7), we have either $\bar{t} \gg^{\mathsf{f}}_{\mathsf{ho}} \bar{s}$, $\bar{t} \ll^{\mathsf{f}}_{\mathsf{ho}} \bar{s}$, or $\bar{s} = \bar{t}$. In the first two cases, we have $t >_{\mathsf{ho}} s$ or $t <_{\mathsf{ho}} s$ by rule H3. In the third case, we have $s = t$. □

Having now established the main properties of $>_{\mathsf{ho}}$, we turn to the correspondence between $>_{\mathsf{ho}}$, its optimized variant $>_{\mathsf{oh}}$, and the first-order RPO $>_{\mathsf{fo}}$.

Lemma 20. *If $u >_{\mathsf{oh}} t$ and $t >_{\mathsf{oh}} s$, then $u >_{\mathsf{oh}} s$.*

Proof. Analogous to the proof of Theorem 12. □

Lemma 21. $s\ t >_{\mathsf{oh}} s$.

Proof. By rule O3 and properties X8 and X10. □

Theorem 22 (Coincidence with Optimized Variant). *Let $>_{\mathsf{ho}}$ and $>_{\mathsf{oh}}$ be orders induced by the same precedence $\succ$ and extension operator family $\gg^{\mathsf{f}}$ (which must satisfy property X10 by the definition of $>_{\mathsf{oh}}$). Then $t >_{\mathsf{ho}} s$ if and only if $t >_{\mathsf{oh}} s$.*

Proof. By strong induction on $|s| + |t|$. We start by showing that $t >_{\mathsf{ho}} s$ implies $t >_{\mathsf{oh}} s$; then we prove the other direction.

If $t >_{\mathsf{ho}} s$ was derived by rule H1, we have $t = t_1\ t_2$ and $t_j \geq_{\mathsf{ho}} s$ for some j. Hence $t_j \geq_{\mathsf{oh}} s$ by the induction hypothesis, and $t >_{\mathsf{oh}} t_j$ by Lemma 21 or rule O1. We get $t >_{\mathsf{oh}} s$ either immediately or by Lemma 20.

If $t >_{\mathsf{ho}} s$ was derived by rule H2, we derive $t >_{\mathsf{oh}} s$ by applying O2. We must show that $\mathit{chksubs}$ implies $\mathit{chkargs}$. We have $s = s_1\ s_2$ with $t >_{\mathsf{ho}} s_1$ and $t >_{\mathsf{ho}} s_2$. Let $s = \zeta\ \bar{s}\ s_2$. We must show that $t >_{\mathsf{oh}} s'$ for all $s' \in \bar{s} \cup \{s_2\}$. If $s' = s_2$, we have $t >_{\mathsf{ho}} s_2$ immediately. Otherwise, from $t >_{\mathsf{ho}} s_1$, we have $t >_{\mathsf{ho}} s'$ by the subterm property (Theorem 14). In both cases, we get $\mathit{chkargs}(t, \bar{s})$ by the induction hypothesis.

If $t >_{\mathsf{ho}} s$ was derived by rule H3, we derive $t >_{\mathsf{oh}} s$ by applying O3. The $\mathit{chkargs}(t, \bar{s})$ condition is proved as in the H2 case. From $\bar{t} \gg^{\mathsf{f}}_{\mathsf{ho}} \bar{s}$, we derive $\bar{t} \gg^{\mathsf{f}}_{\mathsf{oh}} \bar{s}$ by the induction hypothesis and monotonicity of $\gg^{\mathsf{f}}$ (property X1).

In the other direction, if $t >_{\mathsf{oh}} s$ was derived by rule O1, we have $t' \geq_{\mathsf{oh}} s$ for all $t' \in \bar{t}$. If $t' >_{\mathsf{oh}} s$, we have $t' >_{\mathsf{ho}} s$ by the induction hypothesis; then $t >_{\mathsf{ho}} s$ follows by the subterm property (Theorem 14). Otherwise, $t' = s$, and the subterm property applies.

If $t >_{\mathsf{oh}} s$ was derived by rule O2, we derive $t >_{\mathsf{ho}} s$ by applying H2. We must show that $\mathit{chkargs}$ implies $\mathit{chksubs}$. We have $s = \zeta\ \bar{s}$ with $t >_{\mathsf{ho}} s'$ for all $s' \in \bar{s}$. The case where $\bar{s}$ is empty is trivial. Let $s = s_1\ s_2$. We must show $t >_{\mathsf{fo}} s_1$ and $t >_{\mathsf{fo}} s_2$. We have $s >_{\mathsf{oh}} s_1$ by Lemma 21; hence, $t >_{\mathsf{oh}} s >_{\mathsf{oh}} s_1$ by Lemma 20. For s_2, we have $t >_{\mathsf{oh}} s_2$ because $s_2 \in \bar{s}$. In both cases, we get $\mathit{chksubs}(t, s)$ by the induction hypothesis.

If $t >_{\mathsf{oh}} s$ was derived by rule O3, we derive $t >_{\mathsf{ho}} s$ by applying H3. The $\mathit{chksubs}(t, s)$ condition is proved as in the O2 case above. From $\bar{t} \gg^{\mathsf{f}}_{\mathsf{oh}} \bar{s}$, we derive $\bar{t} \gg^{\mathsf{f}}_{\mathsf{ho}} \bar{s}$ by the induction hypothesis and monotonicity of $\gg^{\mathsf{f}}$ (property X1). □

Corollary 23 (Coincidence with First-Order RPO). *Let* $>_{\mathsf{ho}}$ *and* $>_{\mathsf{fo}}$ *be orders induced by the same precedence* $\succ$ *and extension operator family* $\gg^{\mathsf{f}}$ *satisfying minimality of the empty tuple* (*property X10*)*. Then* $>_{\mathsf{ho}}$ *and* $>_{\mathsf{fo}}$ *coincide on first-order terms.*

Proof. By straightforward inductions on the derivation rules of $>_{\mathsf{oh}}$ and $>_{\mathsf{fo}}$. □

5 Formalization

All the proofs of Section 4 have been fully formalized in Isabelle/HOL [33] and are part of the *Archive of Formal Proofs* [8]. The formal development relies on no custom axioms; at most local assumptions such as "$\succ$ is a well-founded total order on Σ" are made. The properties of the extension operators, briefly described in Section 2, have also been formally proved. There are admittedly some discrepancies between the informal presentation and the formalization, mostly due to the divide between the familiar, set-theoretic notations used in the paper and the higher-order logic (simple type theory) used in the formalization.

The simplicity of $>_{\mathsf{ho}}$ fails to do justice to the labor of exploring the design space. Methodologically, the use of Isabelle/HOL, including its model finder Nitpick [7] and a portfolio of automatic theorem provers [6], was invaluable for designing the orders, proving their properties, and carrying out various experiments. As one example among many, at a late stage in the design process, we generalized the rules H2 and O2 to allow variable heads. Thanks to the tool support, which keeps track of what must be changed, it took us less than one hour to adapt the main proofs and convince ourselves that the new approach worked, and a few more hours to complete the proofs. Performing such changes on paper is a less reliable, and less satisfying, enterprise. Another role of the formal proofs is to serve as companions to the informal proofs, clarifying finer points.

6 Examples

Although our motivation was to design a term order suitable for higher-order superposition, we can use $>_{\mathsf{ho}}$ (and $>_{\mathsf{oh}}$) to show the termination of λ-free higher-order term rewriting systems or, equivalently, applicative term rewriting systems [25]. We present a selection of examples of how this can be done, illustrating the strengths and weaknesses of the order in this context. Many of the examples are taken from the literature, with minor adaptations—for example, we write f where some authors have $\lambda x.\, f\, x$. Since $>_{\mathsf{ho}}$ coincides with the standard RPO on first-order terms, we consider only examples featuring higher-order constructs.

To establish termination of a term rewriting system, a standard approach is to show that all of its rewrite rules $t \to s$ can be oriented as $t > s$ by a single *reduction order*: a well-founded partial order that is compatible with contexts and stable under substitutions. Regrettably, $>_{\mathsf{ho}}$ is not a reduction order since it lacks compatibility with arguments. But the conditional Theorem 16 is often sufficient in practice. Assuming that the extension operator is compatible with appending (property X9), we may apply H2 and H3 to orient rewrite rules. Moreover, we may even use H1 for rewrite rules that operate on non-function terms; supplying an argument to a non-function would violate typing.

To identify non-functions and to restrict instantiations, we assume that terms respect the typing discipline of the simply typed λ-calculus, but ignore the types when applying the term orders. Together, property X9 and the restriction on the application of H1 achieve the same effect as η-saturation [18].

For simplicity, the examples are all monolithic, but a modern termination prover would use the dependency pair framework [2] to break down a large term rewriting system into smaller components that can be analyzed separately, using different methods—or different instances of the same methods. Unless mentioned otherwise, the RPO instances considered employ the length-lexicographic extension operator for all symbols. We consistently use italics for variables and sans serif for symbols—thus, the variable f is distinct from the symbol f.

Example 24. Consider the following term rewriting system:

$$\mathsf{insert}\ (f\ n)\ (\mathsf{image}\ f\ A) \xrightarrow{1} \mathsf{image}\ f\ (\mathsf{insert}\ n\ A) \qquad \mathsf{square}\ n \xrightarrow{2} \mathsf{times}\ n\ n$$

Rule 1 captures a set-theoretic property: $\{f(n)\} \cup f[A] = f[\{n\} \cup A]$, where $f[A]$ denotes the image of set A under function f. We can prove termination using $>_{\mathsf{ho}}$: By letting $\mathsf{insert} \succ \mathsf{image}$ and $\mathsf{square} \succ \mathsf{times}$, both rules can be oriented by H2. In contrast, rule 2 is beyond the reach of the applicative order $>_{\mathsf{ap}}$ for the same reason that $\mathsf{g}\ x \not>_{\mathsf{ap}} \mathsf{f}\ x\ x$ in Example 9. The system is also beyond the scope of the uncurrying approach of Hirokawa et al. [18] because of the variable application $f\ n$ on the left-hand side of rule 1.

Example 25. The following system specifies a map function on an ML-style option type equipped with two constructors, None and Some:

$$\mathsf{omap}\ f\ \mathsf{None} \xrightarrow{1} \mathsf{None} \qquad \mathsf{omap}\ f\ (\mathsf{Some}\ n) \xrightarrow{2} \mathsf{Some}\ (f\ n)$$

To establish termination, it would appear that it suffices to apply H2 to orient both rules, using a precedence such that $\mathsf{omap} \succ \mathsf{None}, \mathsf{Some}$. However, a closer inspection reveals that the *chksubs* condition blocks the application of H2 to orient rule 2: We would need $\mathsf{omap}\ f\ (\mathsf{Some}\ n) >_{\mathsf{ho}} f\ n$, which cannot be established without further assumptions. With a typing discipline that distinguishes between options and other data, f cannot be instantiated by a term having omap as its head: omap returns an option, whereas f must return a data element. Thus, we can safely restrict $\mathit{ghd}(f)$ to $\Sigma - \{\mathsf{omap}\}$ and assign the highest precedence to omap. We then have $\mathsf{omap}\ f\ (\mathsf{Some}\ n) >_{\mathsf{ho}} f\ n$ by H2, as required to orient rule 2.

The above example suggests a general strategy for coping with variables that occur unapplied on the left-hand side of a rewrite rule and applied on the right-hand side.

Example 26. The next system is taken from Lysne and Piris [29, Example 5], with an additional rule adapted from Lifantsev and Bachmair [28, Example 6]:

$$\begin{aligned} \mathsf{iter}\ f\ n\ \mathsf{Nil} &\xrightarrow{1} n & \mathsf{sum}\ ms &\xrightarrow{3} \mathsf{iter}\ \mathsf{plus}\ \mathsf{0}\ ms \\ \mathsf{iter}\ f\ n\ (\mathsf{Cons}\ m\ ms) &\xrightarrow{2} \mathsf{iter}\ f\ (f\ n\ m)\ ms & \mathsf{iter}\ \mathsf{times}\ \mathsf{1}\ ms &\xrightarrow{4} \mathsf{prod}\ ms \end{aligned}$$

The iter function is a general iterator on lists of numbers. Reasoning about the types, we can safely take $\mathit{ghd}(f) = \Sigma - \{\mathsf{iter}, \mathsf{sum}\}$. By letting $\mathsf{sum} \succ \mathsf{iter}$ and ensuring that iter is greater than any other symbol, rule 1 can be oriented by H1, rule 2 can be oriented by H3, and rules 3 and 4 can be oriented by H2. The application of H1 is legitimate if numbers are distinguished from functions.

Example 27. The following rules are taken from Jouannaud and Rubio [23, Sect. 4.2]:

$$\mathsf{fmap}\ x\ \mathsf{Nil} \to \mathsf{Nil} \qquad \mathsf{fmap}\ x\ (\mathsf{Cons}\ f\ fs) \to \mathsf{Cons}\ (f\ x)\ (\mathsf{fmap}\ x\ fs)$$

The fmap function applies each function from a list to a value x and returns the list of results. The typing discipline allows us to take $\mathit{ghd}(f) = \Sigma - \{\mathsf{fmap}\}$. By making fmap greater than any other symbol, both rules can be oriented by H2.

Example 28. The next system is from Toyama [36, Example 4]:

$$\begin{aligned} \mathsf{ite}\ \mathsf{true}\ xs\ ys &\xrightarrow{1} xs & \mathsf{filter}\ q\ \mathsf{Nil} &\xrightarrow{3} \mathsf{Nil} \\ \mathsf{ite}\ \mathsf{false}\ xs\ ys &\xrightarrow{2} ys & \mathsf{filter}\ q\ (\mathsf{Cons}\ x\ xs) &\xrightarrow{4} \mathsf{ite}\ (q\ x)\ (\mathsf{Cons}\ x\ (\mathsf{filter}\ q\ xs))\ (\mathsf{filter}\ q\ xs) \end{aligned}$$

The typing discipline allows us to take $\mathit{ghd}(q) = \Sigma - \{\mathsf{filter}\}$. Given $\mathsf{filter} \succ \mathsf{f}$ for all $\mathsf{f} \in \Sigma$, rules 1 and 2 can be oriented by H1, and rules 3 and 4 can be oriented by H2. The application of H1 is legitimate if lists are distinguished from functions.

Example 29. Sternagel and Thiemann [34, Example 1] compare different approaches to uncurrying on the following system:

$$\begin{aligned} \mathsf{minus}\ \mathsf{0} &\xrightarrow{1} \mathsf{K}\ \mathsf{0} & \mathsf{K}\ m\ n &\xrightarrow{5} m \\ \mathsf{minus}\ m\ \mathsf{0} &\xrightarrow{2} m & \mathsf{map}\ f\ \mathsf{Nil} &\xrightarrow{6} \mathsf{Nil} \\ \mathsf{minus}\ m\ m &\xrightarrow{3} \mathsf{0} & \mathsf{map}\ f\ (\mathsf{Cons}\ m\ ms) &\xrightarrow{7} \mathsf{Cons}\ (f\ m)\ (\mathsf{map}\ f\ ms) \\ \mathsf{minus}\ (\mathsf{S}\ m)\ (\mathsf{S}\ n) &\xrightarrow{4} \mathsf{minus}\ m\ n \end{aligned}$$

The minus function implements subtraction on Peano numbers, whereas map applies a function elementwise to a finite list. We establish termination by employing $>_{\mathsf{ho}}$ with a precedence such that $\mathsf{minus} \succ \mathsf{K}, \mathsf{0}$ and $\mathsf{map} \succ \mathsf{Cons}$. Rules 2, 5, and 6 are oriented by H1; rules 1, 3, and 7 are oriented by H2; and rule 4 is oriented by H3. The application of H1 is legitimate if numbers and lists are distinguished from functions.

Example 30. Lifantsev and Bachmair [28, Example 8] define a higher-order function that applies its first argument twice to its second argument:

$$\mathsf{twice}\ f\ x \to f\ (f\ x)$$

This rewrite rule is problematic in our framework, because we cannot rely on the typing discipline to prevent the instantiation of f by a term with twice as its head. Indeed, $\mathsf{twice}\ (\mathsf{twice}\ \mathsf{S})$ is a natural way to specify the function $x \mapsto \mathsf{S}\ (\mathsf{S}\ (\mathsf{S}\ (\mathsf{S}\ x)))$.

Example 31. Toyama's recursor specification [36, Example 6] exhibits the same limitation in a more general context:

$$\mathsf{rec}\ n\ f\ \mathsf{0} \to n \qquad \mathsf{rec}\ n\ f\ (\mathsf{S}\ m) \to f\ (\mathsf{S}\ m)\ (\mathsf{rec}\ n\ f\ m)$$

Example 32. Let $\mathit{ghd}(f) = \mathit{ghd}(g) = \{\mathsf{prod}\}$, and consider the system

$$\begin{array}{ll} \mathsf{plus}\ \mathsf{0}\ m \xrightarrow{1} m & f\ \mathsf{prod} \xrightarrow{3} f \\ \mathsf{plus}\ (\mathsf{S}\ m)\ n \xrightarrow{2} \mathsf{plus}\ m\ (\mathsf{S}\ n) & f\ (g\ m) \xrightarrow{4} f\ m\ g \end{array}$$

These rules can be used to simplify nested prod terms; for example:

$$\begin{array}{l} \mathsf{prod}\ (\mathsf{prod}\ \mathsf{a}\ (\mathsf{plus}\ (\mathsf{S}\ \mathsf{0})\ \mathsf{b})) \xrightarrow{2} \mathsf{prod}\ (\mathsf{prod}\ \mathsf{a}\ (\mathsf{plus}\ \mathsf{0}\ (\mathsf{S}\ \mathsf{b}))) \xrightarrow{1} \mathsf{prod}\ (\mathsf{prod}\ \mathsf{a}\ (\mathsf{S}\ \mathsf{b})) \xrightarrow{4} \\ \mathsf{prod}\ (\mathsf{S}\ \mathsf{b})\ (\mathsf{prod}\ \mathsf{a}) \xrightarrow{4} \mathsf{prod}\ (\mathsf{S}\ \mathsf{b})\ \mathsf{a}\ \mathsf{prod} \xrightarrow{3} \mathsf{prod}\ (\mathsf{S}\ \mathsf{b})\ \mathsf{a} \end{array}$$

The $>_{\mathsf{ho}}$ order can be employed by taking $\gg^{\mathsf{prod}}$ to be the multiset extension and by relying on typing to orient rule 1 with H1. The applicative order $>_{\mathsf{ap}}$ fails because a combination of lexicographic and multiset extensions is needed to orient rules 2 and 4. The uncurrying approach of Hirokawa et al. [18] also fails because of the applied variables on the left-hand side of rule 4.

Carsten Fuhs, a developer of the AProVE termination prover, generously offered to apply his tool to our examples, expressed as untyped applicative term rewriting systems. Using AProVE's web interface with a 60 s time limit, he could establish the termination of Examples 24, 25, 29, 30, and 32. The tool timed out for Examples 26–28 and 31. For Example 32, the tool found a complex proof involving several applications of linear polynomial interpretations, dependency pairs, and 2×2 matrix interpretations (to cope with rule 4). Although our focus is on superposition, it would be interesting to implement the new RPO in a tool such as AProVE and to conduct a more systematic evaluation on standard higher-order termination benchmarks against higher-order termination provers such as THOR [11] and WANDA [26].

7 Discussion

Rewriting of λ-free higher-order terms has been amply studied in the literature, under various names such as applicative term rewriting [25] and simply typed term rewriting [38]. Translations from higher-order to first-order term rewriting systems were designed by Aoto and Yamada [1], Toyama [36], Hirokawa et al. [18], and others. Toyama also studied S-expressions, a formalism that regards $((\mathsf{f}\ \mathsf{a})\ \mathsf{b})$ and $(\mathsf{f}\ \mathsf{a}\ \mathsf{b})$ as distinct. For higher-order terms with λ-abstraction, various frameworks have been proposed, including Nipkow's higher-order rewrite systems [32], Jouannaud and Okada's algebraic functional systems [22], Blanqui's inductive data type systems [9], and Kop's algebraic functional systems with metavariables [26]. Kop's thesis [26, Chapter 3] includes a comprehensive overview.

When designing our RPO $>_{\mathsf{ho}}$, we aimed at full coincidence with the first-order case. Our goal is to gradually transform first-order automatic provers into higher-order provers. By carefully generalizing the proof calculi and data structures, we aim at designing provers that behave like first-order provers on first-order problems, perform mostly like first-order provers on higher-order problems that are mostly first-order, and scale up to arbitrary higher-order problems.

The $>_{\mathsf{ho}}$ order is in some ways less flexible than the hybrid curried–uncurried approaches, where the currying is one more parameter that can be adjusted. In exchange, it

raises the level of abstraction, by providing a uniform view of higher-order terms, and it works in the open-world setting of higher-order proof search. For example, consider the proof obligation $\exists g.\, \forall x, y.\, g\ x\ y = \mathsf{f}\ y\ x$ and the SK combinator definitions $\forall x, y.\, \mathsf{K}\ x\ y = x$ and $\forall x, y, z.\, \mathsf{S}\ x\ y\ z = x\ z\ (y\ z)$. A prover will need to synthesize the witness $\mathsf{S}\ (\mathsf{K}\ (\mathsf{S}\ \mathsf{f}))\ \mathsf{K}$, representing $\lambda x\, y.\, \mathsf{f}\ y\ x$, for the existential variable g. A hybrid approach such as HORPO might infer arity 2 for f based on the problem, but then the witness, in which f appears unapplied, cannot be expressed.

An open question is whether it is possible to design an order that largely coincides with the first-order RPO while enjoying compatibility with arbitrary contexts. This could presumably be achieved by weakening rule H1 and strengthening the *chksubs* condition of H2 and H3 accordingly; so far, our attempts have resulted only in a rediscovery of the applicative RPO.

For superposition, richer type systems would be desirable. These could be incorporated either by simply ignoring the types, by encoding them in the terms, or by generalizing the order. Support for λ-abstraction would be useful but challenging. Any well-founded order enjoying the subterm property would need to distinguish β-equivalent terms, to exclude the cycle $\mathsf{a} =_\beta (\lambda x.\, \mathsf{a})\ (\mathsf{f}\ \mathsf{a}) > \mathsf{f}\ \mathsf{a} > \mathsf{a}$. We could aim at compatibility with β-reduction, but even this property might be irrelevant for higher-order superposition. It might even be preferable to avoid λ-abstractions altogether, by relying on SK combinators or by adding new symbols and their definitions during proof search.

Acknowledgment. We are grateful to Stephan Merz, Tobias Nipkow, and Christoph Weidenbach for making this research possible; to Heiko Becker and Dmitriy Traytel for proving some theorems on the lexicographic and multiset extensions in Isabelle/HOL; to Carsten Fuhs for his invaluable feedback and his experiments; to Alexander Steen for pointing us to related work; to Alexander Bentkamp for finding an illuminating example; and to Simon Cruanes, Mark Summerfield, Dmitriy Traytel, and the anonymous reviewers for suggesting textual improvements.

Blanchette has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 713999, Matryoshka). Wand is supported by the Deutsche Forschungsgemeinschaft (DFG) grant Hardening the Hammer (NI 491/14-1).

References

[1] Aoto, T., Yamada, T.: Termination of simply typed term rewriting by translation and labelling. In: Nieuwenhuis, R. (ed.) Rewriting Techniques and Applications (RTA 2003). LNCS, vol. 2706, pp. 380–394. Springer (2003)

[2] Arts, T., Giesl, J.: Termination of term rewriting using dependency pairs. Theor. Comput. Sci. 236(1-2), 133–178 (2000)

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

[4] Benzmüller, C., Kohlhase, M.: Extensional higher-order resolution. In: Kirchner, C., Kirchner, H. (eds.) Conference on Automated Deduction (CADE-15). LNCS, vol. 1421, pp. 56–71. Springer (1998)

[5] Benzmüller, C., Miller, D.: Automation of higher-order logic. In: Siekmann, J.H. (ed.) Computational Logic, Handbook of the History of Logic, vol. 9, pp. 215–254. Elsevier (2014)

[6] Blanchette, J.C., Kaliszyk, C., Paulson, L.C., Urban, J.: Hammering towards QED. J. Formalized Reasoning 9(1), 101–148 (2016)

[7] Blanchette, J.C., Nipkow, T.: Nitpick: A counterexample generator for higher-order logic based on a relational model finder. In: Kaufmann, M., Paulson, L.C. (eds.) Interactive Theorem Proving (ITP 2010). LNCS, vol. 6172, pp. 131–146. Springer (2010)
[8] Blanchette, J.C., Waldmann, U., Wand, D.: Formalization of recursive path orders for lambda-free higher-order terms. Archive of Formal Proofs (2016), formal proof development, https://isa-afp.org/entries/Lambda_Free_RPOs.shtml
[9] Blanqui, F.: Termination and confluence of higher-order rewrite systems. In: Bachmair, L. (ed.) Rewriting Techniques and Applications (RTA 2000). LNCS, vol. 1833, pp. 47–61. Springer (2000)
[10] Blanqui, F., Jouannaud, J., Rubio, A.: The computability path ordering. Log. Meth. Comput. Sci. 11(4) (2015)
[11] Bofill, M., Borralleras, C., Rodríguez-Carbonell, E., Rubio, A.: The recursive path and polynomial ordering for first-order and higher-order terms. J. Log. Comput. 23(1), 263–305 (2013)
[12] Bofill, M., Rubio, A.: Paramodulation with non-monotonic orderings and simplification. J. Autom. Reasoning 50(1), 51–98 (2013)
[13] Brown, C.E., Smolka, G.: Analytic tableaux for simple type theory and its first-order fragment. Log. Meth. Comput. Sci. 6(2) (2010)
[14] Codish, M., Giesl, J., Schneider-Kamp, P., Thiemann, R.: SAT solving for termination proofs with recursive path orders and dependency pairs. J. Autom. Reasoning 49(1), 53–93 (2012)
[15] Dershowitz, N., Manna, Z.: Proving termination with multiset orderings. Commun. ACM 22(8), 465–476 (1979)
[16] Ganzinger, H., Hagen, G., Nieuwenhuis, R., Oliveras, A., Tinelli, C.: DPLL(T): Fast decision procedures. In: Alur, R., Peled, D.A. (eds.) Computer Aided Verification (CAV 2004). LNCS, vol. 3114, pp. 175–188. Springer (2004)
[17] Henkin, L.: Completeness in the theory of types. J. Symb. Log. 15(2), 81–91 (1950)
[18] Hirokawa, N., Middeldorp, A., Zankl, H.: Uncurrying for termination and complexity. J. Autom. Reasoning 50(3), 279–315 (2013)
[19] Huet, G., Oppen, D.C.: Equations and rewrite rules: A survey. In: Book, R.V. (ed.) Formal Language Theory: Perspectives and Open Problems. pp. 349–405. Academic Press (1980)
[20] Huet, G.P.: A mechanization of type theory. In: Nilsson, N.J. (ed.) International Joint Conference on Artificial Intelligence (IJCAI-73). pp. 139–146. William Kaufmann (1973)
[21] Hughes, R.J.M.: Super-combinators: A new implementation method for applicative languages. In: ACM Symposium on LISP and Functional Programming (LFP '82). pp. 1–10. ACM Press (1982)
[22] Jouannaud, J., Okada, M.: A computation model for executable higher-order algebraic specification languages. In: Logic in Computer Science (LICS '91). pp. 350–361. IEEE Computer Society (1991)
[23] Jouannaud, J., Rubio, A.: A recursive path ordering for higher-order terms in η-long β-normal form. In: Ganzinger, H. (ed.) Rewriting Techniques and Applications (RTA-96). LNCS, vol. 1103, pp. 108–122. Springer (1996)
[24] Jouannaud, J., Rubio, A.: Polymorphic higher-order recursive path orderings. J. ACM 54(1), 2:1–2:48 (2007)
[25] Kennaway, R., Klop, J.W., Sleep, M.R., de Vries, F.: Comparing curried and uncurried rewriting. J. Symb. Comput. 21(1), 15–39 (1996)
[26] Kop, C.: Higher Order Termination. Ph.D. thesis, Vrije Universiteit Amsterdam (2012)
[27] Kop, C., van Raamsdonk, F.: A higher-order iterative path ordering. In: Cervesato, I., Veith, H., Voronkov, A. (eds.) Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2008). LNCS, vol. 5330, pp. 697–711. Springer (2008)

[28] Lifantsev, M., Bachmair, L.: An LPO-based termination ordering for higher-order terms without λ-abstraction. In: Grundy, J., Newey, M.C. (eds.) Theorem Proving in Higher Order Logics (TPHOLs '98). LNCS, vol. 1479, pp. 277–293. Springer (1998)

[29] Lysne, O., Piris, J.: A termination ordering for higher order rewrite system. In: Hsiang, J. (ed.) Rewriting Techniques and Applications (RTA-95). LNCS, vol. 914, pp. 26–40. Springer (1995)

[30] Miller, D., Cohen, E.L., Andrews, P.B.: A look at TPS. In: Loveland, D.W. (ed.) Conference on Automated Deduction (CADE-6). LNCS, vol. 138, pp. 50–69. Springer (1982)

[31] Nieuwenhuis, R., Rubio, A.: Paramodulation-based theorem proving. In: Robinson, J.A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. I, pp. 371–443. Elsevier and MIT Press (2001)

[32] Nipkow, T.: Higher-order critical pairs. In: Logic in Computer Science (LICS '91). pp. 342–349. IEEE Computer Society (1991)

[33] Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL: A Proof Assistant for Higher-Order Logic, LNCS, vol. 2283. Springer (2002)

[34] Sternagel, C., Thiemann, R.: Generalized and formalized uncurrying. In: Tinelli, C., Sofronie-Stokkermans, V. (eds.) Frontiers of Combining Systems (FroCoS 2011). LNCS, vol. 6989, pp. 243–258. Springer (2011)

[35] Thiemann, R., Allais, G., Nagele, J.: On the formalization of termination techniques based on multiset orderings. In: Tiwari, A. (ed.) Rewriting Techniques and Applications (RTA '12). LIPIcs, vol. 15, pp. 339–354. Schloss Dagstuhl—Leibniz-Zentrum für Informatik (2012)

[36] Toyama, Y.: Termination of S-expression rewriting systems: Lexicographic path ordering for higher-order terms. In: van Oostrom, V. (ed.) Rewriting Techniques and Applications (RTA 2004). LNCS, vol. 3091, pp. 40–54. Springer (2004)

[37] Turner, D.A.: A new implementation technique for applicative languages. Software: Practice and Experience 9(1), 31–49 (1979)

[38] Yamada, T.: Confluence and termination of simply typed term rewriting systems. In: Middeldorp, A. (ed.) Rewriting Techniques and Applications (RTA 2001). LNCS, vol. 2051, pp. 338–352. Springer (2001)

[39] Zantema, H.: Termination. In: Bezem, M., Klop, J.W., de Vrijer, R. (eds.) Term Rewriting Systems, Cambridge Tracts in Theoretical Computer Science, vol. 55, pp. 181–259. Cambridge University Press (2003)