The Embedding Path Order for Lambda-Free Higher-Order Terms

Alexander Bentkamp
Vrije Universiteit Amsterdam, De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands
a.bentkamp@vu.nl
https://orcid.org/0000-0002-7158-3595

Abstract

This paper describes the embedding path order, a variant of the recursive path order (RPO), for untyped λ-free higher-order terms (also called applicative first-order terms). Unlike other higher-order variants of RPO, it is a ground-total simplification order, making it more suitable for the superposition calculus. However, it does not coincide with RPO on first-order terms. I formally proved the order's theoretical properties in Isabelle/HOL and evaluated the order in the superposition prover Zipperposition.

2012 ACM Subject Classification Computing methodologies → Theorem proving algorithms

Keywords and phrases term order, superposition, applicative first-order logic

1 Introduction

Superposition [2] is one of the most successful calculi for proof search in first-order logic with equality. To restrict the search space, it uses a term order, which in practice is usually the Knuth-Bendix order (KBO) or the recursive path order (RPO). Extending this calculus to higher-order logic requires suitable higher-order term orders.

Besides soundness, the most important theoretical property of the superposition calculus is refutational completeness. The proof of refutational completeness of the first-order superposition calculus relies on a ground-total and well-founded simplification order. For a higher-order superposition calculus, a simplification order for higher-order terms modulo β-conversion would be desirable but does not exist, as demonstrated by $\mathsf{a} =_\beta (\lambda x.\ \mathsf{a})\ \mathsf{a} > \mathsf{a}$, where the inequality follows from the subterm property.

However, for λ-free higher-order terms, also known as applicative first-order terms, ground-total simplification orders exist, for instance the generalization of KBO developed by Becker et al. [3] and the term order presented in this paper. Such term orders can be put to use in superposition variants for λ-free higher-order logic, as implemented in Ehoh [38], and also for richer higher-order logics, as in my recent work [5] and in unpublished work by Bhayat and Reger [7].

In contrast to KBO, a λ-free higher-order variant of RPO that is a simplification order and coincides with first-order RPO on (curried) first-order terms is impossible, which the following example shows: If $\mathsf{g} \succ \mathsf{f} \succ \mathsf{b} \succ \mathsf{a}$, then $\mathsf{g}\ \mathsf{b} > \mathsf{f}\ (\mathsf{g}\ \mathsf{a})\ \mathsf{b}$ by coincidence with first-order RPO, corresponding to $\mathsf{g}(\mathsf{b}) > \mathsf{f}(\mathsf{g}(\mathsf{a}), \mathsf{b})$ in first-order syntax, but $\mathsf{g} < \mathsf{f}\ (\mathsf{g}\ \mathsf{a})$ by the subterm property and hence $\mathsf{g}\ \mathsf{b} < \mathsf{f}\ (\mathsf{g}\ \mathsf{a})\ \mathsf{b}$ by compatibility with contexts, yielding a contradiction.

This leaves the question of whether there is a ground-total simplification order on λ-free higher-order terms that resembles RPO but does not coincide with RPO on first-order terms. One candidate is the applicative RPO, which is obtained by encoding λ-free higher-order terms applicatively into first-order logic via a binary symbol @ representing application and then using first-order RPO. The drawback of this approach is that the symbol @ becomes pervasive, which undermines RPO's principle of comparing the precedence of different symbols.

Moreover, it is impossible to assign different extension orders such as the lexicographic or multiset extension to different function symbols because the only applied function symbol in the encoding is @.

The embedding path order (EPO[1]) presented here allows different extension operators for different function symbols (Section 3). The main difference to RPO lies in using the notion of embeddings where RPO uses the notion of direct subterms (Section 4). EPO is a ground-total simplification order and I have formally proved this property in Isabelle/HOL (Section 5). I illustrate the strengths and weaknesses of EPO on several examples (Section 6). I have implemented EPO in the superposition prover Zipperposition (Section 7) and evaluated it on TPTP [37] and Sledgehammer [34] benchmarks (Section 8).

In the literature, there are several other variants of RPO for higher-order terms. Blanchette et al.'s RPO for λ-free higher-order terms [9] resembles EPO the most, but it sacrifices compatibility with contexts to be able to coincide with first-order RPO. Superposition with such non-monotonic orders is possible but compromises theoretical simplicity and to some degree practical efficiency, as demonstrated by Bofill and Rubio [13] and in my previous work [6]. Although targeting the more difficult problem of providing useful orders for full higher-order terms with λ-abstractions, the following RPO variants are vaguely related to the present work: Lifantsev and Bachmair's lexicographic path-order on λ-free higher-order terms [29], Jouannaud and Rubio's higher-order RPO (HORPO) [24], Kop and Van Raamsdonk's iterative HORPO [26], the HORPO extension with polynomial interpretation orders by Bofill et al. [12], and the computability path order by Blanqui et al. [11]. However, these orders all lack ground-totality and, except for Lifantsev and Bachmair's order, the subterm property for terms of different types.

2 Preliminaries

We fix a set of variables $\mathcal{V}$ and a non-empty (possibly infinite) set of symbols Σ. We reserve the names x, y, z for variables and $\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{f}, \mathsf{g}, \mathsf{h}$ for symbols.

In untyped λ-free higher-order logic, a term is defined inductively as being either a variable, a symbol, or an application $s\ t$, where s and t are terms.

We reserve the names t, s, v, u for terms and use $\mathcal{T}$ to denote the set of all terms. Application is left-associative, i.e., $s\ t\ u = (s\ t)\ u$. These terms are isomorphic to applicative terms [25]. Any term can be written as $\zeta\ \bar{t}_n$ using spine notation [15], where ζ is a non-application term, called *head*, and $\bar{t}_n$ is a tuple of terms, called *arguments*. It represents the term $\zeta\ t_1\ \ldots\ t_n$. Here and elsewhere, $\bar{t}_n$ or $\bar{t}$ stands for the tuple $(t_1, \ldots, t_n)$. We write $()$ for the empty tuple, t for the singleton tuple (t), and $\bar{s} \cdot \bar{t}$ for the concatenation of the tuples $\bar{s}$ and $\bar{t}$.

The *size* $|t|$ of a term t is inductively defined as 1 if $t \in \mathcal{V} \cup \Sigma$ and as $|t_1| + |t_2|$ if t is an application $t_1\ t_2$. A *subterm* of a term t is inductively defined as being either t itself or, if t is an application $t_1\ t_2$, a subterm of t_1 or of t_2. The *positions* of a term are tuples containing the elements left and right and are defined as follows: If the given term is a head, its only position is $()$; if the given term is an application $t\ s$, the tuple $\mathsf{left} \cdot p$ is a position of $t\ s$ for each position p of t and $\mathsf{right} \cdot q$ is a position of $t\ s$ for each position q of s.

The well-known embedding step relation $\longrightarrow_{\mathsf{emb}}$ is inductively defined as follows: For any terms s and t, there is a left-embedding step $t\ s \longrightarrow_{\mathsf{emb}} t$ at position $()$ and a right-embedding

[1] The unrelated exptime path order [19] has the same abbreviation. In case ambiguity is possible, this abbreviation should be avoided.

step $t\ s \longrightarrow_{\mathsf{emb}} s$ at position $()$. If for $X = \mathsf{left}$ or $X = \mathsf{right}$ there is an X-embedding step $t \longrightarrow_{\mathsf{emb}} t'$ at some position p, then there is an X-embedding step $t\ s \longrightarrow_{\mathsf{emb}} t'\ s$ at position $\mathsf{left} \cdot p$ and an X-embedding step $s\ t \longrightarrow_{\mathsf{emb}} s\ t'$ at position $\mathsf{right} \cdot p$. Let the embedding relation $\trianglerighteq_{\mathsf{emb}}$ be the reflexive transitive closure of $\longrightarrow_{\mathsf{emb}}$. For example, $\mathsf{f\ a\ b\ c\ d} \longrightarrow_{\mathsf{emb}} \mathsf{a\ b\ c\ d}$ is a right-embedding step at position $(\mathsf{left}, \mathsf{left}, \mathsf{left})$; $\mathsf{f\ a\ b\ c\ d} \longrightarrow_{\mathsf{emb}} \mathsf{f\ a\ c\ d}$ is a left-embedding step at position $(\mathsf{left}, \mathsf{left})$; and $\mathsf{f\ (g\ (h\ a)\ b)\ c} \longrightarrow_{\mathsf{emb}} \mathsf{f\ (g\ h\ b)\ c}$ is a left-embedding step at position $(\mathsf{left}, \mathsf{right}, \mathsf{left}, \mathsf{right})$.

For a term $\xi\ \bar{t}_n$ with $n > 0$, we define $\mathit{chop}(\xi\ \bar{t}_n)$ as the term resulting from applying t_1 to the remaining arguments, i.e., $\mathit{chop}(\xi\ \bar{t}_n) = t_1\ t_2\ \ldots\ t_n$. For example, $\mathit{chop}(\mathsf{f\ (g\ a)\ (h\ b)}) = \mathsf{g\ a\ (h\ b)}$.

Given a binary 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 binary relation $>$ on λ-free higher-order terms is a *simplification order* if it is irreflexive (i.e., $t \not> t$), is transitive (i.e., $u > t > s \Rightarrow u > s$), is compatible with contexts (i.e., $t > s \Rightarrow u\ (t\ \bar{v}) > u\ (s\ \bar{v})$), is stable under substitutions (i.e., $t > s \Rightarrow t\sigma > s\sigma$), and has the subterm property (i.e., $t \geq s$ if s is a subterm of t). It is *ground-total* if for all distinct ground terms s and t either $t > s$ or $t < s$. It is *well founded* if there is no infinite descending chain $t_1 > t_2 > \cdots$.

3 Extension Operators

In the spirit of RPO, EPO compares the heads of terms and, in case of equality, proceeds to compare the argument tuples. There is a variety of ways to extend a binary relation $>$ on an arbitrary set A to a binary relation $\gg$ on tuples A^*, which we call extension operators. We define extension operators on binary relations, not on partial orders, because they are used in the definition of EPO at a point where we have not shown EPO to be a partial order yet.

▶ **Definition 1.** We define the following properties of extension operators $> \mapsto \gg$, which are required for EPO to be a ground-total and well-founded simplification order. Here, given a function $h : A \to A$, we write $h(\bar{a})$ for the componentwise application of h to $\bar{a}$.

X1. Monotonicity: $\bar{b} \gg_1 \bar{a}$ implies $\bar{b} \gg_2 \bar{a}$ if for all $a, b \in A$, $b >_1 a$ implies $b >_2 a$
X2. Preservation of stability:
$\bar{b} \gg \bar{a}$ implies $h(\bar{b}) \gg h(\bar{a})$ if for all $a, b \in \bar{a} \cup \bar{b}$, $b > a$ implies $h(b) > h(a)$
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 $\bar{c} \cdot b \cdot \bar{d} \gg \bar{c} \cdot a \cdot \bar{d}$
X7. Preservation of totality: $\gg$ is total if $>$ is total
X8. Compatibility with prepending: $\bar{b} \gg \bar{a}$ implies $c \cdot \bar{b} \gg c \cdot \bar{a}$
X9. Compatibility with appending: $\bar{b} \gg \bar{a}$ implies $\bar{b} \cdot c \gg \bar{a} \cdot c$
X10. Minimality of the empty tuple: $a \gg ()$ for all $a \in A$

The length-lexicographic extension operator, left-to-right or right-to-left, fulfills all these properties:

▶ **Definition 2.** The *left-to-right length-lexicographic extension operator* $> \mapsto \gg^{\mathrm{ltr}}$ is defined inductively as follows: $\bar{a}_m \gg^{\mathrm{ltr}} \bar{b}_n$ if $m > n$; or $m = n > 0$ and $a_1 > b_1$; or $m = n > 0$, $a_1 = b_1$, and $(a_2, \ldots, a_m) \gg^{\mathrm{ltr}} (b_2, \ldots, b_n)$. The *right-to-left length-lexicographic extension operator* $> \mapsto \gg^{\mathrm{rtl}}$ is defined inductively as follows: $\bar{a}_m \gg^{\mathrm{rtl}} \bar{b}_n$ if $m > n$; or $m = n > 0$ and $a_m > b_n$; or $m = n > 0$, $a_m = b_n$, and $(a_1, \ldots, a_{m-1}) \gg^{\mathrm{rtl}} (b_1, \ldots, b_{n-1})$.

The multiset extension operator fulfills all properties except X7, but if combined with a lexicographic comparison as a tie-breaker, it fulfills all properties as well:

▶ **Definition 3.** The *multiset extension operator with tie-breaker* $> \mapsto \gg^{\mathrm{ms}}$ is defined as follows: $\bar{a} \gg^{\mathrm{ms}} \bar{b}$ if the multiset containing the elements of $\bar{a}$ is larger than the multiset containing the elements of $\bar{b}$ with respect to Dershowitz and Manna's multiset order [18]; or if the two multisets are equal and $\bar{a} \gg^{\mathrm{ltr}} \bar{b}$.

Blanchette et al. [9] give a more detailed account of different extension operators. Their list of properties is identical with the one above, except for X2, which they originally formulated differently but corrected in their technical report [8].

4 The Order

Any simplification order has the embedding property, i.e., the property that $t \trianglerighteq_{\mathsf{emb}} s$ implies $t \succeq s$ [1, Lemma 5.4.7]. The fundamental idea of EPO is to enforce the embedding property by replacing the notion of subterms used in the definition of RPO by the notion of embeddings. Performed naively, this causes issues with stability under substitution and with the time complexity of the order computation due to the large amount of possible embedding steps. Both of these issues are addressed by EPO.

▶ **Definition 4** (EPO)**.** Let $\succ$ be a well-founded total order on Σ. For each $\mathsf{f} \in \Sigma$, let $> \mapsto \gg^{\mathsf{f}}$ be an extension operator satisfying the properties of Definition 1. The induced *embedding path order* $>_{\mathsf{ep}}$ is inductively defined as follows: $t >_{\mathsf{ep}} s$ if any of the following conditions is met, where $t = \xi\ \bar{t}_n$ and $s = \zeta\ \bar{s}_m$:

E1. $n > 0$ and $\mathit{chop}(t) \geq_{\mathsf{ep}} s$
E2. $\xi, \zeta \in \Sigma$, $\xi \succ \zeta$, and either $m = 0$ or $t >_{\mathsf{ep}} \mathit{chop}(s)$
E3. $\xi, \zeta \in \Sigma$, $\xi = \zeta$, $\bar{t}_n \gg^{\zeta}_{\mathsf{ep}} \bar{s}_m$, and either $m = 0$ or $t >_{\mathsf{ep}} \mathit{chop}(s)$
E4. $\xi, \zeta \in \mathcal{V}$, $\xi = \zeta$, $\bar{t}_n \gg^{\mathsf{f}}_{\mathsf{ep}} \bar{s}_m$ for all $\mathsf{f} \in \Sigma$, $n > 0$, and either $m = 0$ or $\mathit{chop}(t) >_{\mathsf{ep}} \mathit{chop}(s)$

The following examples illustrate the differences between RPO and EPO on first-order terms. We use the precedence $\mathsf{g} \succ \mathsf{f} \succ \mathsf{c} \succ \mathsf{b} \succ \mathsf{a}$ and the left-to-right length-lexicographic extension for both orders.

$\mathsf{f}\ (\mathsf{g}\ \mathsf{a})\ \mathsf{b} <_{\mathsf{rp}} \mathsf{g}\ \mathsf{b}$	$\mathsf{f}\ (\mathsf{g}\ \mathsf{a})\ \mathsf{c} <_{\mathsf{rp}} \mathsf{g}\ \mathsf{b}$	$\mathsf{f}\ \mathsf{b}\ x\ y >_{\mathsf{rp}} \mathsf{f}\ \mathsf{a}\ y\ y$
$\mathsf{f}\ (\mathsf{g}\ \mathsf{a})\ \mathsf{b} >_{\mathsf{ep}} \mathsf{g}\ \mathsf{b}$	$\mathsf{f}\ (\mathsf{g}\ \mathsf{a})\ \mathsf{c} >_{\mathsf{ep}} \mathsf{g}\ \mathsf{b}$	$\mathsf{f}\ \mathsf{b}\ x\ y \not\gtrless_{\mathsf{ep}} \mathsf{f}\ \mathsf{a}\ y\ y$

The first term pair illustrates that RPO does not have the embedding property, whereas EPO does. The relation $\mathsf{f}\ (\mathsf{g}\ \mathsf{a})\ \mathsf{b} >_{\mathsf{ep}} \mathsf{g}\ \mathsf{b}$ can be shown by applying E1. E1 requires $\mathsf{g}\ \mathsf{a}\ \mathsf{b} >_{\mathsf{ep}} \mathsf{g}\ \mathsf{b}$, which holds by E3. Finally we need E2 to show $\mathsf{g}\ \mathsf{a}\ \mathsf{b} >_{\mathsf{ep}} \mathsf{b}$. The second term pair shows that there are further disagreements between the two orders, even if one term is not an embedding of the other. As above, $\mathsf{f}\ (\mathsf{g}\ \mathsf{a})\ \mathsf{c} >_{\mathsf{ep}} \mathsf{g}\ \mathsf{b}$ can be established by applying E1, followed by E3 and E2. The third term pair is comparable with RPO but incomparable with EPO. In general, EPO cannot judge a term to be smaller if it contains more occurrences of a variable. I conjecture that there are no first-order terms comparable with EPO but incomparable with RPO. In this sense, EPO is weaker than RPO on first-order terms.

The remainder of this section justifies some of the design decisions in the definition of EPO and explains how they contribute to make EPO a ground-total simplification order that can be computed fairly efficiently.

Condition E1 enforces the embedding property in a similar way as RPO enforces the subterm property. This underlying idea gives EPO its name. A naive approach would be to test all embedding steps to enforce the embedding property, but it is sufficient to test only the embedding step chop, yielding a better computational complexity. The remaining conditions follow a similar structure as RPO, but contain subconditions on chop where RPO has subconditions on subterms.

To achieve stability under substitutions, it is essential to demand $\mathit{chop}(t) >_{\mathsf{ep}} \mathit{chop}(s)$ instead of $t >_{\mathsf{ep}} \mathit{chop}(s)$ in E4, as the following examples show. If $>'_{\mathsf{ep}}$ is the relation obtained from $>_{\mathsf{ep}}$ by replacing '$\mathit{chop}(t)$' by 't' in E4, then we have

$$x\ \mathsf{f}\ \mathsf{f} >'_{\mathsf{ep}} x\ x, \text{ but } \mathsf{f}\ y\ \mathsf{f}\ \mathsf{f} \not>'_{\mathsf{ep}} \mathsf{f}\ y\ (\mathsf{f}\ y) \qquad x\ \mathsf{f}\ x >'_{\mathsf{ep}} x\ (x\ \mathsf{f}), \text{ but } y\ \mathsf{f}\ \mathsf{f}\ (y\ \mathsf{f}) \not>'_{\mathsf{ep}} y\ \mathsf{f}\ (y\ \mathsf{f}\ \mathsf{f})$$

Using $>_{\mathsf{ep}}$, all of these pairs are incomparable.

In condition E4, it is crucial to check $\bar{t}_n \gg^{\mathsf{f}}_{\mathsf{ep}} \bar{s}_m$ for all $\mathsf{f} \in \Sigma$. In contrast, λ-free KBO [3] and λ-free RPO [9] allow us to use a map ghd from variables to possible ground heads that might occur when a variable is instantiated. The corresponding condition in these orders then states '$\bar{t}_n \gg^{\mathsf{f}}_{\mathsf{ep}} \bar{s}_m$ for all $\mathsf{f} \in \mathit{ghd}(\zeta)$'. For EPO, this approach cannot be used. For example, assume $\mathsf{b} \succ \mathsf{a}$, $\mathit{ghd}(x) = \{\mathsf{f}\}$, and that f uses the left-to-right length-lexicographic extension. Then we would have $x\ \mathsf{b}\ \mathsf{a} > x\ \mathsf{a}\ \mathsf{b}$ if we checked only the extension orders for $\mathit{ghd}(x)$. This contradicts stability under substitutions because, if g uses the right-to-left length-lexicographic extension, $y\ \mathsf{g}\ \mathsf{b}\ \mathsf{a}$ and $y\ \mathsf{g}\ \mathsf{a}\ \mathsf{b}$ are incomparable, assuming $\mathit{ghd}(y) = \{\mathsf{f}\}$.

EPO is not a simplification order when (non-length-)lexicographic extensions are used. With the left-to-right lexicographic extension, it lacks compatibility with contexts because for $\mathsf{g} \succ \mathsf{f} \succ \mathsf{b} \succ \mathsf{a}$, we have $\mathsf{f}\ (\mathsf{g}\ \mathsf{a}) >_{\mathsf{ep}} \mathsf{g}$ but $\mathsf{f}\ (\mathsf{g}\ \mathsf{a})\ \mathsf{b} <_{\mathsf{ep}} \mathsf{g}\ \mathsf{b}$. With the right-to-left lexicographic extension, it lacks stability under substitutions because $x\ \mathsf{f} > x$ but $\mathsf{f}\ y\ \mathsf{f} \not> \mathsf{f}\ y$. With the right-to-left lexicographic extension, it also lacks well-foundedness because for $\mathsf{f} \succ \mathsf{b} \succ \mathsf{a}$, we have $\mathsf{f}\ \mathsf{b} >_{\mathsf{ep}} \mathsf{f}\ \mathsf{b}\ \mathsf{a} >_{\mathsf{ep}} \mathsf{f}\ \mathsf{b}\ \mathsf{a}\ \mathsf{a} >_{\mathsf{ep}} \cdots$.

5 Properties of the Order

EPO fulfills all the properties of a ground-total simplification order. The proofs in this section have been developed in Isabelle/HOL and published in the Archive of Formal Proofs [4]. They are inspired by the corresponding proofs about λ-free RPO [9], which in turn were adapted from Baader and Nipkow [1] and Zantema [39].

▶ **Theorem 5** (Transitivity). *$u >_{\mathsf{ep}} t$ and $t >_{\mathsf{ep}} s$ implies $u >_{\mathsf{ep}} s$.*

Proof. By well-founded induction on the multiset $\{\!|\, |u|, |t|, |s| \,|\!\}$. with respect to the multiset extension [18] of $>$ on $\mathbb{N}$. Let $u = \psi\ \bar{u}_r$, $t = \xi\ \bar{t}_n$ and $s = \zeta\ \bar{s}_m$.

If $u >_{\mathsf{ep}} t$ is derived by E1, then $r > 0$ and $\mathit{chop}(u) \geq_{\mathsf{ep}} t$. Applying the induction hypothesis to $\mathit{chop}(u)$, t, s, it follows that $\mathit{chop}(u) >_{\mathsf{ep}} s$ and hence $u >_{\mathsf{ep}} s$ by E1.

If $u >_{\mathsf{ep}} t$ is derived by E2 or E3 and $t >_{\mathsf{ep}} s$ is derived by E1, then $n > 0$ and $u >_{\mathsf{ep}} \mathit{chop}(t) \geq_{\mathsf{ep}} s$. Applying the induction hypothesis to u, $\mathit{chop}(t)$, s, it follows that $u >_{\mathsf{ep}} s$.

If $u >_{\mathsf{ep}} t$ is derived by E4 and $t >_{\mathsf{ep}} s$ is derived by E1, then $r > 0$, $n > 0$, and $\mathit{chop}(u) >_{\mathsf{ep}} \mathit{chop}(t) \geq_{\mathsf{ep}} s$. By applying the induction hypothesis to $\mathit{chop}(u)$, $\mathit{chop}(t)$, s, we get $\mathit{chop}(u) >_{\mathsf{ep}} s$. By E1, it follows that $u >_{\mathsf{ep}} s$.

If $u >_{\mathsf{ep}} t$ and $t >_{\mathsf{ep}} s$ are derived by E2 and E2, by E2 and E3, or by E3 and E2, respectively, then $\psi \succ \zeta$ and $t >_{\mathsf{ep}} \mathit{chop}(s)$. If $m = 0$, we can apply E2 directly to obtain $u >_{\mathsf{ep}} s$. If $m > 0$, by the induction hypothesis for u, t, $\mathit{chop}(s)$, it follows from $u >_{\mathsf{ep}} t$ and $t >_{\mathsf{ep}} \mathit{chop}(s)$ that $u >_{\mathsf{ep}} \mathit{chop}(s)$. Then we can apply E2 to obtain $u >_{\mathsf{ep}} s$.

If $u >_{\mathsf{ep}} t$ and $t >_{\mathsf{ep}} s$ are both derived by E3, then $\psi = \xi = \zeta \in \Sigma$, $\bar{u} \gg^{\xi}_{\mathsf{ep}} \bar{t}$, $\bar{t} \gg^{\zeta}_{\mathsf{ep}} \bar{s}$, and either $m = 0$ or $t >_{\mathsf{ep}} \mathit{chop}(s)$. By the induction hypothesis and by preservation of transitivity (property X3) on the set consisting of the elements of $\bar{u}$, $\bar{t}$ and $\bar{s}$, it follows that $\bar{u} \gg^{\zeta}_{\mathsf{ep}} \bar{s}$. If $m = 0$, we obtain $u >_{\mathsf{ep}} s$ directly by E3. If $m > 0$, we have $t >_{\mathsf{ep}} \mathit{chop}(s)$ and by applying the induction hypothesis to u, t, $\mathit{chop}(s)$, it follows that $u >_{\mathsf{ep}} \mathit{chop}(s)$. By E3, we have $u >_{\mathsf{ep}} s$.

If $u >_{\mathsf{ep}} t$ and $t >_{\mathsf{ep}} s$ are both derived by E4, then $\psi = \xi = \zeta \in \Sigma$, $\bar{u} \gg^{\mathsf{f}}_{\mathsf{ep}} \bar{t}$, $\bar{t} \gg^{\mathsf{f}}_{\mathsf{ep}} \bar{s}$ for all $\mathsf{f} \in \Sigma$, $r > 0$, $n > 0$, $\mathit{chop}(u) >_{\mathsf{ep}} \mathit{chop}(t)$, and either $m = 0$ or $\mathit{chop}(t) >_{\mathsf{ep}} \mathit{chop}(s)$. As above, by the induction hypothesis and by preservation of transitivity (property X3) on the set consisting of the elements of $\bar{u}$, $\bar{t}$ and $\bar{s}$, it follows that $\bar{u} \gg^{\mathsf{f}}_{\mathsf{ep}} \bar{s}$ for all $\mathsf{f} \in \Sigma$. If $m = 0$, we obtain $u >_{\mathsf{ep}} s$ directly by E4. If $m > 0$, we have $\mathit{chop}(u) >_{\mathsf{ep}} \mathit{chop}(t) >_{\mathsf{ep}} \mathit{chop}(s)$. By applying the induction hypothesis to $\mathit{chop}(u)$, $\mathit{chop}(t)$, $\mathit{chop}(s)$, it follows that $\mathit{chop}(u) >_{\mathsf{ep}} \mathit{chop}(s)$. By E4, we have $u >_{\mathsf{ep}} s$.

If one of the inequalities $u >_{\mathsf{ep}} t$ and $t >_{\mathsf{ep}} s$ is derived by E2 or E3, the other cannot be derived by E4 because ξ must be either a variable or a symbol. ◀

▶ **Theorem 6** (Irreflexivity). *$s \not>_{\mathsf{ep}} s$.*

Proof. By strong induction on $|s|$. We suppose that $s >_{\mathsf{ep}} s$ and derive a contradiction. Let $s = \zeta\ \bar{s}_m$.

If $s >_{\mathsf{ep}} s$ is derived by E1, then $m > 0$ and $\mathit{chop}(s) \geq_{\mathsf{ep}} s$. From the definition of chop, it is clear that $\mathit{chop}(s) \neq s$. Hence, $\mathit{chop}(s) >_{\mathsf{ep}} s$. By E1, we have $s >_{\mathsf{ep}} \mathit{chop}(s)$. By transitivity (Theorem 5), it follows that $\mathit{chop}(s) >_{\mathsf{ep}} \mathit{chop}(s)$, which contradicts the induction hypothesis.

If $s >_{\mathsf{ep}} s$ is derived by E2, we have $\zeta \succ \zeta$, in contradiction to $\succ$ being a total order.

If $s >_{\mathsf{ep}} s$ is derived by E3 or E4, we have $\bar{s} \gg^{\mathsf{f}}_{\mathsf{ep}} \bar{s}$ for some $\mathsf{f} \in \Sigma$. By preservation of irreflexivity (property X4) on the set consisting of the elements of $\bar{s}$ and by transitivity of $>_{\mathsf{ep}}$ (Theorem 5), it follows that $s' >_{\mathsf{ep}} s'$ for some $s' \in \bar{s}$. This contradicts the induction hypothesis. ◀

▶ **Lemma 7** (Embedding Step Property). *$t \longrightarrow_{\mathsf{emb}} s$ implies $t >_{\mathsf{ep}} s$.*

Proof. By strong induction on $|s| + |t|$. Let $s = \zeta\ \bar{s}_m$ and $t = \xi\ \bar{t}_n$. We distinguish the following three cases, depending on the position of the embedding $t \longrightarrow_{\mathsf{emb}} s$.

- *Generalized Chop:* right-embedding at a position left^j for some $0 \leq j < n$,
  i.e., $s = t_i\ t_{i+1}\ \ldots\ t_n$ where $i = n - j$.
- *Remove Argument:* left-embedding at a position left^j for some $0 \leq j < n$,
  i.e., $s = \xi\ t_1\ \ldots\ t_{i-1}\ t_{i+1}\ \ldots\ t_n$ where $i = n - j$.
- *Reduce Argument:* embeddings at other positions,
  i.e., $s = \xi\ t_1\ \ldots\ t_{i-1}\ u\ t_{i+1}\ \ldots\ t_n$ for some i and some u such that $t_i \longrightarrow_{\mathsf{emb}} u$.

Case 1 (Generalized Chop): Then $s = t_i\ t_{i+1}\ \ldots\ t_n$ for some i. If $i = 1$, then $s = \mathit{chop}(t)$, which implies $t >_{\mathsf{ep}} s$ by E1. If $i > 1$, there is a right-embedding at position left^{n-i} from $\mathit{chop}(t)$ to s. By the induction hypothesis, $\mathit{chop}(t) >_{\mathsf{ep}} s$ and hence $t >_{\mathsf{ep}} s$ by E1.

Case 2 (Remove Argument): Then $\zeta = \xi$ and $\bar{s}_m = t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n$. Depending on whether $\zeta \in \mathcal{V}$ or $\zeta \in \Sigma$, we will use E3 or E4 to show $t >_{\mathsf{ep}} s$. To apply either condition, we show $\bar{t}_n \gg^{\mathsf{f}}_{\mathsf{ep}} \bar{s}_m$ for all symbols $\mathsf{f} \in \Sigma$. Since $\bar{s}_m = t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n$, this follows from properties X8, X9, and X10. If $m = 0$, we can apply E3 or E4 directly to obtain $t >_{\mathsf{ep}} s$.

Otherwise, both m and n are non-zero and we show that $\mathit{chop}(t) \longrightarrow_{\mathsf{emb}} \mathit{chop}(s)$ by a case distinction on whether $i = 1$ or $i > 1$ as follows. If $i = 1$, then $\mathit{chop}(t) = t_1\ t_2\ \ldots\ t_n$ and $\mathit{chop}(s) = t_2\ t_3\ \ldots\ t_n$. Hence, there is a right-embedding step at position left^{n-2} from $\mathit{chop}(t)$

to $\mathit{chop}(s)$. If $i > 1$, then $\mathit{chop}(t) = t_1\ t_2\ \ldots\ t_n$ and $\mathit{chop}(s) = t_1\ t_2\ \ldots\ t_{i-1}\ t_{i+1}\ \ldots\ t_n$. Hence, there is a left-embedding step at position left^{n-i} from $\mathit{chop}(t)$ to $\mathit{chop}(s)$.

By the induction hypothesis, $\mathit{chop}(t) >_{\mathsf{ep}} \mathit{chop}(s)$. If $\zeta \in \mathcal{V}$, we can then apply E4 to obtain $t >_{\mathsf{ep}} s$. Otherwise, $\zeta \in \Sigma$, and we apply E1 to obtain $t >_{\mathsf{ep}} \mathit{chop}(s)$ and E3 to obtain $t >_{\mathsf{ep}} s$.

Case 3 (Reduce Argument): Then $\zeta = \xi$ and $\bar{s} = t_1, \ldots, t_{i-1}, u, t_{i+1}, \ldots, t_n$ for some i and some u such that $t_i \longrightarrow_{\mathsf{emb}} u$. As above, depending on whether $\zeta \in \mathcal{V}$ or $\zeta \in \Sigma$, we will use E3 or E4 to show $t >_{\mathsf{ep}} s$.

To apply either condition, we show $\bar{t}_n \gg^{\mathsf{f}}_{\mathsf{ep}} \bar{s}_m$ for all symbols $\mathsf{f} \in \Sigma$. By the induction hypothesis, $t_i \longrightarrow_{\mathsf{emb}} u$ implies $t_i >_{\mathsf{ep}} u$. By property X6, we have $\bar{t}_n \gg^{\mathsf{f}}_{\mathsf{ep}} \bar{s}_m$ for all $\mathsf{f} \in \Sigma$.

Both m and n are non-zero because $0 < i \leq m = n$. We observe that $\mathit{chop}(t) \longrightarrow_{\mathsf{emb}} \mathit{chop}(s)$ because the only difference between $\mathit{chop}(t)$ and $\mathit{chop}(s)$ is that $\mathit{chop}(t)$ has the subterm t_i where $\mathit{chop}(s)$ has the subterm u and we have $t_i \longrightarrow_{\mathsf{emb}} u$. By the induction hypothesis, it follows that $\mathit{chop}(t) >_{\mathsf{ep}} \mathit{chop}(s)$. If $\zeta \in \mathcal{V}$, we apply E4 to obtain $t >_{\mathsf{ep}} s$. Otherwise, $\zeta \in \Sigma$, and we apply E1 to obtain $t >_{\mathsf{ep}} \mathit{chop}(s)$ and E3 to obtain $t >_{\mathsf{ep}} s$. ◀

▶ **Theorem 8** (Embedding Property). *$t \trianglerighteq_{\mathsf{emb}} s$ implies $t \geq_{\mathsf{ep}} s$.*

Proof. Follows by induction on $t \trianglerighteq_{\mathsf{emb}} s$ from Lemma 7 and Theorem 5. ◀

▶ **Theorem 9** (Subterm Property). *For all subterms s of a term t, we have $t \geq_{\mathsf{ep}} s$.*

Proof. Follows directly from Theorem 8. ◀

▶ **Lemma 10** (Compatibility with Functions). *If $v >_{\mathsf{ep}} u$, then $s\ v >_{\mathsf{ep}} s\ u$.*

Proof. By induction on $|s|$.

Let $s = \zeta\ \bar{s}$. Depending on whether $\zeta \in \Sigma$ or $\zeta \in \mathcal{V}$, we show $s\ v >_{\mathsf{ep}} s\ u$ by applying E3 or E4. By compatibility with tuple contexts (property X6), $v >_{\mathsf{ep}} u$ implies $\bar{s} \cdot v \gg^{\mathsf{f}}_{\mathsf{ep}} \bar{s} \cdot u$ for all $\mathsf{f} \in \Sigma$. Obviously, the tuples $\bar{s} \cdot v$ and $\bar{s} \cdot u$ are not empty. So it remains to show $s\ v >_{\mathsf{ep}} \mathit{chop}(s\ u)$ if $\zeta \in \Sigma$ or $\mathit{chop}(s\ v) >_{\mathsf{ep}} \mathit{chop}(s\ u)$ if $\zeta \in \mathcal{V}$. By E1, it suffices to show $\mathit{chop}(s\ v) >_{\mathsf{ep}} \mathit{chop}(s\ u)$ in both cases.

If $\bar{s} = ()$, then $\mathit{chop}(s\ v) = v >_{\mathsf{ep}} u = \mathit{chop}(s\ u)$ by assumption. Otherwise, $\mathit{chop}(s\ v) = \mathit{chop}(s)\ v >_{\mathsf{ep}} \mathit{chop}(s)\ u = \mathit{chop}(s\ u)$ by the induction hypothesis. ◀

▶ **Lemma 11.** *If $t >_{\mathsf{ep}} s$ and $v \geq_{\mathsf{ep}} u$, then $t\ v >_{\mathsf{ep}} s\ u$.*

Proof. By induction on $|t| + |s|$ and a case distinction on how $t >_{\mathsf{ep}} s$ is derived. Let $t = \xi\ \bar{t}_n$ and $s = \zeta\ \bar{s}_m$.

If $t >_{\mathsf{ep}} s$ is derived by E1, then $\mathit{chop}(t) \geq_{\mathsf{ep}} s$. By E1, $t\ v >_{\mathsf{ep}} \mathit{chop}(t\ v) = \mathit{chop}(t)\ v$. So it suffices to show $\mathit{chop}(t)\ v \geq_{\mathsf{ep}} s\ u$. If $\mathit{chop}(t) = s$, this follows from Lemma 10. Otherwise, we have $\mathit{chop}(t) >_{\mathsf{ep}} s$ and hence $\mathit{chop}(t)\ v >_{\mathsf{ep}} s\ u$ holds by the induction hypothesis.

If $t >_{\mathsf{ep}} s$ is derived by E2, then $\xi \succ \zeta$ and either $m = 0$ or $t >_{\mathsf{ep}} \mathit{chop}(s)$. To derive $t\ v >_{\mathsf{ep}} s\ u$ using E2, it remains to show $t\ v >_{\mathsf{ep}} \mathit{chop}(s\ u)$. If $m = 0$, then $\mathit{chop}(s\ u) = u$. Therefore, by the subterm property (Theorem 9), $t\ v >_{\mathsf{ep}} v \geq_{\mathsf{ep}} u = \mathit{chop}(s\ u)$. If $m > 0$, then $t >_{\mathsf{ep}} \mathit{chop}(s)$, and hence by the induction hypothesis, $t\ v >_{\mathsf{ep}} \mathit{chop}(s)\ u = \mathit{chop}(s\ u)$.

If $t >_{\mathsf{ep}} s$ is derived by E3 or E4, we need to show that $\bar{t}_n \gg^{\mathsf{f}}_{\mathsf{ep}} \bar{s}_m$ implies $\bar{t}_n \cdot v \gg^{\mathsf{f}}_{\mathsf{ep}} \bar{s}_m \cdot u$ for all $\mathsf{f} \in \Sigma$. We have $\bar{t}_n \cdot v \gg^{\mathsf{f}}_{\mathsf{ep}} \bar{s}_m \cdot v$ by compatibility with appending (property X9). If $v = u$, we are done. Otherwise, since $\bar{s}_m \cdot v \gg^{\mathsf{f}}_{\mathsf{ep}} \bar{s}_m \cdot u$ by compatibility with tuple contexts (property X6), it follows that $\bar{t}_n \cdot v \gg^{\mathsf{f}}_{\mathsf{ep}} \bar{s}_m \cdot u$ by preservation of transitivity (property X3) and transitivity of $>_{\mathsf{ep}}$ (Theorem 5).

If $t >_{\mathsf{ep}} s$ is derived by E3, we can apply E3 to derive $t\ v >_{\mathsf{ep}} s\ u$. The condition $t\ v >_{\mathsf{ep}} \mathit{chop}(s\ u)$ can be shown as we did for E2 above.

If $t >_{\mathsf{ep}} s$ is derived by E4, we can apply E4 to derive $t\ v >_{\mathsf{ep}} s\ u$. The proof for the condition $\mathit{chop}(t\ v) >_{\mathsf{ep}} \mathit{chop}(s\ u)$ is similar to the argument made for E2 above. ◀

▶ **Theorem 12** (Compatibility with Contexts). *If $t >_{\mathsf{ep}} s$, then $u\ (t\ \bar{v}) >_{\mathsf{ep}} u\ (s\ \bar{v})$.*

Proof. By repeatedly applying Lemma 11 and finally Lemma 10. ◀

▶ **Theorem 13** (Stability under Substitutions). *If $t >_{\mathsf{ep}} s$, then $t\sigma >_{\mathsf{ep}} s\sigma$.*

Proof. By well-founded induction on the multiset $\{|t|, |s|\}$ with respect to the multiset extension [18] of $>$ on $\mathbb{N}$, followed by a case distinction on how $t >_{\mathsf{ep}} s$ is derived. Let $t = \xi\ \bar{t}_n$ and $s = \zeta\ \bar{s}_m$.

If $t >_{\mathsf{ep}} s$ is derived by E1, then $\mathit{chop}(t) \geq_{\mathsf{ep}} s$. By the induction hypothesis, $\mathit{chop}(t)\sigma \geq_{\mathsf{ep}} s\sigma$. Since $t\sigma \longrightarrow_{\mathsf{emb}} \mathit{chop}(t)\sigma$, we have $t\sigma >_{\mathsf{ep}} \mathit{chop}(t)\sigma$ by the embedding property (Theorem 8). Hence, by transitivity $t\sigma >_{\mathsf{ep}} s\sigma$.

If $t >_{\mathsf{ep}} s$ is derived by E2, then $\xi, \zeta \in \Sigma$, $\xi \succ \zeta$, and either $m = 0$ or $t >_{\mathsf{ep}} \mathit{chop}(s)$. We show $t\sigma >_{\mathsf{ep}} s\sigma$ by applying E2. Since $\xi, \zeta \in \Sigma$, the head of $t\sigma$ is ξ, the head of $s\sigma$ is ζ, and the number of arguments of $s\sigma$ is m. Hence, it only remains to show that $t >_{\mathsf{ep}} \mathit{chop}(s)$ implies $t\sigma >_{\mathsf{ep}} \mathit{chop}(s\sigma)$, which follows from the induction hypothesis and from $\mathit{chop}(s)\sigma = \mathit{chop}(s\sigma)$.

If $t >_{\mathsf{ep}} s$ is derived by E3, then $\xi = \zeta \in \Sigma$, $\bar{t}_n \gg^{\zeta}_{\mathsf{ep}} \bar{s}_m$, and either $m = 0$ or $t >_{\mathsf{ep}} \mathit{chop}(s)$. Since $\xi, \zeta \in \Sigma$, the head of $t\sigma$ is ξ, the head of $s\sigma$ is ζ, and $\bar{t}_n\sigma$ and $\bar{s}_m\sigma$ are the respective argument tuples of $t\sigma$ and $s\sigma$. By the induction hypothesis and preservation of stability (property X2) on the set of elements of $\bar{t}_n$ and $\bar{s}_m$, we have $\bar{t}_n\sigma \gg^{\zeta}_{\mathsf{ep}} \bar{s}_m\sigma$. We apply E3 to show $t\sigma >_{\mathsf{ep}} s\sigma$. It remains to show that $t >_{\mathsf{ep}} \mathit{chop}(s)$ implies $t\sigma >_{\mathsf{ep}} \mathit{chop}(s\sigma)$, which follows from the induction hypothesis and from $\mathit{chop}(s)\sigma = \mathit{chop}(s\sigma)$.

If $t >_{\mathsf{ep}} s$ is derived by E4, then $\xi = \zeta \in \mathcal{V}$, $\bar{t}_n \gg^{\mathsf{f}}_{\mathsf{ep}} \bar{s}_m$ for all $\mathsf{f} \in \Sigma$, $n > 0$, and either $m = 0$ or $\mathit{chop}(t) >_{\mathsf{ep}} \mathit{chop}(s)$. We will show that $u\ (\bar{t}_n\sigma) >_{\mathsf{ep}} u\ (\bar{s}_m\sigma)$ for all u with $|u| \leq |\zeta\sigma|$. For $u = \zeta\sigma$, it then follows that $t\sigma >_{\mathsf{ep}} s\sigma$. We show this by induction on $|u|$. We will refer to this induction as the inner induction and to the induction on the multiset $\{|t|, |s|\}$ as the outer induction.

We have to show $u\ (\bar{t}_n\sigma) >_{\mathsf{ep}} u\ (\bar{s}_m\sigma)$. We apply E3 or E4 to do so, depending on whether the head of u is a symbol or a variable. We write $u = \psi\ \bar{u}_r$.

First, we show that $\bar{u}_r \cdot (\bar{t}_n\sigma) \gg^{\mathsf{f}}_{\mathsf{ep}} \bar{u}_r \cdot (\bar{s}_m\sigma)$ for all $\mathsf{f} \in \Sigma$. As above, by the outer induction hypothesis and preservation of stability (property X2) on the set of elements of $\bar{t}_n$ and $\bar{s}_m$, we have $\bar{t}_n\sigma \gg^{\mathsf{f}}_{\mathsf{ep}} \bar{s}_m\sigma$. Then $\bar{u}_r \cdot (\bar{t}_n\sigma) \gg^{\mathsf{f}}_{\mathsf{ep}} \bar{u}_r \cdot (\bar{s}_m\sigma)$ follows by compatibility with prepending (property X8).

If $m = 0$ and $r = 0$, we can apply E3 or E4 directly to show $u\ (\bar{t}_n\sigma) >_{\mathsf{ep}} u\ (\bar{s}_m\sigma)$.

If $r > 0$, then $\mathit{chop}(u\ (\bar{t}_n\sigma)) = \mathit{chop}(u)\ (\bar{t}_n\sigma) >_{\mathsf{ep}} \mathit{chop}(u)\ (\bar{s}_m\sigma) = \mathit{chop}(u\ (\bar{s}_m\sigma))$ by the inner induction hypothesis. If $\psi \in \mathcal{V}$, we can then apply E4 to obtain $u\ (\bar{t}_n\sigma) >_{\mathsf{ep}} u\ (\bar{s}_m\sigma)$. Otherwise, $\psi \in \Sigma$, and we can apply E1 to obtain $u\ (\bar{t}_n\sigma) >_{\mathsf{ep}} \mathit{chop}(u\ (\bar{s}_m\sigma))$ and then E3 to obtain $u\ (\bar{t}_n\sigma) >_{\mathsf{ep}} u\ (\bar{s}_m\sigma)$.

If $m > 0$ and $r = 0$, then $\mathit{chop}(t) >_{\mathsf{ep}} \mathit{chop}(s)$, $\mathit{chop}(u\ (\bar{t}_n\sigma)) = \mathit{chop}(t)\sigma$ and $\mathit{chop}(u\ (\bar{s}_m\sigma)) = \mathit{chop}(s)\sigma$. By the outer induction hypothesis, $\mathit{chop}(t)\sigma >_{\mathsf{ep}} \mathit{chop}(s)\sigma$, i.e., $\mathit{chop}(u\ (\bar{t}_n\sigma)) >_{\mathsf{ep}} \mathit{chop}(u\ (\bar{s}_m\sigma))$. As above, if $\psi \in \mathcal{V}$, we can then apply E4 to obtain $u\ (\bar{t}_n\sigma) >_{\mathsf{ep}} u\ (\bar{s}_m\sigma)$. Otherwise, $\psi \in \Sigma$, and we can apply E1 to obtain $u\ (\bar{t}_n\sigma) >_{\mathsf{ep}} \mathit{chop}(u\ (\bar{s}_m\sigma))$ and then E3 to obtain $u\ (\bar{t}_n\sigma) >_{\mathsf{ep}} u\ (\bar{s}_m\sigma)$.

This concludes the inner and the outer induction. ◀

▶ **Theorem 14** (Ground Totality). *For ground terms t and s, we have $t <_{\mathsf{ep}} s$, $t = s$, or $t >_{\mathsf{ep}} s$.*

Proof. By well-founded induction on the multiset $\{|t|, |s|\}$ with respect to the multiset extension [18] of $>$ on $\mathbb{N}$. Let $t = \xi\ \bar{t}_n$ and $s = \zeta\ \bar{s}_m$. Then $\xi, \zeta \in \Sigma$ because t and s are ground.

If $n > 0$ and $\mathit{chop}(t) \not<_{\mathsf{ep}} s$, then by the induction hypothesis $\mathit{chop}(t) \geq_{\mathsf{ep}} s$ and hence $t >_{\mathsf{ep}} s$ by E1. Thus we can assume that either $n = 0$ or $s >_{\mathsf{ep}} \mathit{chop}(t)$. Analogously, we can assume that either $m = 0$ or $t >_{\mathsf{ep}} \mathit{chop}(s)$.

If $\xi \succ \zeta$ or $\xi \prec \zeta$, we have $t >_{\mathsf{ep}} s$ or $t <_{\mathsf{ep}} s$ by E2. Otherwise, we have $\xi = \zeta$ by totality of $\succ$. If either $\bar{t} \gg^{\zeta}_{\mathsf{ep}} \bar{s}$ or $\bar{t} \ll^{\zeta}_{\mathsf{ep}} \bar{s}$, then we have $t >_{\mathsf{ep}} s$ or $t <_{\mathsf{ep}} s$ by E3. By the induction hypothesis and preservation of totality (property X7) on the set of elements of $\bar{s}$ and $\bar{t}$, if $\bar{t} \not\gg^{\zeta}_{\mathsf{ep}} \bar{s}$ and $\bar{t} \not\ll^{\zeta}_{\mathsf{ep}} \bar{s}$, then $\bar{t} = \bar{s}$ and hence $t = s$. ◀

▶ **Theorem 15** (Well-Foundedness). *The order $>_{\mathsf{ep}}$ is well founded.*

Proof. We suppose that there exists an infinite descending chain $s_0 >_{\mathsf{ep}} s_1 >_{\mathsf{ep}} \cdots$ and derive a contradiction. We use a minimal counterexample argument [20].

A term s is *bad* if there is an infinite descending $>_{\mathsf{ep}}$-chain from s. Other terms are *good*. 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 u with $s_i >_{\mathsf{ep}} u$.

For each i, let $U_i = \{u \mid s_i \rhd_{\mathsf{emb}} u\}$, where $\rhd_{\mathsf{emb}}$ is the irreflexive counterpart of $\trianglerighteq_{\mathsf{emb}}$. Let $U = \bigcup_i U_i$. All terms in U are good: If there existed a bad $u \in U_0$, then $|s_0| > |u|$, contradicting the minimality of s_0. If there existed a bad $u \in U_{i+1}$ for some i, then $s_i >_{\mathsf{ep}} s_{i+1} >_{\mathsf{ep}} u$ by the embedding property (Theorem 8), contradicting the minimality of s_{i+1}.

Only conditions E2, E3, and E4 can have been used to derive $s_i >_{\mathsf{ep}} s_{i+1}$. If E1 was used, then $\mathit{chop}(s_i) \geq_{\mathsf{ep}} s_{i+1} >_{\mathsf{ep}} s_{i+2}$. But then there would be an infinite descending chain $\mathit{chop}(s_i) >_{\mathsf{ep}} s_{i+2} >_{\mathsf{ep}} s_{i+3} >_{\mathsf{ep}} \cdots$ from $\mathit{chop}(s_i)$, contradicting the goodness of $\mathit{chop}(s_i) \in U$.

E2 can have been used only finitely many times in the chain since E3 and E4 preserve the head and E2 makes the head smaller with respect to the well-founded relation $\succ$. Hence, there is a number k such that the entire chain $s_k >_{\mathsf{ep}} s_{k+1} >_{\mathsf{ep}} \cdots$ has been derived by E3 and E4. Let $s_i = \zeta\ \bar{u}_i$ (where contrary to our usual convention the indices of $\bar{u}_i$ identify the tuple and do not denote its length). Then we have an infinite chain $\bar{u}_k \gg^{\mathsf{f}}_{\mathsf{ep}} \bar{u}_{k+1} \gg^{\mathsf{f}}_{\mathsf{ep}} \cdots$ for some f. All elements of these tuples are in U because each element of $\bar{u}_i$ is embedded in s_i. However, since all elements of U are good, $>_{\mathsf{ep}}$ is well founded on U. By preservation of well-foundedness (property X5), $\gg^{\mathsf{f}}_{\mathsf{ep}}$ is well founded on U^*, which contradicts the existence of the above $\gg^{\mathsf{f}}_{\mathsf{ep}}$-chain. ◀

6 Examples

The following examples illustrate the benefits of EPO for term rewriting and superposition.

▶ **Example 16.** Consider the following term rewrite system:

$$\mathsf{f}\ x\ \mathsf{Nil} \xrightarrow{1} x \qquad \mathsf{f}\ x\ (\mathsf{A}\ y) \xrightarrow{2} \mathsf{f}\ (\mathsf{A}\ (\mathsf{B}\ x))\ y \qquad \mathsf{f}\ x\ (\mathsf{B}\ y) \xrightarrow{3} \mathsf{f}\ (\mathsf{B}\ (\mathsf{A}\ x))\ y$$

This rewrite system can be interpreted as a definition of a function on strings. In this interpretation, Nil represents the empty string, and chains of applications of the functions A and B to Nil represent strings over the alphabet $\{\mathsf{A}, \mathsf{B}\}$; thus, $\mathsf{A}\ (\mathsf{B}\ (\mathsf{B}\ \mathsf{Nil}))$ represents the string ABB. The function f takes two such strings, reverses the second string, replaces in the resulting string each A by AB and each B by BA, and finally appends the first string.

All three rules are orientable by EPO with the right-to-left length-lexicographic extension for f and precedence $\mathsf{f} \succ \mathsf{A}, \mathsf{B}$. Rule 1 can clearly be oriented because of the subterm property (Theorem 9). To show that rule 2 can be oriented, we apply E3. To do so, we need to prove $(x, \mathsf{A}\ y) \gg^{\mathsf{f}}_{\mathsf{ep}} ((\mathsf{A}\ (\mathsf{B}\ x)), y)$ and $\mathsf{f}\ x\ (\mathsf{A}\ y) >_{\mathsf{ep}} \mathsf{A}\ (\mathsf{B}\ x)\ y$. The former holds by the definition of the right-to-left length-lexicographic extension and by E1. For the latter, we apply E2. To show $\mathsf{f}\ x\ (\mathsf{A}\ y) >_{\mathsf{ep}} \mathsf{B}\ x\ y$, we apply E2 again. To show $\mathsf{f}\ x\ (\mathsf{A}\ y) >_{\mathsf{ep}} x\ y$, we apply E1. To show $x\ (\mathsf{A}\ y) >_{\mathsf{ep}} x\ y$, we apply E4. Finally, $\mathsf{A}\ y >_{\mathsf{ep}} y$ holds by E1. The proof for rule 3 is analogous.

To my knowledge, the literature contains no other ground-total simplification order for λ-free higher-order terms that can orient all three of these rules. Rules 2 and 3 are not orientable by applicative KBO or applicative RPO. With applicative KBO, the weight of the right-hand sides is always too large. With applicative RPO, too many heads are the application symbol @, preventing us from finding an appropriate precedence. With λ-free KBO [3], one of the two rules 2 and 3 can be oriented by assigning either A or B zero weight, but the system as a whole is not orientable with this order either. With λ-free RPO [9], we can orient all three rules, but λ-free RPO is not a simplification order.

This example suggests that EPO with a right-to-left length-lexicographic extension is generally stronger than left-to-right. If the two arguments of f were swapped, one would intuitively attempt to use the left-to-right extension for f, but fail because $\mathsf{f}\ (\mathsf{A}\ y)\ x \not>_{\mathsf{ep}} y\ (\mathsf{A}\ (\mathsf{B}\ x))$. For this system with the arguments of f swapped, applicative RPO can orient all three rules. However, swapping arguments cannot be used as a general approach to orient rewrite systems if the affected function appears unapplied.

The term order's ability to orient equations in the right way can have considerable effects on the performance of superposition provers. The observations made in Example 16 have implications for the efficiency of the superposition calculus:

▶ **Example 17.** Consider the rewrite rules from Example 16, recast as equations, and the negated conjecture given below, for some $k \in \mathbb{N}$:

$$\mathsf{f}\ x\ \mathsf{Nil} \approx x \qquad \mathsf{f}\ x\ (\mathsf{A}\ y) \approx \mathsf{f}\ (\mathsf{A}\ (\mathsf{B}\ x))\ y \qquad \mathsf{f}\ x\ (\mathsf{B}\ y) \approx \mathsf{f}\ (\mathsf{B}\ (\mathsf{A}\ x))\ y$$
$$\mathsf{f}\ \mathsf{c}\ (\mathsf{AB})^{k+1} \not\approx \mathsf{B}\ (\mathsf{A}\ (\mathsf{f}\ \mathsf{c}\ ((\mathsf{AB})^{k}\mathsf{A})))$$

where $(\mathsf{AB})^{k+1}$ stands for $\mathsf{A}\ (\mathsf{B}\ \ldots (\mathsf{A}\ (\mathsf{B}\ \mathsf{Nil})) \ldots)$ and $(\mathsf{AB})^{k}\mathsf{A}$ for $\mathsf{A}\ (\mathsf{B}\ \ldots (\mathsf{A}\ \mathsf{Nil}) \ldots)$. Using the EPO from Example 16 that can orient the equations left to right, superposition provers can solve this problem by simplification rules only. Simplification rules are much more efficient than inference rules because simplifications replace clauses and do not add new ones. Using an order that can orient only the first equation from left to right, we would need at least k inferences; using an order that can orient the first equation and only one of the other two, we would need at least $k/2$ inferences.

7 Implementation

I implemented EPO in the Zipperposition prover. Zipperposition [16, 17] is an open source[2] superposition-based theorem prover for first- and higher-order logic written in OCaml. In previous work [6], together with colleagues I extended it with refutationally complete modes for λ-free higher order logic, also known as applicative first-order logic. We will focus on

[2] `https://github.com/sneeuwballen/zipperposition`

```
$\mathsf{epo}(\xi\ \bar{t}_n \text{ as } t,\ \zeta\ \bar{s}_m \text{ as } s) =$
    if $t = s$ then $\mathsf{Equal}$
    elif $t \in \mathcal{V}$ and $s \in \mathcal{V}$ then $\mathsf{Incomparable}$
    elif $t \in \mathcal{V}$ then (if $t$ occurs in $s$ then $\mathsf{LessThan}$ else $\mathsf{Incomparable}$)
    elif $s \in \mathcal{V}$ then (if $s$ occurs in $t$ then $\mathsf{GreaterThan}$ else $\mathsf{Incomparable}$)
    else
        if $\xi \succ \zeta$ then $\mathsf{check}_{\mathsf{E2,E3}}(t, s)$
        elif $\xi \prec \zeta$ then $\mathsf{check}^{\mathsf{inv}}_{\mathsf{E2,E3}}(t, s)$
        elif $\xi = \zeta$ and $\zeta \in \Sigma$ then
            if $\bar{t}_n \gg^{\zeta}_{\mathsf{ep}} \bar{s}_m$ then $\mathsf{check}_{\mathsf{E2,E3}}(t, s)$
            elif $\bar{t}_n \ll^{\zeta}_{\mathsf{ep}} \bar{s}_m$ then $\mathsf{check}^{\mathsf{inv}}_{\mathsf{E2,E3}}(t, s)$
            else $\mathsf{check}_{\mathsf{E1}}(t, s)$
        elif $\xi = \zeta$ and $\zeta \in \mathcal{V}$ then
            if $\bar{t}_n \gg^{\mathsf{f}}_{\mathsf{ep}} \bar{s}_m$ for all $\mathsf{f} \in \Sigma$ and $n > 0$ then $\mathsf{check}_{\mathsf{E4}}(t, s)$
            elif $\bar{t}_n \ll^{\mathsf{f}}_{\mathsf{ep}} \bar{s}_m$ for all $\mathsf{f} \in \Sigma$ and $m > 0$ then $\mathsf{check}^{\mathsf{inv}}_{\mathsf{E4}}(t, s)$
            else $\mathsf{check}_{\mathsf{E1}}(t, s)$
        else $\mathsf{check}_{\mathsf{E1}}(t, s)$

$\mathsf{check}_{\mathsf{E1}}(\xi\ \bar{t}_n \text{ as } t,\ \zeta\ \bar{s}_m \text{ as } s) =$
    if $n > 0$ and $\mathit{chop}(t) \geq_{\mathsf{ep}} s$ then $\mathsf{GreaterThan}$
    elif $m > 0$ and $t \leq_{\mathsf{ep}} \mathit{chop}(s)$ then $\mathsf{LessThan}$
    else $\mathsf{Incomparable}$

$\mathsf{check}_{\mathsf{E2,E3}}(\xi\ \bar{t}_n \text{ as } t,\ \zeta\ \bar{s}_m \text{ as } s) =$
    if $m = 0$ or $t >_{\mathsf{ep}} \mathit{chop}(s)$ then $\mathsf{GreaterThan}$ else $\mathsf{check}_{\mathsf{E1}}(t, s)$

$\mathsf{check}^{\mathsf{inv}}_{\mathsf{E2,E3}}(\xi\ \bar{t}_n \text{ as } t,\ \zeta\ \bar{s}_m \text{ as } s) =$
    if $n = 0$ or $\mathit{chop}(t) <_{\mathsf{ep}} s$ then $\mathsf{LessThan}$ else $\mathsf{check}_{\mathsf{E1}}(t, s)$

$\mathsf{check}_{\mathsf{E4}}(\xi\ \bar{t}_n \text{ as } t,\ \zeta\ \bar{s}_m \text{ as } s) =$
    if $m = 0$ or $\mathit{chop}(t) >_{\mathsf{ep}} \mathit{chop}(s)$ then $\mathsf{GreaterThan}$ else $\mathsf{check}_{\mathsf{E1}}(t, s)$

$\mathsf{check}^{\mathsf{inv}}_{\mathsf{E4}}(\xi\ \bar{t}_n \text{ as } t,\ \zeta\ \bar{s}_m \text{ as } s) =$
    if $n = 0$ or $\mathit{chop}(t) <_{\mathsf{ep}} \mathit{chop}(s)$ then $\mathsf{LessThan}$ else $\mathsf{check}_{\mathsf{E1}}(t, s)$
```

Figure 1 Pseudocode of the EPO implementation

the mode that performed best in the evaluation of that paper, the "nonpurifying intensional variant". It is designed to deal with orders that do not have compatibility with arguments, such as λ-free RPO, but falls back to a simpler calculus with orders that have full compatibility with contexts, such as λ-free KBO or EPO.

The pseudocode of the implementation of EPO is given in Figure 1. As usual in superposition provers, the procedure compares two terms in both directions, yielding one of the answers GreaterThan, Equal, LessThan, or Incomparable. When the pseudocode refers to $>_{\mathsf{ep}}$, $\geq_{\mathsf{ep}}$, and $\gg^{\mathsf{f}}_{\mathsf{ep}}$, this is to be interpreted in terms of the function epo. The syntax '$\xi\ \bar{t}_n$ as t' in the arguments of function definitions means that t denotes the entire term, ξ denotes its head, and $\bar{t}_n$ denotes its arguments. Zipperposition's terms use hash consing, allowing for fast equality checks of terms.

It is crucial to the performance of this implementation to use a cache on the function epo. For example, to compute that $\mathsf{f}^m\ x \not\lessgtr_{\mathsf{ep}} \mathsf{f}^n\ y$ for $m \leq n$, we need at least 4^m calls to epo if the cache is inactive. With a cache however, only $(m+1)(n+1)$ of these calls to epo have to be computed; the other return values can be found in the cache. More generally, the following lemma holds:

▶ **Lemma 18.** *To calculate the order of two terms t and s, the pseudocode in Figure 1 needs at most* $\mathrm{depth}(t) \cdot \mathrm{depth}(s) \cdot |t| \cdot |s|$ *distinct calls to* epo*. Here, the depth of a term $\zeta\ \bar{u}_m$ is* 1 *if $m = 0$ and* $\max_{u \in \bar{u}}(\mathrm{depth}(u)) + 1$ *otherwise.*

Proof. We define a set S_t that overapproximates the set of all embeddings of t that may be involved in computing the order of t with some other term.

To this end, let $\rhd_{\mathsf{arg}}$ be the relation defined by $\zeta\ \bar{u}_n \rhd_{\mathsf{arg}} u_i$ for all terms $\zeta\ \bar{u}_n$ and all i. Let $\rhd_{\mathsf{chop}}$ be the relation defined by $\zeta\ \bar{u}_n \rhd_{\mathsf{chop}} \mathit{chop}(\zeta\ \bar{u}_n)$ for all terms $\zeta\ \bar{u}_n$ with $n > 0$. Finally, let S_t be the set of all terms u such that $t\ (\rhd_{\mathsf{arg}} \cup \rhd_{\mathsf{chop}})^*\ u$. In other words, S_t is inductively defined as follows: Let $t \in S_t$. For any term $\zeta\ \bar{u}_n \in S_t$, let $\mathit{chop}(\zeta\ \bar{u}_n) \in S_t$ and $u_i \in S_t$ for all i.

Inspecting the pseudocode, it is obvious that S_t and S_s together overapproximate all terms that are involved in computing the order for the two terms t and s.

In a derivation of $(\rhd_{\mathsf{arg}} \cup \rhd_{\mathsf{chop}})^*$, any $\rhd_{\mathsf{chop}}$ step before a $\rhd_{\mathsf{arg}}$ step can be eliminated. More precisely, we show that $(\rhd_{\mathsf{arg}} \cup \rhd_{\mathsf{chop}})^* = (\rhd_{\mathsf{arg}}^* \circ \rhd_{\mathsf{chop}}^*)$ by proving that $(\rhd_{\mathsf{chop}} \circ \rhd_{\mathsf{arg}}) \subseteq (\rhd_{\mathsf{arg}}^*)$. We assume that $w \rhd_{\mathsf{chop}} v \rhd_{\mathsf{arg}} u$ for some terms w, v, and u. Let $w = \zeta\ \bar{w}_n$. Then $v = \mathit{chop}(\zeta\ \bar{w}_n) = w_1\ w_2\ \ldots\ w_n$. Let $w_1 = \xi\ \bar{v}_n$. Then $v = \xi\ \bar{v}_n\ w_2\ \ldots\ w_n$. Hence $u \in \bar{v}_n$ or $u \in \{w_2, \ldots, w_n\}$. In the first case, we have $w \rhd_{\mathsf{arg}} w_1 \rhd_{\mathsf{arg}} u$; In the second case $w \rhd_{\mathsf{arg}} u$. Either way, $w\ (\rhd_{\mathsf{arg}}^*)\ u$, which is what we needed to show.

Hence, S_t is the set of all terms v such that $t\ (\rhd_{\mathsf{arg}}^* \circ \rhd_{\mathsf{chop}}^*)\ v$. Therefore, we can overapproximate the size of S_t as follows:

$$|S_t| \leq \sum_{t \rhd_{\mathsf{arg}}^* u}^{u} |\{v \mid u \rhd_{\mathsf{chop}}^* v\}| \leq \sum_{t \rhd_{\mathsf{arg}}^* u}^{u} |u| \leq \mathrm{depth}(t) \cdot |t|$$

The last inequality holds because for any number of steps k,

$$\sum_{t \rhd_{\mathsf{arg}}^k u}^{u} |u| \leq |t|$$

and the number of $\rhd_{\mathsf{arg}}$ steps from t is bounded by $\mathrm{depth}(t)$.

Since S_t and S_s together overapproximate all terms that are involved in computing the order for the two terms t and s, we can overapproximate the number of distinct calls to epo by $|S_t \times S_s| = |S_t| \cdot |S_s| \leq \mathrm{depth}(t) \cdot |t| \cdot \mathrm{depth}(s) \cdot |s|$ ◀

We can use this lemma to derive the computational complexity of epo. The following theorem is stated only for the length-lexicographic extension operators since other extension operators may have a higher computational complexity.

▶ **Theorem 19.** *For each* $\mathsf{f} \in \Sigma$*, let* $> \mapsto \gg^{\mathsf{f}}$ *be either the left-to-right or the right-to-left length-lexicographic extension operator. For terms t and s, the computational complexity of* $\mathsf{epo}(t, s)$ *as given in Figure 1 is* $\mathrm{O}(\mathrm{depth}(t) \cdot \mathrm{depth}(s) \cdot |t| \cdot |s| \cdot \max(|t|, |s|))$ *if recursive calls are cached.*

Proof. Let $R(t, s)$ be the set of term pairs (v, u), for which $\mathsf{epo}(t, s)$ triggers directly or indirectly a call to $\mathsf{epo}(v, u)$. Let $C(v, u)$ be the complexity of $\mathsf{epo}(u, v)$ assuming constant time for all recursive calls. Then the computational complexity of $\mathsf{epo}(t, s)$ is

$$\mathrm{O}\left(\sum_{(v,u) \in R(t,s)} C(v, u)\right) \qquad (*)$$

We assume constant time for the recursive calls in the definition of $C(v, u)$ because each recursive call is either the first one for this argument pair and therefore counted by another summand of the sum above, or it is not the first one for this argument pair and can therefore be retrieved from the cache in constant time.

To determine $C(v, u)$, we analyze the implementation in Figure 1, assuming that all recursive calls are $\mathrm{O}(1)$. Searching for occurrences of a given variable in a term, computing *chop*, counting the number of arguments of a term, and iterating through the arguments for the length-lexicographic comparison are $\mathrm{O}(\max(|v|, |u|))$. All other operations are $\mathrm{O}(1)$. Hence, $C(v, u)$ is $\mathrm{O}(\max(|v|, |u|))$. Since the term sizes do not increase in recursive calls, $C(v, u)$ is also $\mathrm{O}(\max(|t|, |s|))$ for all $(v, u) \in R(t, s)$. By Lemma 18, $|R(t, s)| \leq \mathrm{depth}(t) \cdot \mathrm{depth}(s) \cdot |t| \cdot |s|$. Hence, by (*), the computational complexity of $\mathsf{epo}(t, s)$ is $\mathrm{O}(\mathrm{depth}(t) \cdot \mathrm{depth}(s) \cdot |t| \cdot |s| \cdot \max(|t|, |s|))$. ◀

Compared with first-order KBO or RPO, this is rather slow. Löchner [30, 31] showed that, with a lexicographic extension, KBO can be computed in $\mathrm{O}(|t| + |s|)$ and RPO in $\mathrm{O}(|t| \cdot |s|)$. RPO can be implemented so efficiently because the computation of the lexicographic order of the arguments, i.e., computing $\bar{t}_n \gg^{\zeta}_{\mathsf{ep}} \bar{s}_m$, can be merged with testing other conditions, i.e., the condition corresponding to $\mathsf{check}_{\mathsf{E2,E3}}(t, s)$. It is an open question whether a similar optimization is possible for EPO, although it is definitely not as straight-forward as for RPO.

8 Evaluation

The following evaluation compares the implementation of EPO with other orders in Zipperposition. It was performed with a CPU time limit of 300 s on StarExec nodes equipped with Intel Xeon E5-2609 0 CPUs clocked at 2.40 GHz. The raw evaluation results are available online and reproducible.[3]

From the TPTP [37], 665 higher-order problems in THF format were used, containing both monomorphic and polymorphic problems and excluding problems that contain arithmetic, tuples, the `$distinct` predicate, or the `$ite` symbol, as well as problems whose clausal normal form falls outside the λ-free fragment.

The Sledgehammer (SH) benchmarks, corresponding to Isabelle's Judgment Day suite [14], were regenerated to target λ-free higher-order logic, encoding λ-expressions as λ-lifted supercombinators [32]. The SH benchmarks comprise 1253 problems, each including 256 Isabelle facts.

Besides EPO, I evaluated RPO, KBO, and their applicative counterparts (appRPO, appKBO). Each of the orders were evaluated twice, once using the left-to-right length-lexicographic extension (LTR) and once using the right-to-left length-lexicographic extension (RTL) for all symbols. In principle, EPO also allows for different extension operators for different symbols, but it is unclear how to design appropriate heuristics. The calculus used for EPO, RPO, and KBO is the intensional non-purifying variant of the calculus described in my earlier work [6]. For the monotonic orders EPO and KBO, the calculus degrades to essentially first-order superposition, with the addition of an argument congruence rule that adds arguments of partially applied functions. In the case of the non-monotonic order RPO, the calculus performs additional superposition inferences into variable positions to remain complete. These inferences are known to harm performance, which is why we would generally expect a better performance with monotonic orders. To evaluate the applicative counterparts

[3] `http://matryoshka.gforge.inria.fr/pubs/epo_data/`

		LTR					RTL				
		#sat	#uns	∅tim	%ord	∅cla	#sat	#uns	∅tim	%ord	∅cla
TPTP	EPO	120	463	1.3	6.9	2155	120	462	1.2	6.8	2163
	RPO	119	472	0.3	0.9	1196	119	471	0.3	1.0	1171
	KBO	121	474	0.1	1.6	430	121	473	0.2	1.6	600
	appRPO	138	472	0.6	1.1	749	123	472	1.6	2.0	1489
	appKBO	122	476	0.1	1.9	306	122	476	0.3	2.0	462
SH	EPO	1	509	2.6	23.5	6356	1	505	3.1	23.2	6251
	RPO	1	550	1.6	4.7	7130	1	549	2.4	4.8	8612
	KBO	1	594	1.6	8.8	9206	1	590	1.3	8.7	6949
	appRPO	1	481	13.3	8.1	26346	1	462	17.9	16.3	28897
	appKBO	1	502	10.6	11.3	25236	1	502	10.9	11.6	26202

Figure 2 Evaluation

appKBO and appRPO, I apply the applicative encoding to the given problem directly after the clausal normal form transformation and use first-order KBO and RPO, respectively, on the resulting problem. The results for these last two orders are therefore to be interpreted with care because the applicative encoding also influences various unrelated heuristics in Zipperposition.

Figure 2 displays the number of problems found to be satisfiable (#sat), the number of problems found to be unsatisfiable (#uns), the average CPU time per problem (∅tim), the average percentage of the CPU time used to compute order comparisons (%ord), and the average number of clauses produced during a run (∅cla). When computing the three averages, satisfiable problems and problems that at least one of the ten configurations failed to solve within the time limit were excluded.

From first-order provers, it is well known that KBO generally outperforms RPO. In the #uns columns, we observe the same effect. In the present setting, the advantage of KBO is possibly even greater because the calculus performs inferences into variable positions with RPO. Although these additional superposition inferences are not performed when using EPO, the #uns results for EPO are worse than RPO and KBO. The %ord columns reveal that this is probably because EPO takes considerably more time to compute. I hypothesized that a second reason could be that generally more term pairs are incomparable under EPO and thus more inferences need to be performed and more clauses are produced. Although the numbers in the ∅cla column on the TPTP benchmark set confirm this hypothesis, the corresponding numbers on the SH benchmark set contradict it because on those benchmarks, EPO is actually producing the least amount of clauses.

The raw data show that despite the poor performance of RPO and EPO these orders can be put to good use in a portfolio prover. The RPO configurations can solve 16 problems that neither of the KBO configurations can solve. The EPO configurations can solve 11 problems that neither of the RPO configurations can solve, 12 problems that neither of the KBO configurations can solve, 51 problems that neither of the appRPO configurations can solve, 66 problems that neither of the appKBO configurations can solve, and 4 problems that no other configuration can solve. Most of the problems where EPO outperforms other orders are in the SH benchmark set. Overall, RPO is preferable over EPO if one is willing to face the complications of a non-monotonic order in theory and in implementation.

The direction (LTR or RTL) of the length-lexicographic extension does not have a large impact. For KBO and appKBO, this is to be expected since the lexicographic comparison comes in to play only when weights are equal. For EPO, the advantage of RTL that Example 16 suggests is not corroborated by the evaluation. Only with appRPO, LTR performs better than RTL. This might be because LTR tends to put more importance to the symbols that were at the heads of terms before the applicative encoding, yielding a better measure of the complexity of a term.

9 Discussion

I presented a ground-total simplification order for λ-free higher-order terms resembling RPO. In first-order logic, KBO generally outperforms RPO, but RPO with well-chosen parameters behaves better than KBO on many examples. In λ-free higher-order logic, the situation appears to be similar. However, RPO cannot be easily used for superposition in this logic if we want the calculus to remain complete because the natural generalization [9] lacks compatibility with contexts. EPO seems to be a good replacement to fill the role of RPO in λ-free higher-order logic if one wants to avoid the complications of non-monotonic orders. Otherwise, calculi specialized to deal with non-monotonic orderings [6, 12] are the better choice.

To explore different candidate definitions for EPO, I formalized my ideas early on in Isabelle/HOL [33]. This allowed me to keep track of changes in the definition and how they influence the properties and their proofs more easily. To find examples explaining why certain properties do not hold for some tentative definitions of EPO, Lazy SmallCheck [35] was of great help. For instance, Lazy SmallCheck found the example $x\ \mathsf{f}\ \mathsf{f} >'_{\mathsf{ep}} x\ x$ versus $\mathsf{f}\ y\ \mathsf{f}\ \mathsf{f} \not>'_{\mathsf{ep}} \mathsf{f}\ y\ (\mathsf{f}\ y)$ mentioned in Section 4, which I was unable to find with pen and paper.

In future work, I would like to investigate whether the computation of EPO can be optimized further. To put EPO to use in practice, implementing it in E prover [36] would be a good target because E's λ-free higher-order mode is designed for ground-total simplification orders and its calculus is more efficient for those than Zipperposition's by circumventing the argument congruence rule.

Another application of EPO could lie in termination of λ-free higher-order term rewriting. Together with λ-free KBO [3], it could serve as a basis for a generalization of the dependency pair framework [23] to λ-free higher-order terms, which would compete with dependency pair methods for full higher-order terms [10, 21, 27, 28]. The TPDB [22] already contains some applicative first-order benchmarks that such a generalized framework would attempt to prove termination for.

Acknowledgments I am grateful to the maintainers of StarExec for letting me use their service; to Jasmin Blanchette for his excellent supervision and for providing benchmarks; to Ahmed Bhayat, Simon Cruanes, Carsten Fuhs, and Petar Vukmirović for the stimulating discussions and the constructive feedback; to Jürgen Giesl and Femke van Raamsdonk for pointers to the literature; and to the anonymous reviewers for suggesting many improvements to this text. My research 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).

References

1 Franz Baader and Tobias Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

2 Leo Bachmair and Harald Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *J. Log. Comput.*, 4(3):217–247, 1994.

3 Heiko Becker, Jasmin Christian Blanchette, Uwe Waldmann, and Daniel Wand. A transfinite Knuth–Bendix order for lambda-free higher-order terms. In Leonardo de Moura, editor, *CADE-26*, volume 10395 of *LNCS*, pages 432–453. Springer, 2017.

4 Alexander Bentkamp. Formalization of the embedding path order for lambda-free higher-order terms. *Archive of Formal Proofs*, 2018. `http://isa-afp.org/entries/Lambda_Free_EPO.html`.

5 Alexander Bentkamp, Jasmin Blanchette, Sophie Tourret, Petar Vukmirović, and Uwe Waldmann. Superposition with lambdas. In Pascal Fontaine, editor, *CADE-27*, volume 11716 of *LNCS*, pages 55–73. Springer, 2019.

6 Alexander Bentkamp, Jasmin Christian Blanchette, Simon Cruanes, and Uwe Waldmann. Superposition for lambda-free higher-order logic. In Didier Galmiche, Stephan Schulz, and Roberto Sebastiani, editors, *IJCAR 2018*, volume 10900 of *LNCS*, pages 28–46. Springer, 2018.

7 Ahmed Bhayat and Giles Reger. A complete superposition calculus for higher-order logic. Submitted.

8 Jasmin Christian Blanchette, Uwe Waldmann, and Daniel Wand. A lambda-free higher-order recursive path order. Tech. report, `http://people.mpi-inf.mpg.de/~jblanche/lambda_free_rpo_rep.pdf`, 2016.

9 Jasmin Christian Blanchette, Uwe Waldmann, and Daniel Wand. A lambda-free higher-order recursive path order. In Javier Esparza and Andrzej S. Murawski, editors, *FOSSACS 2017*, volume 10203 of *LNCS*, pages 461–479. Springer, 2017.

10 Frédéric Blanqui. Higher-order dependency pairs. *CoRR*, abs/1804.08855, 2018.

11 Frédéric Blanqui, Jean-Pierre Jouannaud, and Albert Rubio. The computability path ordering. *Log. Meth. Comput. Sci.*, 11(4), 2015.

12 Miquel Bofill, Cristina Borralleras, Enric Rodríguez-Carbonell, and Albert Rubio. The recursive path and polynomial ordering for first-order and higher-order terms. *J. Log. Comput.*, 23(1):263–305, 2013.

13 Miquel Bofill and Albert Rubio. Paramodulation with non-monotonic orderings and simplification. *J. Autom. Reason.*, 50(1):51–98, 2013.

14 Sascha Böhme and Tobias Nipkow. Sledgehammer: Judgement Day. In Jürgen Giesl and Reiner Hähnle, editors, *IJCAR 2010*, volume 6173 of *LNCS*, pages 107–121. Springer, 2010.

15 Iliano Cervesato and Frank Pfenning. A linear spine calculus. *J. Log. Comput.*, 13(5):639–688, 2003.

16 Simon Cruanes. *Extending Superposition with Integer Arithmetic, Structural Induction, and Beyond.* Ph.D. thesis, École polytechnique, 2015.

17 Simon Cruanes. Superposition with structural induction. In Clare Dixon and Marcelo Finger, editors, *FroCoS 2017*, volume 10483 of *LNCS*, pages 172–188. Springer, 2017.

18 Nachum Dershowitz and Zohar Manna. Proving termination with multiset orderings. *Commun. ACM*, 22(8):465–476, 1979.

19 Naohi Eguchi. A lexicographic path order with slow growing derivation bounds. *Math. Log. Q.*, 55(2):212–224, 2009.

20 Maria C. F. Ferreira and Hans Zantema. Well-foundedness of term orderings. In Nachum Dershowitz and Naomi Lindenstrauss, editors, *CTRS-94*, volume 968 of *LNCS*, pages 106–123. Springer, 1994.

21 Carsten Fuhs and Cynthia Kop. A static higher-order dependency pair framework. In Luís Caires, editor, *ESOP 2019*, volume 11423 of *LNCS*, pages 752–782. Springer, 2019.

22 Jürgen Giesl, Albert Rubio, Christian Sternagel, Johannes Waldmann, and Akihisa Yamada. The termination and complexity competition. In Dirk Beyer, Marieke Huisman, Fabrice Kordon, and Bernhard Steffen, editors, *TACAS 2019*, volume 11429 of *LNCS*, pages 156–166. Springer, 2019.

23 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, and Stephan Falke. Mechanizing and improving dependency pairs. *J. Autom. Reason.*, 37(3):155–203, 2006.

24 Jean-Pierre Jouannaud and Albert Rubio. Polymorphic higher-order recursive path orderings. *J. ACM*, 54(1):2:1–2:48, 2007.

25 Richard Kennaway, Jan Willem Klop, M. Ronan Sleep, and Fer-Jan de Vries. Comparing curried and uncurried rewriting. *J. Symb. Comput.*, 21(1):15–39, 1996.

26 Cynthia Kop and Femke van Raamsdonk. A higher-order iterative path ordering. In Iliano Cervesato, Helmut Veith, and Andrei Voronkov, editors, *LPAR 2008*, volume 5330 of *LNCS*, pages 697–711. Springer, 2008.

27 K Kusakari. On proving termination of term rewriting systems with higher-order variables. *IPSJ Transactions on Programming*, 42(7):35–45, 2001.

28 Keiichirou Kusakari and Masahiko Sakai. Static dependency pair method for simply-typed term rewriting and related techniques. *IEICE Transactions*, 92-D(2):235–247, 2009.

29 Maxim Lifantsev and Leo Bachmair. An LPO-based termination ordering for higher-order terms without λ-abstraction. In Jim Grundy and Malcolm C. Newey, editors, *TPHOLs '98*, volume 1479 of *LNCS*, pages 277–293. Springer, 1998.

30 Bernd Löchner. Things to know when implementing KBO. *J. Autom. Reason.*, 36(4):289–310, 2006.

31 Bernd Löchner. Things to know when implementing LPO. *Internat. J. Artificial Intelligence Tools*, 15(1):53–80, 2006.

32 Jia Meng and Lawrence C. Paulson. Translating higher-order clauses to first-order clauses. *J. Autom. Reason.*, 40(1):35–60, 2008.

33 Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

34 Lawrence C. Paulson and Jasmin Christian Blanchette. Three years of experience with Sledgehammer, a practical link between automatic and interactive theorem provers. In Geoff Sutcliffe, Stephan Schulz, and Eugenia Ternovska, editors, *IWIL-2010*, volume 2 of *EPiC*, pages 1–11. EasyChair, 2012.

35 Jason S. Reich, Matthew Naylor, and Colin Runciman. Advances in Lazy SmallCheck. In Ralf Hinze, editor, *IFL*, volume 8241 of *LNCS*, pages 53–70. Springer, 2012.

36 Stephan Schulz, Simon Cruanes, and Petar Vukmirović. Faster, higher, stronger: E 2.3. In Pascal Fontaine, editor, *CADE-27*, volume 11716 of *LNCS*, pages 495–507. Springer, 2019.

37 G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure. From CNF to TH0, TPTP v6.4.0. *J. Autom. Reason.*, 59(4):483–502, 2017.

38 Petar Vukmirović, Jasmin Blanchette, Simon Cruanes, and Stephan Schulz. Extending a brainiac prover to lambda-free higher-order logic. In Tomás Vojnar and Lijun Zhang, editors, *TACAS 2019*, volume 11427 of *LNCS*, pages 192–210. Springer, 2019.

39 Hans Zantema. Termination. In Marc Bezem, Jan Willem Klop, and Roel de Vrijer, editors, *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*, pages 181–259. Cambridge University Press, 2003.