TERM ORDERS FOR OPTIMISTIC LAMBDA-SUPERPOSITION

ALEXANDER BENTKAMP [a], JASMIN BLANCHETTE [a], AND MATTHIAS HETZENBERGER [b]

[a] Ludwig-Maximilians-Universität München, Geschwister-Scholl-Platz 1, 80539 München, Germany
e-mail address: a.bentkamp@ifi.lmu.de,jasmin.blanchette@ifi.lmu.de

[b] TU Wien Informatics, Favoritenstraße 9–11, 1040 Vienna, Austria
e-mail address: matthias.hetzenberger@tuwien.ac.at

Abstract. We introduce λKBO and λLPO, two variants of the Knuth–Bendix order (KBO) and the lexicographic path order (LPO) designed for use with the λ-superposition calculus. We establish the desired properties via encodings into the familiar first-order KBO and LPO.

1. Introduction

The λ-superposition calculus, by Bentkamp et al. [3], is a highly competitive [12] approach for proving higher-order problems automatically. It works by saturation, performing inferences between available clauses until the empty clause $\perp$ is derived. A clause consists of literals, which are predicates (often equality $\approx$) applied to arguments. Terms are equivalence classes modulo the α-, β-, and η-conversions of the λ-calculus. Thus, f, $\lambda x.\, \mathsf{f}\; x$, and $(\lambda y.\, y)\; \mathsf{f}$ are all considered syntactically equal.

To break symmetries in the search space, λ-superposition uses an order $\succ$ on the terms. The stronger the order, the fewer clauses need to be generated to saturate the clause set. Yet the *derived higher-order orders* used by the only implementation of λ-superposition [3, Sect. 3] are a crude encoding in terms of a standard term order, whether the Knuth–Bendix order (KBO) or the lexicographic path order (LPO) [13]. It is very weak in the presence of applied variables; for example, it cannot orient the terms $y\; \mathsf{b}$ and $y\; \mathsf{a}$, even with the precedence $\mathsf{b} > \mathsf{a}$.

In this work, we introduce two stronger orders, called λKBO and λLPO. As the names suggest, λKBO and λLPO are variants of KBO and LPO, which are the most widely used orders with superposition calculi. KBO compares terms by first comparing their syntactic weight, resorting to a lexicographic comparison as a tiebreaker. LPO essentially performs a lexicographic comparison while ensuring the subterm property (i.e., the property that a term is larger than its proper subterms). We define three versions of λKBO and λLPO, of increasing expressiveness: for ground (i.e., closed) terms, monomorphic nonground (i.e., open) terms, and polymorphic nonground terms.

The ground orders (Sect. 3) form the first level of a development by stepwise refinement. The monomorphic orders (Sect. 4) add support for term variables; their properties are justified in terms of the ground level. Similarly, the polymorphic orders (Sect. 5) add support

Preprint submitted to
Logical Methods in Computer Science

for type variables on top of the monomorphic level. For λKBO, weights are computed as polynomials over indeterminates whose values depend on the variables occurring in the terms. These polynomials can be compared symbolically.

Both orders are specified as a strict relation $\succ$ and a nonstrict relation $\succsim$, along the lines of Sternagel and Thiemann [11]. The nonstrict orders make the comparison $y\,\mathsf{b} \succsim y\,\mathsf{a}$ possible if $\mathsf{b} > \mathsf{a}$. In this example, a strict comparison would fail because y could be instantiated by $\lambda x.\,\mathsf{c}$, which ignores the argument and makes both terms equal.

A requirement imposed by the *optimistic λ-superposition* calculus, for which the two orders are specifically designed, is that the order must ensure $u \succ u\,\mathsf{diff}\langle\tau,\upsilon\rangle(s,t)$ for a dedicated Skolem symbol diff, for all ground terms s,t,u, and for all ground types τ and υ. This allows optimistic λ-superposition to provide special support for the functional extensionality axiom

$$z\,(\mathsf{diff}\langle\alpha,\beta\rangle(\lambda\,z\,0,\lambda\,y\,0)) \not\approx y\,(\mathsf{diff}\langle\alpha,\beta\rangle(\lambda\,z\,0,\lambda\,y\,0)) \lor (\lambda\,z\,0) \approx (\lambda\,y\,0) \qquad (\textsc{Ext})$$

Notice that the two arguments of the Skolem symbol diff are specified in parentheses, as mandatory arguments or parameters.

2. Preliminaries

We use the notation $\bar{x}_n$ or $\bar{x}$ for tuples or lists $x_1,\ldots,x_n$ of length $|\bar{x}| = n \geq 0$. Applying a unary function f to such a tuple applies it pointwise: $f(\bar{x}_n) = (f(x_1),\ldots,f(x_n))$.

We write $\mathbb{N}$ for the set of natural numbers starting with 0 and $\mathbb{N}_{>0}$ for $\mathbb{N}\setminus\{0\}$. We write $\mathbf{O}$ for the set of ordinals below ϵ_0 and $\mathbf{O}_{>0}$ for $\mathbf{O}\setminus\{0\}$.

2.1. **Terms.** We will need both untyped first-order and typed higher-order terms:

- Given an untyped first-order signature Σ, we write $\mathcal{T}(\Sigma, X)$ for the set of arity-respecting terms built using symbols from Σ and the variables X—the (Σ-)*terms*. A first-order term t is *ground* if it contains no variables, or equivalently if $t \in \mathcal{T}(\Sigma,\emptyset)$.
- For the higher order, the types and terms are those of polymorphic higher-order logic, as defined in Bentkamp et al. [3], but with a few specificities noted below.

A higher-order signature $(\Sigma_{\mathsf{ty}},\Sigma)$ consists of a type signature Σ_{ty} and a term signature Σ, which depends on Σ_{ty}. With each type constructor $\kappa \in \Sigma_{\mathsf{ty}}$ is associated an arity—the number of arguments it takes. The set of types $\mathcal{T}\!y(\Sigma_{\mathsf{ty}}, X_{\mathsf{ty}})$ over X_{ty} is built using variables from X_{ty} and type constructors applied to the expected number of arguments. The functional type constructor $\to$ is distinguished. We abbreviate $\tau_1 \to \cdots \to \tau_n \to \upsilon$ to $\bar{\tau}_n \to \upsilon$.

The first departure from Bentkamp et al. is that we find it convenient to represent λ-terms using a locally nameless notation [6] based on De Bruijn indices [5]. This notation is essentially isomorphic to a nominal notation, with α-equivalence built in. For example, $\lambda x.\,\lambda y.\,x$ will be represented as $\lambda\,\lambda\,1$, where the De Bruijn index 1 is a "nameless dummy." We allow leaking De Bruijn indices—indices that point beyond all λ-binders—but these will be ignored by substitutions. The operator $t{\uparrow}^n$ shifts all leaking De Bruijn indices by n; if omitted, $n = 1$.

The second departure from Bentkamp et al. is that we will use the η-long β-normal form as representatives for βη-equivalence classes, or "*terms*," where they used the η-short β-normal form. The main advantage of η-long is that it makes it possible to obtain the desired maximality result for the functional extensionality axiom. Moreover, since with η-long terms of function type are always λ-abstractions, we will find that this simplifies the arithmetic when defining the λKBO. Given a λ-term t, we will denote its η-long β-normal form as $t\!\uparrow_\eta$.

The main disadvantage of the η-long β-normal form arises with polymorphism: Instantiating a type variable with a functional type can result in an η-expansion, dramatically changing the term's shape. For example, if $z : \alpha$, then $z\{\alpha \mapsto (\kappa \to \kappa)\} = \lambda\, z\, 0$. This can be accounted for by the orders, but only at the cost of some weakening.

The third departure from Bentkamp et al. is that the symbols (also called constants) may take parameters, passed in parentheses, in addition to their regular curried arguments. These parameters do not count as subterms. This mechanism is used for diff. Parameters are supported by the optimistic λ-superposition calculus.

We write $t : \tau$ to indicate that t has type τ. With each symbol $\mathsf{f} \in \Sigma$ is associated a typing $\Pi\bar{\alpha}_m.\ \bar{\tau}_n \Rightarrow \upsilon$, where $\bar{\alpha}_m$ is a tuple of distinct variables that contains all type variables from $\bar{\tau}_n$ and υ, $\bar{\tau}_n$ is the tuple of parameter types, and υ is the (possibly functional) body type. Given f, we let $\mathit{tyarity}(\mathsf{f}) = m$ and $\mathit{arity}(\mathsf{f}) = n$. We specify a type instance by specifying a tuple $\bar{\alpha}_m\sigma$ of types in angle brackets corresponding to the type arguments: $\mathsf{f}\langle\bar{\alpha}_m\sigma\rangle : \bar{\tau}_n\sigma \Rightarrow \upsilon\sigma$. Parameters are passed in parentheses.

The set of λ-preterms is built from the following expressions:

- a variable $x\langle\tau\rangle : \tau$ for $x \in X$ and a type τ;
- a symbol $\mathsf{f}\langle\bar{\upsilon}_m\rangle(\bar{u}_n) : \tau$ for a constant $\mathsf{f} \in \Sigma$ with type declaration $\Pi\bar{\alpha}_m.\ \bar{\tau}_n \Rightarrow \tau$, types $\bar{\upsilon}_m$, and λ-preterms $\bar{u} : \bar{\tau}_n$ such that all De Bruijn indices in $\bar{u}$ are bound;
- a De Bruijn index $n\langle\tau\rangle : \tau$ for a natural number $n \geq 0$ and a type τ, where τ represents the type of the bound variable;
- a λ-expression $\lambda\langle\tau\rangle\, t : \tau \to \upsilon$ for a type τ and a λ-preterm $t : \upsilon$ such that all De Bruijn indices bound by the new $\lambda\langle\tau\rangle$ have type τ;
- an application $s\, t : \upsilon$ for λ-preterms $s : \tau \to \upsilon$ and $t : \tau$.

The type arguments $\langle\bar{\tau}\rangle$ carry enough information to enable typing of any λ-preterm without any context. We often leave them implicit, when they are irrelevant or can be inferred. In $\mathsf{f}\langle\bar{\upsilon}_m\rangle(\bar{u}_n) : \tau$, we call $\bar{u}_n$ the parameters. We omit () when a symbol has no parameters. As a syntactic convenience, symbols corresponding to infix operators are applied infix. Notice that it is possible for a term to contain multiple occurrences of the same free De Bruijn index with different types. In contrast, the types of bound De Bruijn indices always match.

A λ-term is a λ-preterm without free De Bruijn indices.

The size $|\ |$ of a λ-preterm is defined recursively by the following equations:

$$|x| = 1 \qquad |\mathsf{f}(\bar{u})| = 1 + \sum\nolimits_i |u_i| \qquad |n| = 1 \qquad |\lambda\, t| = 1 + |t| \qquad |s\, t| = |s| + |t|$$

The set $\mathcal{T}^\infty_{\mathrm{pre}}(\Sigma_{\mathsf{ty}}, \Sigma, X_{\mathsf{ty}}, X)$ of *preterms* consists of the $\beta\eta$-equivalence classes of λ-preterms. The set $\mathcal{T}^\infty(\Sigma_{\mathsf{ty}}, \Sigma, X_{\mathsf{ty}}, X)$ of "*terms*" consists of the $\beta\eta$-equivalence classes of λ-terms. Preterms have the following four mutually exclusive forms, where $t, \bar{t}$ are terms:

- a fully applied variable $x\langle\tau\rangle\, \bar{t}$;
- a fully applied symbol $\mathsf{f}\langle\bar{\tau}\rangle(\bar{u})\, \bar{t}$;
- a fully applied De Bruijn index $n\langle\tau\rangle\, \bar{t}$;
- a λ-abstraction $\lambda\langle\tau\rangle\, t$.

"Fully applied" means that the preterm as a whole has nonfunctional type. The above view is reminiscent of first-order terms: The variable and symbol cases are essentially as for first-order terms, De Bruijn indices are regarded as symbols, and even the λ-abstraction $\lambda\, t$

can be thought of as a unary function application $\lambda(t)$. This will be the key to adapting the first-order KBO and LPO to higher-order preterms.

A type τ is monomorphic if $\tau \in \mathcal{Ty}(\Sigma_{\mathsf{ty}}, \emptyset)$—i.e., if it contains no variables; otherwise, it is polymorphic. A preterm is monomorphic if all its type arguments are monomorphic; otherwise, it is polymorphic. A preterm t is *ground* if it is closed and monomorphic, or equivalently if $t \in \mathcal{T}^{\infty}_{\mathrm{pre}}(\Sigma_{\mathsf{ty}}, \Sigma, \emptyset, \emptyset)$.

Substitutions are defined as mappings from a set of type variables to types and from term variables to terms of the same type. A *monomorphizing* type substitution maps all type variables to ground types and leaves term variables unchanged. A *grounding* substitution maps all variables to ground types and terms.

We will say that a preterm is *steady* if its type is neither of function type nor a type variable.

Unless otherwise specified, all preterms will be presented in η-long normal form.

2.2. **Orders.**

Definition 2.1. Given a binary relation $\succ$, we write $\succeq$ for its reflexive closure.

Definition 2.2. Given a binary relation $\succ$, we write $\succ^{\mathsf{lex}}$ for its left-to-right lexicographic extension, defined as follows on same-length tuples: $() \succ^{\mathsf{lex}} ()$ does not hold, and for $n \geq 1$, $(y_1, \ldots, y_n) \succ^{\mathsf{lex}} (x_1, \ldots, x_n)$ holds if and only if $y_1 \succ x_1$ or else $y_1 = x_1$ and $(y_2, \ldots, y_n) \succ^{\mathsf{lex}} (x_2, \ldots, x_n)$.

Definition 2.3. Given binary relations $\succ$ and $\succsim$, we write $\succsim^{\mathsf{lex}}$ for their left-to-right strict lexicographic extension, defined as follows on same-length tuples: $() \succsim^{\mathsf{lex}} ()$ does not hold, and for $n \geq 1$, $(y_1, \ldots, y_n) \succsim^{\mathsf{lex}} (x_1, \ldots, x_n)$ holds if and only if $y_1 \succ x_1$ or else $y_1 \succsim x_1$ and $(y_2, \ldots, y_n) \succsim^{\mathsf{lex}} (x_2, \ldots, x_n)$.

Definition 2.4. Given binary relations $\succ$ and $\succsim$, we write $\succapprox^{\mathsf{lex}}$ for their left-to-right nonstrict lexicographic extension, defined as follows on same-length tuples: $() \succapprox^{\mathsf{lex}} ()$ holds, and for $n \geq 1$, $(y_1, \ldots, y_n) \succapprox^{\mathsf{lex}} (x_1, \ldots, x_n)$ holds if and only if $y_1 \succ x_1$ or else $y_1 \succsim x_1$ and $(y_2, \ldots, y_n) \succapprox^{\mathsf{lex}} (x_2, \ldots, x_n)$.

Definition 2.5. A *precedence* $>$ on a set A is a well-founded total order $>$ on A.

The first-order Knuth–Bendix order will constitute a useful stepping stone. Like the original [7], the version we use is untyped. Unlike the original, but like the transfinite KBO [10], it uses ordinal weights instead of natural numbers and supports argument coefficients.

Definition 2.6. Let $\mathcal{w} : \Sigma \to \mathbf{O}_{>0}$ and $\mathcal{k} : \Sigma \times \mathbb{N}_{>0} \to \mathbf{O}_{>0}$. Define the weight function $\mathcal{W} : \mathcal{T}(\Sigma, \emptyset) \to \mathbf{O}$ recursively by

$$\mathcal{W}(x) = 0 \qquad\qquad \mathcal{W}(\mathsf{f}(\bar{s}_m)) = \mathcal{w}(\mathsf{f}) + \sum\nolimits_{i=1}^{m} \mathcal{k}(\mathsf{f}, i)\mathcal{W}(s_i)$$

Definition 2.7. Let $\mathcal{w}, \mathcal{k}, \mathcal{W}$ be as in Definition 2.6. Let $>$ be an order (typically, a precedence) on an untyped signature Σ. The *strict first-order KBO* $\succ_{\mathsf{kbo}}$ induced by $\mathcal{w}, \mathcal{k}, >$ on nonground Σ-terms is defined inductively so that $t \succ_{\mathsf{kbo}} s$ if every variable occurring in s occurs at least as many times in t as in s and if any of these conditions is met:

(1) $\mathcal{W}(t) > \mathcal{W}(s)$;
(2) $\mathcal{W}(t) = \mathcal{W}(s)$, $t = \mathsf{g}(\bar{t})$, $s = \mathsf{f}(\bar{s})$, and $\mathsf{g} > \mathsf{f}$;
(3) $\mathcal{W}(t) = \mathcal{W}(s)$, $t = \mathsf{g}(\bar{t})$, $s = \mathsf{g}(\bar{s})$, and $\bar{t} \succ^{\mathsf{lex}}_{\mathsf{kbo}} \bar{s}$.

Definition 2.8. Let $>$ be an order (typically, a precedence) on an untyped signature Σ. The *strict first-order LPO* $\succ_{\mathsf{lpo}}$ induced by $>$ on nonground Σ-terms is defined inductively so that $t \succ_{\mathsf{lpo}} s$ if any of these conditions is met, where $t = \mathsf{g}(\bar{t}_k)$:

(1) $t_i \succeq_{\mathsf{lpo}} s$ for some $i \in \{1, \ldots, k\}$;
(2) $s = \mathsf{f}(\bar{s})$, $\mathsf{g} > \mathsf{f}$, and $\mathit{chkargs}(t, \bar{s})$;
(3) $s = \mathsf{g}(\bar{s})$, $\bar{t} \succ_{\mathsf{lpo}}^{\mathsf{lex}} \bar{s}$, and $\mathit{chkargs}(t, \bar{s})$

where $\mathit{chkargs}(t, \bar{s}_k)$ if and only if $t \succ_{\mathsf{lpo}} s_i$ for every $i \in \{1, \ldots, k\}$.

3. The Ground Level

We start by defining the λKBO and λLPO on ground preterms. We connect them to the first-order KBO and LPO via an encoding so that we can lift various properties, such as totality, well-foundedness, and compatibility with a wide class of contexts. For λKBO, in addition to $\mathcal{w}$, we will use the parameter $\mathcal{w}_\lambda$ for the weight of a λ and $\mathcal{w}_{\mathsf{db}}$ for the weight of a De Bruijn index.

For the rest of this paper, we fix a higher-order signature $(\Sigma_{\mathsf{ty}}, \Sigma)$ and two infinite sets of variables X_{ty}, X.

3.1. **λKBO.**

Definition 3.1. Let $\mathcal{w} : \Sigma \to \mathbf{O}_{>0}$, $\mathcal{w}_\lambda, \mathcal{w}_{\mathsf{db}} \in \mathbf{O}_{>0}$, and $\mathcal{k} : \Sigma \times \mathbb{N}_{>0} \to \mathbf{O}_{>0}$. Define the ground weight function $\mathcal{W}_{\mathsf{g}} : \mathcal{T}^{\infty}_{\mathrm{pre}}(\Sigma_{\mathsf{ty}}, \Sigma, \emptyset, \emptyset) \to \mathbf{O}_{>0}$ recursively by

$$\mathcal{W}_{\mathsf{g}}(\mathsf{f}(\bar{u})\,\bar{t}_n) = \mathcal{w}(\mathsf{f}) + \sum_{i=1}^{n} \mathcal{k}(\mathsf{f}, i)\,\mathcal{W}_{\mathsf{g}}(t_i) \qquad \mathcal{W}_{\mathsf{g}}(m\,\bar{t}_n) = \mathcal{w}_{\mathsf{db}} + \sum_{i=1}^{n} \mathcal{W}_{\mathsf{g}}(t_i)$$
$$\mathcal{W}_{\mathsf{g}}(\lambda\, t) = \mathcal{w}_\lambda + \mathcal{W}_{\mathsf{g}}(t)$$

Definition 3.2. Let $\mathcal{w}_{\mathsf{ty}} : \Sigma_{\mathsf{ty}} \to \mathbf{O}_{>0}$. Let $>^{\mathsf{ty}}$ be a precedence on Σ_{ty}. Let $\succ_{\mathsf{ty}}$ be the strict first-order KBO induced by $\mathcal{w}_{\mathsf{ty}}$ and $>^{\mathsf{ty}}$ on $\mathcal{T}(\Sigma_{\mathsf{ty}}, \emptyset)$. Let $\mathcal{w}, \mathcal{w}_\lambda, \mathcal{w}_{\mathsf{db}}, \mathcal{k}, \mathcal{W}_{\mathsf{g}}$ be as in Definition 3.1. Let $>$ be a precedence on Σ.

The *strict ground λKBO* $\succ_{\mathsf{g\lambda kbo}}$ induced by $\mathcal{w}_{\mathsf{ty}}, \mathcal{w}, \mathcal{w}_\lambda, \mathcal{w}_{\mathsf{db}}, \mathcal{k}, >^{\mathsf{ty}}, >$ on $\mathcal{T}^{\infty}_{\mathrm{pre}}(\Sigma_{\mathsf{ty}}, \Sigma, \emptyset, \emptyset)$ is defined inductively so that $t \succ_{\mathsf{g\lambda kbo}} s$ if any of these conditions is met:

(1) $\mathcal{W}_{\mathsf{g}}(t) > \mathcal{W}_{\mathsf{g}}(s)$;
(2) $\mathcal{W}_{\mathsf{g}}(t) = \mathcal{W}_{\mathsf{g}}(s)$, t is of the form $\lambda\langle\upsilon\rangle\, t'$, and any of these conditions is met:
  (a) s is of the form $\lambda\langle\tau\rangle\, s'$ and $\upsilon \succ_{\mathsf{ty}} \tau$, or
  (b) s is of the form $\lambda\langle\upsilon\rangle\, s'$ and $t' \succ_{\mathsf{g\lambda kbo}} s'$, or
  (c) s is of the form $m\,\bar{s}$ or $\mathsf{f}(\bar{u})\,\bar{s}$;
(3) $\mathcal{W}_{\mathsf{g}}(t) = \mathcal{W}_{\mathsf{g}}(s)$, t is of the form $n\,\bar{t}$, and any of these conditions is met:
  (a) s is of the form $m\,\bar{s}$ and $n > m$, or
  (b) s is of the form $n\,\bar{s}$ and $\bar{t} \succ_{\mathsf{g\lambda kbo}}^{\mathsf{lex}} \bar{s}$, or
  (c) s is of the form $\mathsf{f}(\bar{u})\,\bar{s}$;
(4) $\mathcal{W}_{\mathsf{g}}(t) = \mathcal{W}_{\mathsf{g}}(s)$, t is of the form $\mathsf{g}\langle\bar{\upsilon}\rangle(\bar{v})\,\bar{t}$, and any of these conditions is met:
  (a) s is of the form $\mathsf{f}(\bar{u})\,\bar{s}$ and $\mathsf{g} > \mathsf{f}$, or
  (b) s is of the form $\mathsf{g}\langle\bar{\tau}\rangle(\bar{u})\,\bar{s}$ and $\bar{\upsilon} \succ_{\mathsf{ty}}^{\mathsf{lex}} \bar{\tau}$, or
  (c) s is of the form $\mathsf{g}\langle\bar{\upsilon}\rangle(\bar{u})\,\bar{s}$ and $(\bar{v}, \bar{t}) \succ_{\mathsf{g\lambda kbo}}^{\mathsf{lex}} (\bar{u}, \bar{s})$.

In rule 3b, we assume that leaking De Bruijn indices in t and s refer to the same variable and hence have the same type. This invariant is preserved by the recursive application in rule 2b. A more defensive approach would be to compare the types first and then the argument tuples as a tie breaker, as in rules 4b and 4c.

Definition 3.3. Given a higher-order signature $(\Sigma_{\mathsf{ty}}, \Sigma)$, let

$$\begin{aligned}\Sigma_{\mathsf{fo}} = &\{\mathsf{f}^{\bar{\tau}}_{\bar{u}} \mid \mathsf{f} \in \Sigma,\, \bar{\tau} \in (\mathcal{Ty}(\Sigma_{\mathsf{ty}}, \emptyset))^{\mathit{tyarity}(\mathsf{f})},\, \bar{u} \in (\mathcal{T}^{\infty}_{\mathrm{pre}}(\Sigma_{\mathsf{ty}}, \Sigma, \emptyset, \emptyset))^{\mathit{arity}(\mathsf{f})}\}\\ &\uplus \{\mathsf{db}^i_k \mid i, k \in \mathbb{N}\} \uplus \{\mathsf{lam}^{\tau} \mid \tau \in \mathcal{Ty}(\Sigma_{\mathsf{ty}}, \emptyset)\}\end{aligned}$$

be an untyped first-order signature.

Definition 3.4. The translation $\mathcal{E}$ defined by the following equations encodes $\mathcal{T}^{\infty}_{\mathrm{pre}}(\Sigma_{\mathsf{ty}}, \Sigma, \emptyset, \emptyset)$ into $\mathcal{T}(\Sigma_{\mathsf{fo}}, \emptyset)$:

$$\mathcal{E}(\mathsf{f}\langle\bar{\tau}\rangle(\bar{u})\; \bar{t}) = \mathsf{f}^{\bar{\tau}}_{\bar{u}}(\mathcal{E}(\bar{t})) \qquad \mathcal{E}(m\; \bar{t}_n) = \mathsf{db}^m_n(\mathcal{E}(\bar{t}_n)) \qquad \mathcal{E}(\lambda\langle\tau\rangle\; t) = \mathsf{lam}^{\tau}(\mathcal{E}(t))$$

Lemma 3.5. *The translation $\mathcal{E}$ is injective on ground terms.*

Proof. By straightforward induction on $\mathcal{E}$'s definition. Since we claim injectivity only for terms, not all preterms, there is no need the type of a De Bruijn index. The type of a De Bruijn index is given by the corresponding enclosing lam^{τ}. Similarly, the type of a parameter is given by the function it is passed to. □

It will be useful to apply $\succ_{\mathsf{kbo}}$ to encoded terms. Let the symbol weights $\mathcal{w}_{\mathsf{fo}}$ and coefficients $\mathcal{k}_{\mathsf{fo}}$ be derived from $\mathcal{w}$ as follows:

$$\begin{aligned} \mathcal{w}_{\mathsf{fo}}(\mathsf{f}^{\bar{\tau}}_{\bar{u}}) &= \mathcal{w}(\mathsf{f}) & \mathcal{w}_{\mathsf{fo}}(\mathsf{db}^i_k) &= \mathcal{w}_{\mathsf{db}} & \mathcal{w}_{\mathsf{fo}}(\mathsf{lam}^{\tau}) &= \mathcal{w}_{\lambda}\\ \mathcal{k}_{\mathsf{fo}}(\mathsf{f}^{\bar{\tau}}_{\bar{u}}, i) &= \mathcal{k}(\mathsf{f}, i) & \mathcal{k}_{\mathsf{fo}}(\mathsf{db}^j_k, i) &= 1 & \mathcal{k}_{\mathsf{fo}}(\mathsf{lam}^{\tau}, i) &= 1 \end{aligned}$$

Next, let $>^{\mathsf{kbo}}$ be the precedence on Σ_{fo} that sorts the elements as follows, from smallest to largest:

(1) Start with the symbols $\mathsf{f}^{\bar{\tau}}_{\bar{u}}$ in $<$-increasing order of their symbols f, using $\prec^{\mathsf{lex}}_{\mathsf{ty}}$ on their superscripts as first tiebreaker and $\prec^{\mathsf{lex}}_{\mathsf{g\lambda kbo}}$ on the subscripts as second tiebreaker.

(2) Continue with the De Bruijn indices: $\mathsf{db}^0_0, \mathsf{db}^0_1, \ldots$, followed by $\mathsf{db}^1_0, \mathsf{db}^1_1, \ldots$, $\mathsf{db}^2_0, \mathsf{db}^2_1, \ldots$, and so on.

(3) Conclude with the symbols lam^{τ} in $\prec_{\mathsf{ty}}$-increasing order of their superscripts.

This definition ensures that symbols from Σ are smallest and lam^{τ} symbols are largest.

Let $\succ_{\mathsf{kbo}}$ denote the first-order KBO instance induced by $\mathcal{w}_{\mathsf{fo}}, \mathcal{k}_{\mathsf{fo}}, >^{\mathsf{kbo}}$, and let $\mathcal{W}_{\mathsf{kbo}}$ denote its weight function. The translation $\mathcal{E}$ is faithful in the following sense:

Lemma 3.6. *$\mathcal{W}_{\mathsf{kbo}}(\mathcal{E}(u)) = \mathcal{W}_{\mathsf{g}}(u)$ for every $u \in \mathcal{T}^{\infty}_{\mathrm{pre}}(\Sigma_{\mathsf{ty}}, \Sigma, \emptyset, \emptyset)$.*

Proof. By induction on the definition of $\mathcal{W}_{\mathsf{g}}$. □

Lemma 3.7. *Given $s, t \in \mathcal{T}^{\infty}_{\mathrm{pre}}(\Sigma_{\mathsf{ty}}, \Sigma, \emptyset, \emptyset)$, we have $t \succ_{\mathsf{g\lambda kbo}} s$ if and only if $\mathcal{E}(t) \succ_{\mathsf{kbo}} \mathcal{E}(s)$.*

Proof. By Lemma 3.6, any preterm u has the same weight according to $\succ_{\mathsf{g\lambda kbo}}$ as $\mathcal{E}(u)$ according to $\succ_{\mathsf{kbo}}$. The rules for establishing $t \succ_{\mathsf{g\lambda kbo}} s$ and $\mathcal{E}(t) \succ_{\mathsf{kbo}} \mathcal{E}(s)$ correspond according to the following table:

$\succ_{\mathsf{kbo}}$	$\succ_{\mathsf{g\lambda kbo}}$
Rule 1	Rule 1
Rule 2	Rule 2a, 2c, 3a, 3c, 4a, 4b, or 4c
Rule 3	Rule 2b, 3b, or 4c

The equivalence can then be established by two proofs by induction on the definition on $\succ_{\mathsf{kbo}}$ and $\succ_{\mathsf{g\lambda kbo}}$, one for each direction of the equivalence. □

Theorem 3.8. *The relation $\succ_{\mathsf{g\lambda kbo}}$ is a strict partial order.*

Proof. This amounts to proving irreflexivity, antisymmetry, and transitivity. The strategy is always the same and is illustrated for irreflexivity below.

IRREFLEXIVITY: We must show $t \not\succ_{\mathsf{g\lambda kbo}} t$. By Lemma 3.7, this amounts to showing $\mathcal{E}(t) \not\succ_{\mathsf{kbo}} \mathcal{E}(t)$, which is obvious since $\succ_{\mathsf{kbo}}$ is irreflexive. □

Lemma 3.9. *The relation $>^{\mathsf{kbo}}$ is a precedence.*

Proof. It is easy to see that the relation is total. For well-foundedness, suppose there exists an infinite descending chain $\mathsf{g}_0 >^{\mathsf{kbo}} \mathsf{g}_1 >^{\mathsf{kbo}} \cdots$.

We say that a symbol g is *bad* if there exists an infinite chain $\mathsf{g} >^{\mathsf{kbo}} \cdots$. Let us define the size $\|\ \|$ of a symbol as follows:

$$\|\mathsf{f}^{\bar{\sigma}}_{\bar{u}}\| = 1 + \sum\nolimits_i \|u_i\| \qquad\qquad \|\mathsf{db}^i_k\| = 1 \qquad\qquad \|\mathsf{lam}^\tau\| = 1$$

We can assume without loss of generality that the chain $\mathsf{g}_0 >^{\mathsf{kbo}} \mathsf{g}_1 >^{\mathsf{kbo}} \cdots$ is minimal in the following sense: g_0 has minimal size among bad symbols, and each g_{i+1} has minimal size among bad symbols g such that $\mathsf{g}_i >^{\mathsf{kbo}} \mathsf{g}$.

The chain must have infinitely many steps of type 1, 2, or 3. Since all steps of the same type are grouped together, there must exist an index k from which all steps are of the same type. We distinguish three cases, corresponding to the three types.

CASE 1: The chain $\mathsf{g}_k >^{\mathsf{kbo}} \mathsf{g}_{k+1} >^{\mathsf{kbo}} \cdots$, where each symbol g_i is of the form $\mathsf{f}^{\bar{\tau}}_{\bar{u}}$, is also an infinite descending chain with respect to the lexicographic order induced by $<_{\mathsf{g}}$, $\prec^{\mathsf{lex}}_{\mathsf{ty}}$ (for a fixed length n given by *tyarity*), and $\prec^{\mathsf{lex}}_{\mathsf{g\lambda kbo}}$ (for a fixed length given by *arity*). Both $<_{\mathsf{g}}$ and $\prec^{\mathsf{lex}}_{\mathsf{ty}}$ are well founded, so there must exist an index l from which the symbol f and its superscript $\bar{\tau}$ are fixed, and only the subscripts $\bar{u}$ change. This means that we have an infinite chain of the form $(\bar{u}_n)_l \succ^{\mathsf{lex}}_{\mathsf{g\lambda kbo}} (\bar{u}_n)_{l+1} \succ^{\mathsf{lex}}_{\mathsf{g\lambda kbo}} \cdots$. By Lemma 3.7, there would also exist a chain $\mathcal{E}((\bar{u}_n)_l) \succ^{\mathsf{lex}}_{\mathsf{kbo}} \mathcal{E}((\bar{u}_n)_{l+1}) \succ^{\mathsf{lex}}_{\mathsf{kbo}} \cdots$.

Since the bounded lexicographic order is well founded, this means that there exists an infinite chain of the form $\mathcal{E}(v_l) \succ_{\mathsf{kbo}} \mathcal{E}(v_{l+1}) \succ_{\mathsf{kbo}} \cdots$. Recall that the standard KBO is well founded if the underlying precedence is well founded. If it is not, the standard well-foundedness argument tells us that there must exist an infinite chain of distinct head symbols $\mathsf{h}_0 >^{\mathsf{kbo}} \mathsf{h}_1 >^{\mathsf{kbo}} \cdots$. Clearly, h_0 is both bad and smaller than g_0, contradicting the minimality of g_0.

CASE 2: The chain $\mathsf{f}_k >^{\mathsf{kbo}} \mathsf{f}_{k+1} >^{\mathsf{kbo}} \cdots$ corresponds to an infinite descending chain with respect to the lexicographic order on pairs of natural numbers. Since that order is well founded, the chain is impossible.

CASE 3: From $\mathsf{lam}^\tau_0 >^{\mathsf{kbo}} \mathsf{lam}^\tau_1 >^{\mathsf{kbo}} \cdots$, we extract an infinite chain $\tau_0 \succ_{\mathsf{ty}} \tau_1 \succ_{\mathsf{ty}} \cdots$, contradicting the well-foundedness of the first-order KBO. □

Theorem 3.10. *The relation $\succ_{\mathsf{g\lambda kbo}}$ is total on ground terms.*

Proof. Assume $t \neq s$. We must show that $t \succ_{\mathsf{g\lambda kbo}} s$ or $t \prec_{\mathsf{g\lambda kbo}} s$. Note that by Lemma 3.5, $\mathcal{E}(t) \neq \mathcal{E}(s)$. Hence, by totality of $\succ_{\mathsf{kbo}}$, either $\mathcal{E}(t) \succ_{\mathsf{kbo}} \mathcal{E}(s)$ or $\mathcal{E}(t) \prec_{\mathsf{kbo}} \mathcal{E}(s)$. We obtain the desired result by applying Lemma 3.7 twice. □

Theorem 3.11. *The relation $\succ_{\mathsf{g\lambda kbo}}$ is well founded.*

Proof. This follows again straightforwardly by Lemma 3.7. If there existed an infinite chain $t_0 \succ_{\mathsf{g\lambda kbo}} t_1 \succ_{\mathsf{g\lambda kbo}} \cdots$, there would also exist an infinite chain $\mathcal{E}(t_0) \succ_{\mathsf{kbo}} \mathcal{E}(t_1) \succ_{\mathsf{kbo}} \cdots$, contradicting the well-foundedness of $\succ_{\mathsf{kbo}}$. □

The λ-superposition calculus relies on notions of green and orange subterms: the core inference rules use green subterms, whereas optional simplification rules use orange subterms. Since all green subterms are orange subterms, we focus on the latter.

Definition 3.12. *Orange subterms* are defined inductively on ground preterms as follows:

(1) Every preterm is an orange subterm of itself.
(2) Every orange subterm of an argument s_i in $\mathsf{f}\langle\bar{\tau}\rangle\,\bar{s}$ is an orange subterm of $\mathsf{f}\langle\bar{\tau}\rangle\,\bar{s}$.
(3) Every orange subterm of an argument s_i in $m\,\bar{s}$ is an orange subterm of $m\,\bar{s}$.
(4) Every orange subterm of u is an orange subterm of $\lambda\,u$.

The context $u[\;]$ surrounding an orange subterm s of $u[s]$ is called an *orange context*. The notation $u\langle\!\langle s\rangle\!\rangle$ indicates that s is an orange subterm in $u[s]$, and $u\langle\!\langle\;\rangle\!\rangle$ indicates that $u[\;]$ is an orange context. The *depth* of an orange context is the number of λs in $u[\;]$ that have the hole in their scope.

Definition 3.13. A relation $\succ$ is *compatible with orange contexts* if $t \succ s$ implies $u\langle\!\langle t{\uparrow}^k\rangle\!\rangle \succ u\langle\!\langle s{\uparrow}^k\rangle\!\rangle$ for every orange context $u\langle\!\langle\;\rangle\!\rangle$, where k is its depth. The relation $\succ$ enjoys the *orange subterm property* if $u\langle\!\langle s{\uparrow}^k\rangle\!\rangle \succeq s$ for every orange context $u\langle\!\langle\;\rangle\!\rangle$, where k is its depth.

Theorem 3.14. *The relation $\succ_{\mathsf{g\lambda kbo}}$ is compatible with orange contexts.*

Proof. Let $u\langle\!\langle\;\rangle\!\rangle$ be an orange context of depth k. Assume $t \succ_{\mathsf{g\lambda kbo}} s$. Note that by Lemma 3.7, $\mathcal{E}(t) \succ_{\mathsf{kbo}} \mathcal{E}(s)$. Moreover, by inspection of the rules of $\succ_{\mathsf{kbo}}$, we find that $\mathcal{E}(t{\uparrow}^k) \succ_{\mathsf{kbo}} \mathcal{E}(s{\uparrow}^k)$. This works because we give all De Bruijn indices the same weight, and the precedence of indices remains stable under shifting.

Now, observe that orange subterms are mapped to first-order subterms by $\mathcal{E}$. In particular, there exists a first-order context $v[\;]$ such that $\mathcal{E}(u\langle\!\langle t{\uparrow}^k\rangle\!\rangle) = v[\mathcal{E}(t{\uparrow}^k)]$ and $\mathcal{E}(u\langle\!\langle s{\uparrow}^k\rangle\!\rangle) = v[\mathcal{E}(s{\uparrow}^k)]$. By compatibility of $\succ_{\mathsf{kbo}}$ with contexts, we have $v[\mathcal{E}(t{\uparrow}^k)] \succ_{\mathsf{kbo}} v[\mathcal{E}(s{\uparrow}^k)]$. Thus, by Lemma 3.7, we get $u\langle\!\langle t{\uparrow}^k\rangle\!\rangle \succ_{\mathsf{g\lambda kbo}} u\langle\!\langle s{\uparrow}^k\rangle\!\rangle$, as desired. □

Theorem 3.15. *The relation $\succ_{\mathsf{g\lambda kbo}}$ has the orange subterm property.*

Proof. The key idea is as in the proof of Theorem 3.14. For any orange context $u\langle\!\langle\;\rangle\!\rangle$ of depth k, there exists a first-order context $v[\;]$ such that $\mathcal{E}(u\langle\!\langle s{\uparrow}^k\rangle\!\rangle) = v[\mathcal{E}(s{\uparrow}^k)]$. By the subterm property of $\succ_{\mathsf{kbo}}$, we have $v[\mathcal{E}(s{\uparrow}^k)] \geq_{\mathsf{kbo}} \mathcal{E}(s{\uparrow}^k)$. By inspection of the rules of $\succ_{\mathsf{kbo}}$, we also have $\mathcal{E}(s{\uparrow}^k) \geq_{\mathsf{kbo}} \mathcal{E}(s)$. By transitivity and Lemma 3.7, we get $u\langle\!\langle s{\uparrow}^k\rangle\!\rangle \succ_{\mathsf{g\lambda kbo}} s$, as desired. □

The last property is necessary for λ-superposition. It is easy to prove.

Theorem 3.16. *Assume $\mathcal{W}_{\mathsf{g}}(\top) = \mathcal{W}_{\mathsf{g}}(\bot) = 1$ and $\top < \bot < \mathsf{f}$ for every $\mathsf{f} \in \Sigma \setminus \{\top, \bot\}$. Then $\top \prec_{\mathsf{g\lambda kbo}} \bot \prec_{\mathsf{g\lambda kbo}} t$ for every $t \in \mathcal{T}^{\infty}_{\mathrm{pre}}(\Sigma_{\mathsf{ty}}, \Sigma, \emptyset, \emptyset) \setminus \{\top, \bot\}$.*

Proof. This follows straightforwardly from the definition of $\succ_{\mathsf{g\lambda kbo}}$. $\square$

The λ-superposition calculus also specifies a requirement on applied quantifiers $\forall$ and $\exists$ occurring in clauses, after clausification. However, this requirement is not met by our order. To circumvent the issue, we can preprocess the quantifiers, replacing $\forall\,(\lambda\, t)$ by $(\lambda\, t) \approx (\lambda\, \top)$ and $\exists\,(\lambda\, t)$ by $(\lambda\, t) \not\approx (\lambda\, \bot)$.

Theorem 3.17. *Assume $\mathcal{w}(\mathsf{diff}) \leq \mathcal{w}_{\mathsf{db}}$ and $\mathcal{k}(\mathsf{diff}, i) = 1$ for every i. For all ground types τ, υ and ground preterms $s, t, u : \tau \to \upsilon$, we have $u \succ_{\mathsf{g\lambda kbo}} u\, \mathsf{diff}\langle\tau, \upsilon\rangle(s, t)$.*

Proof. Since u is of type $\tau \to \upsilon$, in its η-long normal form, it has the form $\lambda\, u'$ for some u'. Since $\mathsf{diff}\langle\tau, \upsilon\rangle(s, t)$ is a symbol, we can obtain the η-long β-normal form of $u\mathsf{diff}\langle\tau, \upsilon\rangle(s, t)$ by replacing free De Bruijn indices of u' by $\mathsf{diff}\langle\tau, \upsilon\rangle(s, t)$. Since $\mathcal{w}(\mathsf{diff}) \leq \mathcal{w}_{\mathsf{db}}$ and $\mathcal{k}(\mathsf{diff}, i) = 1$ for every i, it follows that $\mathcal{W}_{\mathsf{g}}(u\, \mathsf{diff}\langle\tau, \upsilon\rangle(s, t)) \leq \mathcal{W}_{\mathsf{g}}(u') < W_{\mathsf{g}}(u)$. Therefore, $u \succ_{\mathsf{g\lambda kbo}} u\, \mathsf{diff}\langle\tau, \upsilon\rangle(s, t)$ by rule 1. $\square$

3.2. **λLPO.**

Definition 3.18. Let $>^{\mathsf{ty}}$ be a precedence on Σ_{ty}. Let $\succ_{\mathsf{ty}}$ be the strict first-order LPO induced by $>^{\mathsf{ty}}$ on $\mathcal{T}(\Sigma_{\mathsf{ty}}, \emptyset)$. Let $>$ be a precedence on Σ. Let $\mathsf{ws} \in \Sigma$ be a distinguished element called the *watershed*.

The *strict ground λLPO* $\succ_{\mathsf{g\lambda lpo}}$ induced by $>^{\mathsf{ty}}, >$ on $\mathcal{T}^{\infty}_{\mathrm{pre}}(\Sigma_{\mathsf{ty}}, \Sigma, \emptyset, \emptyset)$ is defined inductively so that $t \succ_{\mathsf{g\lambda lpo}} s$ if any of these conditions is met:

(1) t is of the form $\mathsf{g}\langle\bar{\upsilon}\rangle(\bar{v})\, \bar{t}_k$ and any of these conditions is met:
  (a) $t_i \succeq_{\mathsf{g\lambda lpo}} s$ for some $i \in \{1, \ldots, k\}$, or
  (b) $s = \mathsf{f}(\bar{u})\, \bar{s}$, $\mathsf{g} > \mathsf{f}$, and $\mathit{chkargs}(t, \bar{s})$, or
  (c) $s = \mathsf{g}\langle\bar{\tau}\rangle(\bar{u})\, \bar{s}$, $\bar{\upsilon} \succ^{\mathsf{lex}}_{\mathsf{ty}} \bar{\tau}$, and $\mathit{chkargs}(t, \bar{s})$, or
  (d) $s = \mathsf{g}\langle\bar{\upsilon}\rangle(\bar{u})\, \bar{s}$, $(\bar{v}, \bar{t}) \succ^{\mathsf{lex}}_{\mathsf{g\lambda lpo}} (\bar{u}, \bar{s})$, and $\mathit{chkargs}(t, \bar{s})$, or
  (e) $\mathsf{g} > \mathsf{ws}$, s is of the form $m\, \bar{s}$ and $\mathit{chkargs}(t, \bar{s})$ or of the form $\lambda\, s'$ and $\mathit{chkargs}(t, [s'])$;
(2) t is of the form $n\, \bar{t}_k$ and any of these conditions is met:
  (a) $t_i \succeq_{\mathsf{g\lambda lpo}} s$ for some $i \in \{1, \ldots, k\}$, or
  (b) $s = m\, \bar{s}$, $n > m$, and $\mathit{chkargs}(t, \bar{s})$, or
  (c) $s = n\, \bar{s}$, $\bar{t} \succ^{\mathsf{lex}}_{\mathsf{g\lambda lpo}} \bar{s}$, and $\mathit{chkargs}(t, \bar{s})$, or
  (d) s is of the form $\lambda\, s'$ and $\mathit{chkargs}(t, [s'])$ or of the form or $\mathsf{f}(\bar{u})\, \bar{s}$, where $\mathsf{f} \leq \mathsf{ws}$, and $\mathit{chkargs}(t, \bar{s})$;
(3) t is of the form $\lambda\langle\upsilon\rangle\, t'$ and any of these conditions is met:
  (a) $t' \succeq_{\mathsf{g\lambda lpo}} s$, or
  (b) $s = \lambda\langle\tau\rangle\, s'$, $\upsilon \succ_{\mathsf{ty}} \tau$, and $\mathit{chkargs}(t, [s'])$, or
  (c) $s = \lambda\langle\upsilon\rangle\, s'$ and $t' \succ_{\mathsf{g\lambda lpo}} s'$, or
  (d) s is of the form $\mathsf{f}(\bar{u})\, \bar{s}$, where $\mathsf{f} \leq \mathsf{ws}$, and $\mathit{chkargs}(t, \bar{s})$

where $\mathit{chkargs}(t, \bar{s}_k)$ if and only if $t \succ_{\mathsf{g\lambda lpo}} s_i$ for every $i \in \{1, \ldots, k\}$. The notation [ ] is used to represent lists—here, the singleton list.

Let Σ_{fo} be a first-order signature as defined in Sect. 3.1. Let $>^{\mathsf{lpo}}$ be the precedence on Σ_{fo} that orders the elements as follows, from smallest to largest:

(1) Start with the symbols $\mathsf{f}^{\bar{\tau}}_{\bar{u}}$ such that $\mathsf{f} \leq \mathsf{ws}$ in $<$-increasing order of their symbols f, using $\prec^{\mathsf{lex}}_{\mathsf{ty}}$ on their superscripts as first tiebreaker and $\prec^{\mathsf{lex}}_{\mathsf{g\lambda lpo}}$ on the subscripts as second tiebreaker.

(2) Continue with the symbols lam^{τ} in $\prec_{\mathsf{ty}}$-increasing order of their superscripts.
(3) Continue with the De Bruijn indices: $\mathsf{db}^0_0, \mathsf{db}^0_1, \ldots$, followed by $\mathsf{db}^1_0, \mathsf{db}^1_1, \ldots, \mathsf{db}^2_0, \mathsf{db}^2_1, \ldots$, and so on.
(4) Conclude with the symbols $\mathsf{f}^{\bar{\tau}}_{\bar{u}}$ such that $\mathsf{f} > \mathsf{ws}$ in $<$-increasing order of their symbols f, using $\prec^{\mathsf{lex}}_{\mathsf{ty}}$ on their superscripts as first tiebreaker and $\prec^{\mathsf{lex}}_{\mathsf{g\lambda lpo}}$ on the subscripts as second tiebreaker.

This definition ensures that symbols below the watershed are smallest and symbols above the watershed are largest. When considering polymorphism, we will see that it is advantageous to put symbols above the watershed. However, the special symbol diff belongs below the watershed.

Lemma 3.19. *The relation $>^{\mathsf{lpo}}$ is a precedence*

Proof. The proof is analogous to that of Lemma 3.9. □

Let $\succ_{\mathsf{lpo}}$ denote the first-order LPO instance induced by the precedence $>^{\mathsf{lpo}}$.

Lemma 3.20. *Given $s, t \in \mathcal{T}^{\infty}_{\mathrm{pre}}(\Sigma_{\mathsf{ty}}, \Sigma, \emptyset, \emptyset)$, we have $t \succ_{\mathsf{g\lambda lpo}} s$ if and only if $\mathcal{E}(t) \succ_{\mathsf{lpo}} \mathcal{E}(s)$.*

Proof. The rules for establishing $t \succ_{\mathsf{g\lambda lpo}} s$ and $\mathcal{E}(t) \succ_{\mathsf{lpo}} \mathcal{E}(s)$ correspond according to the following table:

$\succ_{\mathsf{lpo}}$	$\succ_{\mathsf{g\lambda lpo}}$
Rule 1	Rule 1a, 2a, or 3a
Rule 2	Rule 1b, 1c, 1d, 1e, 2b, 2d, 3b, or 3d
Rule 3	Rule 1d, 2c, or 3c

The equivalence can then be established by two proofs by induction on the definition on $\succ_{\mathsf{lpo}}$ and $\succ_{\mathsf{g\lambda lpo}}$, one for each direction of the equivalence. The only nontrivial case is that of rule 3c of $\succ_{\mathsf{g\lambda lpo}}$, because it lacks the $\mathcal{chkargs}$ condition of the corresponding rule 3 of $\succ_{\mathsf{lpo}}$. Given $t \succ_{\mathsf{g\lambda lpo}} s$ by rule 3c, to obtain $\mathcal{E}(t) \succ_{\mathsf{lpo}} \mathcal{E}(s)$ by rule 3, we must show $\mathcal{chkargs}(\mathcal{E}(t), (\mathcal{E}(s')))$. We apply transitivity to combine $\mathcal{E}(t) \succ_{\mathsf{lpo}} \mathcal{E}(t')$, which follows from the subterm property, and the induction hypothesis $\mathcal{E}(t') \succ_{\mathsf{lpo}} \mathcal{E}(s')$. □

Using Lemma 3.20, we can prove the following theorems using the same strategy as for Theorems 3.8–3.16:

Theorem 3.21. *The relation $\succ_{\mathsf{g\lambda lpo}}$ is a strict partial order.*

Theorem 3.22. *The relation $\succ_{\mathsf{g\lambda lpo}}$ is total on ground preterms.*

Theorem 3.23. *The relation $\succ_{\mathsf{g\lambda lpo}}$ is well founded.*

Theorem 3.24. *The relation $\succ_{\mathsf{g\lambda lpo}}$ is compatible with orange contexts.*

Theorem 3.25. *The relation $\succ_{\mathsf{g\lambda lpo}}$ has the orange subterm property.*

Theorem 3.26. *Assume $\top < \bot < \mathsf{f}$ for every $\mathsf{f} \in \Sigma \setminus \{\top, \bot\}$ and $\bot \leq \mathsf{ws}$. Then $\top \prec_{\mathsf{g\lambda lpo}} \bot \prec_{\mathsf{g\lambda lpo}} t$ for every $t \in \mathcal{T}^{\infty}_{\mathrm{pre}}(\Sigma_{\mathsf{ty}}, \Sigma, \emptyset, \emptyset) \setminus \{\top, \bot\}$.*

Theorem 3.27. *Let $\mathsf{diff} \leq \mathsf{ws}$. For all ground types τ, υ and ground preterms $s, t, u : \tau \to \upsilon$, we have $u \succ_{\mathsf{g\lambda lpo}} u\, \mathsf{diff}\langle\tau, \upsilon\rangle(s, t)$.*

Proof. Since u is of type $\tau \to \upsilon$, in its η-long normal form, it has the form $\lambda\, u'$ for some u'. Since $\mathsf{diff}\langle\tau,\upsilon\rangle(s,t)$ is a symbol, we can obtain the η-long β-normal form of $u\,\mathsf{diff}\langle\tau,\upsilon\rangle(s,t)$ by replacing the free De Bruijn indices of u' by $\mathsf{diff}\langle\tau,\upsilon\rangle(s,t)$.

So, in order to show that $u \succ_{\mathsf{g\lambda lpo}} u\,\mathsf{diff}\langle\tau,\upsilon\rangle(s,t)$, we apply rule 3a, and it remains to show that $u' \succeq_{\mathsf{g\lambda lpo}} u\,\mathsf{diff}\langle\tau,\upsilon\rangle(s,t)$. We follow the structure of u' and $u\,\mathsf{diff}\langle\tau,\upsilon\rangle(s,t)$ as follows. Whenever heads coincide, we apply rules 1d, 2c, or 3c to decompose both sides. When heads do not coincide, we note that by our observation above, this can only happen when one side is a De Bruijn index and the other side is $\mathsf{diff}\langle\tau,\upsilon\rangle(s,t)$. So we can then apply rule 2d because $\mathsf{diff} \le \mathsf{ws}$. For any *chkargs* conditions arising in this procedure, we apply rules 1a, 2a, or 3a, and use the same procedure for the resulting proof obligations. □

The above proof crucially depends on diff's placement below the watershed. If we allowed $\mathsf{diff} > \mathsf{ws}$, the comparison $0 \succ_{\mathsf{\lambda lpo}} \mathsf{diff}(\ldots)$ would fail. Theorem 3.27 is the watershed's reason for being.

4. The Monomorphic Level

Next, we generalize the definition of λKBO to monomorphic nonground preterms: preterms containing no type variables. The result coincides with the ground λKBO on ground preterms while supporting term variables. Variables give rise to polynomial constraints, which must be solved when comparing terms.

The key idea, already present in the λ-free KBO by Becker et al. [1], is to use polynomials to symbolically represent the weight of a nonground term. The weight of $y\,\mathsf{a}$, where $\mathsf{a} : \kappa$, will be represented symbolically as $1 + \mathbf{w}_y + \mathbf{k}_{y,1}(\mathcal{w}(\mathsf{a}) - \mathcal{w}_{\mathsf{db}})$, where $1 + \mathbf{w}_y$ stands for the weight of whatever term will instantiate y without its leading λs and $\mathbf{k}_{y,1}$ for the number of copies of the first curried argument, here a, that the term will make. If an argument coefficient other than 1 is used, that number of copies will be inflated by the coefficient. The $-\,\mathcal{w}_{\mathsf{db}}$ monomial accounts for the loss of a De Bruijn index occurring in y when passing the argument a and β-reducing.

A subtle difference between the indeterminate $\mathbf{k}_{y,1}$ and the argument coefficient $\mathcal{k}(\mathsf{f}, i)$ is that $\mathbf{k}_{y,1}$ can take a value of 0; for example, $\lambda\,\mathsf{b}$ makes zero copies of its argument. Becker et al. excluded this scenario so that they could get compatibility with arguments, but this property is not needed by λ-superposition.

Another subtlety concerns higher-order functions. The arithmetic above works because the argument a is a simple symbol. If it were a λ-abstraction, it could appear applied inside y and trigger further β-reductions, complicating matters. In such cases, we simply give up and use a single indeterminate $\mathbf{w}_{y\,\bar{t}}$ to represent both the applied variable and its arguments of functional types. We do the same with arguments of variable type, since type variables can be instantiated with functional types.

Some precision can be gained by normalizing the subscript of $\mathbf{w}_{y\,\bar{t}}$. For example, if a and b have the same weight, then $\mathbf{w}_{y\,(\lambda\,\mathsf{a}\,0)}$ and $\mathbf{w}_{y\,(\lambda\,\mathsf{b}\,0)}$ will always evaluate to the same result and can be identified. Our simple analysis merges all symbols and De Bruijn indices with the same weight using a normalization function $[\ \ ]$.

4.1. $\boldsymbol{\lambda}$KBO.

Definition 4.1. Let $(\Sigma_{\mathsf{ty}}, \Sigma)$ be a higher-order signature. We denote by $\mathbf{P}$ the set of $\mathbf{O}$-valued polynomials of the following distinct indeterminates, where $y \in X$, $\bar{t} \in (\mathcal{T}^{\infty}_{\mathrm{pre}}(\Sigma_{\mathsf{ty}}, \Sigma, \emptyset, X))^*$, and $i \in \mathbb{N}_{>0}$:

- $\mathbf{w}_{y\,\bar{t}}$, ranging over $\mathbf{O}$, represents the weight, minus 1, of the variable y applied to the arguments $\bar{t}$ but without any leading λs corresponding to extra arguments;
- $\mathbf{k}_{y\,\bar{t},i}$, ranging over $\mathbf{O}$, represents the coefficient to apply on the ith extra argument of y already applied to $\bar{t}$.

An *assignment* is a mapping from indeterminates to values in the indeterminates' specified ranges. Given a polynomial $\mathcal{w}$ and an assignment A, $\mathcal{w}\big|_A \in \mathbf{O}$ denotes $\mathcal{w}$'s value under A, obtained by replacing each indeterminate $\mathbf{x}$ in A's domain by $A(\mathbf{x})$. Overloading notation, we write $\mathcal{w}\big|_\sigma$ for the application of the polynomial substitution σ to $\mathcal{w}$; for example, if $\sigma = \{\mathbf{w}_y \mapsto \mathbf{w}_z\}$, then $\mathbf{w}_y\big|_\sigma = \mathbf{w}_z$. Given polynomials w, w', we write $w' \geq w$ if we have $w'\big|_A \geq w\big|_A$ for every assignment A, and similarly for $>$, $\leq$, $<$, and $=$.

Definition 4.2. Let $\Sigma' = \Sigma \uplus \{k_\tau \mid k \in \mathbf{O}_{>0} \text{ and } \tau \in \mathcal{Ty}(\Sigma_{\mathsf{ty}}, \emptyset)\}$. Define the normalization function $[\ ] : \mathcal{T}^{\infty}_{\mathrm{pre}}(\Sigma_{\mathsf{ty}}, \Sigma, \emptyset, X) \to \mathcal{T}^{\infty}_{\mathrm{pre}}(\Sigma_{\mathsf{ty}}, \Sigma', \emptyset, X)$ recursively by

$$
\begin{aligned}
[y\,\bar{t}] &= y\,[\bar{t}] \\
[\mathsf{f}(\bar{u})\,\bar{t}] &= \begin{cases} k_\tau\,[\bar{t}] & \text{if } \mathcal{k}(\mathsf{f}, i) = 1 \text{ for every } i \text{, with } \mathcal{w}(\mathsf{f}) = k \text{ and } \mathsf{f}(\bar{u})\,\bar{t} : \tau \\ \mathsf{f}(\bar{u})\,[\bar{t}] & \text{otherwise} \end{cases} \\
[m\langle\tau\rangle\,\bar{t}] &= (\mathcal{w}_{\mathsf{db}})_\tau\,[\bar{t}] \\
[\lambda\,t] &= \lambda\,[t]
\end{aligned}
$$

Definition 4.3. Let $\mathcal{w} : \Sigma \to \mathbf{O}_{>0}$, $\mathcal{w}_\lambda, \mathcal{w}_{\mathsf{db}} \in \mathbf{O}_{>0}$, and $\mathcal{k} : \Sigma \times \mathbb{N}_{>0} \to \mathbf{O}_{>0}$. Given a list of preterms $\bar{t}$, let $\bar{t}^{\star\!\star}$ denote the longest suffix consisting of steady preterms, and let $\bar{t}^{\star}$ denote the complementary prefix. Define the monomorphic weight function $\mathcal{W}_{\mathsf{m}} : \mathcal{T}^{\infty}_{\mathrm{pre}}(\Sigma_{\mathsf{ty}}, \Sigma, \emptyset, X) \to \mathbf{P}$ recursively by

$$
\begin{aligned}
\mathcal{W}_{\mathsf{m}}(y\,\bar{t}) &= 1 + \mathbf{w}_{y\,[\bar{t}^{\star}]} + \sum\nolimits_{i=1}^{|\bar{t}^{\star\star}|} \mathbf{k}_{y\,[\bar{t}^{\star}],i}(\mathcal{W}_{\mathsf{m}}(\bar{t}_i^{\star\star}) - \mathcal{w}_{\mathsf{db}}) \\
\mathcal{W}_{\mathsf{m}}(\mathsf{f}(\bar{u})\,\bar{t}_n) &= \mathcal{w}(\mathsf{f}) + \sum\nolimits_{i=1}^{n} \mathcal{k}(\mathsf{f}, i)\,\mathcal{W}_{\mathsf{m}}(t_i) \\
\mathcal{W}_{\mathsf{m}}(m\,\bar{t}_n) &= \mathcal{w}_{\mathsf{db}} + \sum\nolimits_{i=1}^{n} \mathcal{W}_{\mathsf{m}}(t_i) \\
\mathcal{W}_{\mathsf{m}}(\lambda\,t) &= \mathcal{w}_\lambda + \mathcal{W}_{\mathsf{m}}(t)
\end{aligned}
$$

In the first equation, $\mathcal{W}_{\mathsf{m}}(\bar{t}_i^{\star\star})$ gives the argument's weight, whereas $\mathcal{w}_{\mathsf{db}}$ is the weight of the De Bruijn index that gets replaced by the argument.

Remark 4.4. It is possible to generalize the theory above to let $\bar{t}^{\star\star}$ consist of all steady preterms, regardless of their location. The interpretation of $\mathbf{w}_{y\,\bar{t}}$ and $\mathbf{k}_{y\,\bar{t},i}$ must then be changed to shuffle the λs, pulling those corresponding to the arguments $\bar{t}$ to the front. For example, if $y : \kappa \to (\kappa \to \kappa) \to \kappa$, the indeterminate $\mathbf{w}_{y\,t}$ represents the weight of the term $(\lambda\,\lambda\,y\,0\,1)\,t = \lambda\,y\,0\,(t\!\uparrow)$ (but without its leading λ).

Another possible generalization would be to normalize complex preterms, producing for example $2\langle\tau\rangle$ instead of $1\langle\sigma \to \tau\rangle\,1\langle\sigma\rangle$.

Definition 4.5. Let $\mathcal{w}_{\mathsf{ty}}, \mathcal{w}, \mathcal{w}_{\lambda}, \mathcal{w}_{\mathsf{db}}, \mathcal{k}, \mathcal{W}_{\mathsf{m}}$ be as in Definition 4.3. Let $>^{\mathsf{ty}}$ be a precedence on Σ_{ty}. Let $\succ_{\mathsf{ty}}$ be the strict first-order KBO on $\mathcal{T}(\Sigma_{\mathsf{ty}}, \emptyset)$ induced by $\mathcal{w}_{\mathsf{ty}}$ and $>^{\mathsf{ty}}$. Let $>$ be a precedence on Σ.

The *strict monomorphic λKBO* $\succ_{\mathsf{m\lambda kbo}}$ and the *nonstrict monomorphic λKBO* $\succsim_{\mathsf{m\lambda kbo}}$ induced by $\mathcal{w}_{\mathsf{ty}}, \mathcal{w}, \mathcal{w}_{\lambda}, \mathcal{w}_{\mathsf{db}}, \mathcal{k}, >^{\mathsf{ty}}, >$ on $\mathcal{T}^{\infty}_{\mathrm{pre}}(\Sigma_{\mathsf{ty}}, \Sigma, \emptyset, X)$ are defined by mutual induction. The strict relation is defined so that $t \succ_{\mathsf{m\lambda kbo}} s$ if any of these conditions is met:

(1) $\mathcal{W}_{\mathsf{m}}(t) > \mathcal{W}_{\mathsf{m}}(s)$;
(2) $\mathcal{W}_{\mathsf{m}}(t) \geq \mathcal{W}_{\mathsf{m}}(s)$, t is of the form $\lambda\langle\upsilon\rangle\, t'$, and any of these conditions is met:
  (a) s is of the form $\lambda\langle\tau\rangle\, s'$ and $\upsilon \succ_{\mathsf{ty}} \tau$, or
  (b) s is of the form $\lambda\langle\upsilon\rangle\, s'$ and $t' \succ_{\mathsf{m\lambda kbo}} s'$, or
  (c) s is of the form $m\,\bar{s}$ or $\mathsf{f}(\bar{u})\,\bar{s}$;
(3) $\mathcal{W}_{\mathsf{m}}(t) \geq \mathcal{W}_{\mathsf{m}}(s)$, t is of the form $n\,\bar{t}$, and any of these conditions is met:
  (a) s is of the form $m\,\bar{s}$ and $n > m$, or
  (b) s is of the form $n\,\bar{s}$ and $\bar{t} \succsim^{\mathsf{lex}}_{\mathsf{m\lambda kbo}} \bar{s}$, or
  (c) s is of the form $\mathsf{f}(\bar{u})\,\bar{s}$;
(4) $\mathcal{W}_{\mathsf{m}}(t) \geq \mathcal{W}_{\mathsf{m}}(s)$, t is of the form $\mathsf{g}\langle\bar{\upsilon}\rangle(\bar{v})\,\bar{t}$, and any of these conditions is met:
  (a) s is of the form $\mathsf{f}(\bar{u})\,\bar{s}$ and $\mathsf{g} > \mathsf{f}$, or
  (b) s is of the form $\mathsf{g}\langle\bar{\tau}\rangle(\bar{u})\,\bar{s}$ and $\bar{\upsilon} \succ^{\mathsf{lex}}_{\mathsf{ty}} \bar{\tau}$, or
  (c) s is of the form $\mathsf{g}\langle\bar{\upsilon}\rangle(\bar{u})\,\bar{s}$ and $(\bar{v}, \bar{t}) \succsim^{\mathsf{lex}}_{\mathsf{m\lambda kbo}} (\bar{u}, \bar{s})$.

The nonstrict relation is defined so that $t \succsim_{\mathsf{m\lambda kbo}} s$ if any of these conditions is met:

(1) $\mathcal{W}_{\mathsf{m}}(t) > \mathcal{W}_{\mathsf{m}}(s)$;
(2) t is of the form $y\,\bar{t}$, s is of the form $y\,\bar{s}$, and for every i, t_i is steady and $t_i \succsim_{\mathsf{m\lambda kbo}} s_i$;
(3) $\mathcal{W}_{\mathsf{m}}(t) \geq \mathcal{W}_{\mathsf{m}}(s)$, t is of the form $\lambda\langle\upsilon\rangle\, t'$, and any of these conditions is met:
  (a) s is of the form $\lambda\langle\tau\rangle\, s'$ and $\upsilon \succ_{\mathsf{ty}} \tau$, or
  (b) s is of the form $\lambda\langle\upsilon\rangle\, s'$ and $t' \succsim_{\mathsf{m\lambda kbo}} s'$, or
  (c) s is of the form $m\,\bar{s}$ or $\mathsf{f}(\bar{u})\,\bar{s}$;
(4) $\mathcal{W}_{\mathsf{m}}(t) \geq \mathcal{W}_{\mathsf{m}}(s)$, t is of the form $n\,\bar{t}$, and any of these conditions is met:
  (a) s is of the form $m\,\bar{s}$ and $n > m$, or
  (b) s is of the form $n\,\bar{s}$ and $\bar{t} \succapprox^{\mathsf{lex}}_{\mathsf{m\lambda kbo}} \bar{s}$, or
  (c) s is of the form $\mathsf{f}(\bar{u})\,\bar{s}$;
(5) $\mathcal{W}_{\mathsf{m}}(t) \geq \mathcal{W}_{\mathsf{m}}(s)$, t is of the form $\mathsf{g}\langle\bar{\upsilon}\rangle(\bar{v})\,\bar{t}$, and any of these conditions is met:
  (a) s is of the form $\mathsf{f}(\bar{u})\,\bar{s}$ and $\mathsf{g} > \mathsf{f}$, or
  (b) s is of the form $\mathsf{g}\langle\bar{\tau}\rangle(\bar{u})\,\bar{s}$ and $\bar{\upsilon} \succ^{\mathsf{lex}}_{\mathsf{ty}} \bar{\tau}$, or
  (c) s is of the form $\mathsf{g}\langle\bar{\upsilon}\rangle(\bar{u})\,\bar{s}$ and $(\bar{v}, \bar{t}) \succapprox^{\mathsf{lex}}_{\mathsf{m\lambda kbo}} (\bar{u}, \bar{s})$.

Rules 2 to 4 for $\succ_{\mathsf{m\lambda kbo}}$ and rules 3 to 5 for $\succsim_{\mathsf{m\lambda kbo}}$ use $\geq$ instead of $=$ to compare weights because polynomials cannot always be compared precisely. For example, if $\mathcal{w}(\mathsf{a}) = 1$, where $\mathsf{a} : \kappa$, we can know that $\mathcal{W}_{\mathsf{m}}(x) = 1 + \mathbf{w}_x \geq 1 = \mathcal{W}_{\mathsf{m}}(\mathsf{a})$ even though neither $\mathcal{W}_{\mathsf{m}}(x) > \mathcal{W}_{\mathsf{m}}(\mathsf{a})$ nor $\mathcal{W}_{\mathsf{m}}(x) = \mathcal{W}_{\mathsf{m}}(\mathsf{a})$.

To determine whether one preterm is larger than another, we must solve an inequality, which can be recast into $w \geq 0$ or $w > 0$. The strict case arises in rule 1 of the definitions of $\succ_{\mathsf{m\lambda kbo}}$ and $\succsim_{\mathsf{m\lambda kbo}}$. The two cases can be unified by writing $w > 0$ as $w - 1 \geq 0$. Solving systems of integer polynomial inequalities is in general undecidable. Here, however, we have a single polynomial w, in which indeterminates range only over nonnegative values. This is the key to solving the problem efficiently in practice. If infinite ordinals (e.g. ω) are used as the weight or coefficient associated with any symbols, we must also let the $\mathbf{w}_{y\,\bar{t}}$ and $\mathbf{k}_{y\,\bar{t},i}$ indeterminates range over these.

Specifically, we propose the following procedure to check an inequality of the above form: Put w in standard form. If all monomial coefficients are nonnegative, report that the inequality holds. Otherwise, report that it might not hold. This simple procedure can lose solutions. For example, $(\mathbf{w}_y - 3)\mathbf{w}_y + 3 \geq 0$ holds, yet its standard form $\mathbf{w}_y^2 - 3\mathbf{w}_y + 3$ contains a negative coefficient, which is enough to lead the procedure astray.

Below we will connect the monomorphic λKBO with its ground counterpart to lift its properties.

Lemma 4.6. *For all preterms t, we have $t \succsim_{\mathsf{m\lambda kbo}} t$.*

Proof. By structural induction on t, using rules 3b, 4b, and 5c. □

Lemma 4.7. *If $t \succ_{\mathsf{m\lambda kbo}} s$, then $t \succsim_{\mathsf{m\lambda kbo}} s$.*

Proof. By induction on the derivation of $t \succ_{\mathsf{m\lambda kbo}} s$. For each rule, there is a clearly corresponding rule of $\succsim_{\mathsf{m\lambda kbo}}$ to apply. The induction hypothesis is only required for applying rule 3b. A crucial observation is that $\succsim^{\mathsf{lex}}_{\mathsf{m\lambda kbo}}$ implies $\succapprox^{\mathsf{lex}}_{\mathsf{m\lambda kbo}}$ by definition. □

Definition 4.8. Let θ be a substitution that maps to terms containing only nonfunctional variables. We define an assignment $\mathit{poly}(\theta)$ that maps each indeterminate $\mathbf{x}$ to a value according to the semantics given by Definition 4.1 after applying θ onto the considered preterms. We define $\mathit{poly}(\theta)(\mathbf{w}_{y\,\bar{t}}) = \mathcal{W}_{\mathsf{m}}((y\,\bar{t})\theta!) - 1$, where, given a preterm t, $t!$ denotes the same preterm without any leading λs—e.g., $(\lambda\,\lambda\,\mathsf{f}\,0)! = \mathsf{f}\,0$. As for $\mathit{poly}(\theta)(\mathbf{k}_{y\,\bar{t},i})$, it is defined as the number of De Bruijn indices in $(y\,\bar{t})\theta$ referring to its ith argument, multiplied by all argument coefficients above it. Here, it is crucial that θ maps to terms containing only nonfunctional variables because the argument coefficient to assign to a variable is not always clear.

Lemma 4.9. *Given a substitution θ that maps to terms containing only nonfunctional variables, we have $\mathcal{W}_{\mathsf{m}}(t)\big|_{\mathit{poly}(\theta)} = \mathcal{W}_{\mathsf{m}}(t\theta)$.*

Proof. The proof is by induction on the definition of $\mathcal{W}_{\mathsf{m}}$. Let $A = \mathit{poly}(\theta)$.

Case $t = y\,\bar{t}$: We have

$$
\begin{aligned}
&\mathcal{W}_{\mathsf{m}}(y\,\bar{t})\big|_A \\
&= 1 + \mathbf{w}_{y\,\bar{t}^\star}\big|_A + \sum\nolimits_{i=1}^{|\bar{t}^{\star\star}|} \mathbf{k}_{y\,\bar{t}^\star,i}\big|_A (\mathcal{W}_{\mathsf{m}}(\bar{t}_i^{\star\star})\big|_A - \mathcal{w}_{\mathsf{db}}) \\
&\qquad \text{by definition of } \mathcal{W}_{\mathsf{m}} \text{ and } [\;] \\
&= 1 + \mathbf{w}_{y\,\bar{t}^\star}\big|_A + \sum\nolimits_{i=1}^{|\bar{t}^{\star\star}|} \mathbf{k}_{y\,\bar{t}^\star,i}\big|_A (\mathcal{W}_{\mathsf{m}}(\bar{t}_i^{\star\star}\theta) - \mathcal{w}_{\mathsf{db}}) \\
&\qquad \text{by the induction hypothesis} \\
&= \mathcal{W}_{\mathsf{m}}((y\,\bar{t})\theta) \\
&\qquad \text{by the semantics of } \mathbf{w} \text{ and } \mathbf{k}
\end{aligned}
$$

In the last step, the arithmetic works because all preterms in $\bar{t}^{\star\star}$ are steady. This means that they will not trigger any β-reductions when they replace a De Bruijn index.

Case $t = \mathsf{f}(\bar{u})\,\bar{t}_n$: We have

$$
\begin{aligned}
&\mathcal{W}_{\mathsf{m}}(\mathsf{f}(\bar{u})\,\bar{t}_n)\big|_A \\
&= \mathcal{w}(\mathsf{f}) + \sum\nolimits_{i=1}^{n} \mathcal{k}(\mathsf{f}, i)\,\mathcal{W}_{\mathsf{m}}(t_i)\big|_A \qquad \text{by definition of } \mathcal{W}_{\mathsf{m}}
\end{aligned}
$$

$$\begin{aligned}
&= w(\mathsf{f}) + \textstyle\sum_{i=1}^{n} \mathcal{k}(\mathsf{f}, i)\,\mathcal{W}_{\mathsf{m}}(t_i\theta) && \text{by the induction hypothesis}\\
&= \mathcal{W}_{\mathsf{m}}(\mathsf{f}(\bar{u}\theta)\;(\bar{t}_n\theta)) && \text{by definition of } \mathcal{W}_{\mathsf{m}}\\
&= \mathcal{W}_{\mathsf{m}}((\mathsf{f}(\bar{u})\;\bar{t}_n)\theta) && \text{by definition of substitution}
\end{aligned}$$

CASE $t = m\,\bar{t}_n$: This case is similar to the previous one. We have

$$\begin{aligned}
&\mathcal{W}_{\mathsf{m}}(m\,\bar{t}_n)\big|_A\\
&= w_{\mathsf{db}} + \textstyle\sum_{i=1}^{n} \mathcal{W}_{\mathsf{m}}(t_i)\big|_A && \text{by definition of } \mathcal{W}_{\mathsf{m}}\\
&= w_{\mathsf{db}} + \textstyle\sum_{i=1}^{n} \mathcal{W}_{\mathsf{m}}(t_i\theta) && \text{by the induction hypothesis}\\
&= \mathcal{W}_{\mathsf{m}}(m\;(\bar{t}_n\theta)) && \text{by definition of } \mathcal{W}_{\mathsf{m}}\\
&= \mathcal{W}_{\mathsf{m}}((m\,\bar{t}_n)\theta) && \text{by definition of substitution}
\end{aligned}$$

CASE $t = \lambda\, t$: We have

$$\begin{aligned}
&\mathcal{W}_{\mathsf{m}}(\lambda\, t)\big|_A\\
&= w_{\lambda} + \mathcal{W}_{\mathsf{m}}(t)\big|_A && \text{by definition of } \mathcal{W}_{\mathsf{m}}\\
&= w_{\lambda} + \mathcal{W}_{\mathsf{m}}(t\theta) && \text{by the induction hypothesis}\\
&= \mathcal{W}_{\mathsf{m}}(\lambda\,(t\theta)) && \text{by definition of } \mathcal{W}_{\mathsf{m}}\\
&= \mathcal{W}_{\mathsf{m}}((\lambda\, t)\theta) && \text{by definition of substitution}
\end{aligned}$$ □

The nonground relation $\succ_{\mathsf{m\lambda kbo}}$ underapproximates the ground relation $\succ_{\mathsf{g\lambda kbo}}$ in the following sense:

Theorem 4.10. *If $t \succ_{\mathsf{m\lambda kbo}} s$, then $t\theta \succ_{\mathsf{g\lambda kbo}} s\theta$ for all grounding substitutions θ. If $t \gtrsim_{\mathsf{m\lambda kbo}} s$, then $t\theta \succeq_{\mathsf{g\lambda kbo}} s\theta$ for all grounding substitutions θ.*

Proof. We prove both claims by mutual induction on the shape of the derivation of $t \succ_{\mathsf{m\lambda kbo}} s$ and $t \gtrsim_{\mathsf{m\lambda kbo}} s$. Let $A = \mathit{poly}(\theta)$.

First, we make the following observation: For all tuples of preterms $\bar{t}$ and $\bar{s}$ covered by the induction hypothesis,

- $\bar{t} \gtrsim^{\mathsf{lex}}_{\mathsf{m\lambda kbo}} \bar{s}$ implies $\bar{t}\theta \succ^{\mathsf{lex}}_{\mathsf{g\lambda kbo}} \bar{s}\theta$, and
- $\bar{t} \gtrapprox^{\mathsf{lex}}_{\mathsf{m\lambda kbo}} \bar{s}$ implies $\bar{t}\theta \succ^{\mathsf{lex}}_{\mathsf{g\lambda kbo}} \bar{s}\theta$ or $\bar{t}\theta = \bar{s}\theta$.

This follows from the induction hypothesis and the definitions of the lexicographic extensions (Definitions 2.2, 2.3, and 2.4) by induction on the length of the tuples.

With this observation, we prove the two claims of this theorem as follows. For the first claim, we make a case distinction on the rule deriving $t \succ_{\mathsf{m\lambda kbo}} s$:

RULE 1: From $\mathcal{W}_{\mathsf{m}}(t) > \mathcal{W}_{\mathsf{m}}(s)$, we have $\mathcal{W}_{\mathsf{m}}(t)\big|_A > \mathcal{W}_{\mathsf{m}}(s)\big|_A$, and by Lemma 4.9, we get $\mathcal{W}_{\mathsf{m}}(t\theta) > \mathcal{W}_{\mathsf{m}}(s\theta)$. By definition of W_{g} and W_{m}, they coincide on ground preterms, and thus $\mathcal{W}_{\mathsf{g}}(t\theta) > \mathcal{W}_{\mathsf{g}}(s\theta)$. So, rule 1 of $\succ_{\mathsf{g\lambda kbo}}$ applies.

RULES 2, 3, 4: We have $\mathcal{W}_{\mathsf{m}}(t) \geq \mathcal{W}_{\mathsf{m}}(s)$. If $\mathcal{W}_{\mathsf{m}}(t)\big|_A > \mathcal{W}_{\mathsf{m}}(s)\big|_A$, rule 1 applies as above. Otherwise, $\mathcal{W}_{\mathsf{m}}(t)\big|_A = \mathcal{W}(s)\big|_A$, and the corresponding rule 2, 3, or 4 applies. The only mismatches between the two definitions are the use of $\succ_{\mathsf{m\lambda kbo}}$ versus $\succ_{\mathsf{g\lambda kbo}}$ and $\gtrsim^{\mathsf{lex}}_{\mathsf{m\lambda kbo}}$ versus $\succ^{\mathsf{lex}}_{\mathsf{g\lambda kbo}}$. These are repaired by the induction hypothesis and our observation above.

For the second claim, we make a case distinction on the rule deriving $t \succ_{\mathsf{m\lambda kbo}} s$:

Rule 1: As for $\succ_{\mathsf{m\lambda kbo}}$ above.

Rule 2: We will focus on the case where the argument lists $\bar{t}$ and $\bar{s}$ have length 1 and the corresponding De Bruijn index in $y\theta$ occurs exactly once. The same line of reasoning can be repeated for further arguments or further De Bruijn index occurrences by appealing to the transitivity of $\succsim_{\mathsf{g\lambda kbo}}$.

Since t_1 is of nonfunctional type, $(y\,t_1)\theta$ and $(y\,s_1)\theta$ must be of the forms $t' = u\langle\!\langle t_1\theta{\uparrow}^k\rangle\!\rangle$ and $s' = u\langle\!\langle s_1\theta{\uparrow}^k\rangle\!\rangle$, respectively, where k is the context's depth. We have $t_1 \succsim_{\mathsf{m\lambda lpo}} s_1$ by the rule's condition. By the induction hypothesis, $t_1\theta \succeq_{\mathsf{g\lambda kbo}} s_1\theta$. Thus, either $t' = s'$ or, by Theorem 3.14, $t' \succ_{\mathsf{g\lambda kbo}} s'$.

Rules 3, 4, 5: We have $\mathcal{W}_{\mathsf{m}}(t) \geq \mathcal{W}_{\mathsf{m}}(s)$. If $\mathcal{W}_{\mathsf{m}}(t)\big|_A > \mathcal{W}_{\mathsf{m}}(s)\big|_A$, rule 1 applies as above. Otherwise, $\mathcal{W}_{\mathsf{m}}(t)\big|_A = \mathcal{W}_{\mathsf{m}}(s)\big|_A$. If $t\theta = s\theta$, there is nothing to prove. Otherwise, the corresponding rule 2, 3, or 4 applies. The rest of the proof is as for $\succ_{\mathsf{m\lambda kbo}}$ above. ☐

The converse of Theorem 4.10 does not hold. However, it does hold on ground preterms:

Theorem 4.11. *The relation $\succ_{\mathsf{m\lambda kbo}}$ coincides with $\succ_{\mathsf{g\lambda kbo}}$ on ground preterms.*

Proof. One direction of the equivalence follows by Theorem 4.10. It remains to show that $t \succ_{\mathsf{g\lambda kbo}} s$ implies $t \succ_{\mathsf{m\lambda kbo}} s$. The proof is by induction on the definition of $\succ_{\mathsf{g\lambda kbo}}$. It is easy to see that to every case in the definition of $\succ_{\mathsf{g\lambda kbo}}$ corresponds a case in the definition of $\succ_{\mathsf{m\lambda kbo}}$. As for the weights, $\mathcal{W}_{\mathsf{g}}$ and $\mathcal{W}_{\mathsf{m}}$ coincide. In particular, for a ground preterm, the polynomial returned by $\mathcal{W}_{\mathsf{m}}$ contains no indeterminates. To account for the mismatch between $\succ^{\mathsf{lex}}_{\mathsf{g\lambda kbo}}$ and $\succsim^{\mathsf{lex}}_{\mathsf{m\lambda kbo}}$, we apply Lemma 4.6. ☐

Theorem 4.12. *The relation $\succsim_{\mathsf{m\lambda kbo}}$ coincides with $\succeq_{\mathsf{g\lambda kbo}}$ on ground preterms.*

Proof. One direction of the equivalence follows by Theorem 4.10. It remains to show that $t \succ_{\mathsf{g\lambda kbo}} s$ implies $t \succsim_{\mathsf{m\lambda kbo}} s$. If $t = s$, Lemma 4.6 applies. Otherwise, we appeal to Theorem 4.11 to obtain $t \succ_{\mathsf{m\lambda kbo}} s$. By Lemma 4.7, this implies $t \succsim_{\mathsf{m\lambda kbo}} s$. ☐

Lemma 4.13. *If $t \succsim_{\mathsf{m\lambda kbo}} s$, then $\mathcal{W}_{\mathsf{m}}(t) \geq \mathcal{W}_{\mathsf{m}}(s)$.*

Proof. We proceed by structural induction on t.

For all rules except rule 2, the claim is obvious. If $t \succsim_{\mathsf{m\lambda kbo}} s$ was derived by rule 2, then t is of the form $y\,\bar{t}$ and s is of the form $y\,\bar{s}$, and for every i, t_i is steady and $t_i \succsim_{\mathsf{m\lambda kbo}} s_i$. By the induction hypothesis, $\mathcal{W}_{\mathsf{m}}(t_i) \geq \mathcal{W}_{\mathsf{m}}(s_i)$ for every i. Let υ_i be the type of t_i, which is also the type of s_i. Let n be the length of $\bar{t}$, which is also the length of $\bar{s}$. Then,

$$
\begin{aligned}
\mathcal{W}_{\mathsf{m}}(t) &= 1 + \mathbf{w}_y + \sum\nolimits_{i=1}^{n} \mathbf{k}_{y,i}(\mathcal{W}_{\mathsf{m}}(t_i) - \mathcal{w}_{\mathsf{db}}) \\
&\geq 1 + \mathbf{w}_y + \sum\nolimits_{i=1}^{n} \mathbf{k}_{y,i}(\mathcal{W}_{\mathsf{m}}(s_i) - \mathcal{w}_{\mathsf{db}}) \\
&= \mathcal{W}_{\mathsf{m}}(s)
\end{aligned}
$$

☐

Theorem 4.14. *If $u \succsim_{\mathsf{m\lambda kbo}} t$ and $t \succsim_{\mathsf{m\lambda kbo}} s$, then $u \succsim_{\mathsf{m\lambda kbo}} s$. If in addition $u \succ_{\mathsf{m\lambda kbo}} t$ or $t \succ_{\mathsf{m\lambda kbo}} s$, then even $u \succ_{\mathsf{m\lambda kbo}} s$.*

Proof. We proceed by well-founded induction on the multiset $\{|u|, |t|, |s|\}$.

If $u \succsim_{\mathsf{m\lambda kbo}} t$ or $t \succsim_{\mathsf{m\lambda kbo}} s$ was derived by rule 1, then rule 1 yields $u \succ_{\mathsf{m\lambda kbo}} s$ by transitivity of weight comparison and Lemma 4.13. So, for the remainder of this proof, we may assume that $\mathcal{W}_{\mathsf{m}}(u) = \mathcal{W}_{\mathsf{m}}(t) = \mathcal{W}_{\mathsf{m}}(s)$.

If $u \succsim_{\mathsf{m\lambda kbo}} t$ was derived by rule 2, then $t \succsim_{\mathsf{m\lambda kbo}} s$ must have been derived by rule 2, too. Then rule 2 also yields $u \succsim_{\mathsf{m\lambda kbo}} s$ by the induction hypothesis. Since all three preterms u, s, t are headed by variables, neither $u \succ_{\mathsf{m\lambda kbo}} t$ nor $t \succ_{\mathsf{m\lambda kbo}} s$ hold, and thus we need not prove $u \succ_{\mathsf{m\lambda kbo}} s$.

If $u \succsim_{\mathsf{m\lambda kbo}} t$ was derived by rule 3a or 3b, then $t \succsim_{\mathsf{m\lambda kbo}} s$ must be derived by rule 3. If $t \succsim_{\mathsf{m\lambda kbo}} s$ was derived by rule 3a or 3b as well, then rule 3a or 3b also yield $u \succsim_{\mathsf{m\lambda kbo}} s$ by the induction hypothesis and by transitivity of the strict first-order KBO $\succ_{\mathsf{ty}}$ on $\mathcal{T}(\Sigma_{\mathsf{ty}}, \emptyset)$. If in addition $u \succ_{\mathsf{m\lambda kbo}} t$ or $t \succ_{\mathsf{m\lambda kbo}} s$, then this must be by rule 2a or 2b. Then rule 2a or 2b yield $u \succ_{\mathsf{m\lambda kbo}} s$ by the induction hypothesis and by transitivity of $\succ_{\mathsf{ty}}$. If $t \succsim_{\mathsf{m\lambda kbo}} s$ was derived by rule 3c, then rule 2c yields $u \succ_{\mathsf{m\lambda kbo}} s$.

If $u \succsim_{\mathsf{m\lambda kbo}} t$ was derived by rule 3c, then $t \succsim_{\mathsf{m\lambda kbo}} s$ must be derived by rule 4 or 5. Then 2c yields $u \succ_{\mathsf{m\lambda kbo}} s$.

If $u \succsim_{\mathsf{m\lambda kbo}} t$ was derived by rule 4a or 4b, then $t \succsim_{\mathsf{m\lambda kbo}} s$ must be derived by rule 4. If $t \succsim_{\mathsf{m\lambda kbo}} s$ was derived by rule 4a or 4b as well, then rule 4a or 4b also yield $u \succsim_{\mathsf{m\lambda kbo}} s$ by transitivity of $>$ on natural numbers and by the induction hypothesis, which implies transitivity of $\succsim^{\mathsf{lex}}_{\mathsf{m\lambda kbo}}$ on the relevant preterms. If in addition $u \succ_{\mathsf{m\lambda kbo}} t$ or $t \succ_{\mathsf{m\lambda kbo}} s$, then this must be by rule 3a or 3b. Then rule 3a or 3b yield $u \succ_{\mathsf{m\lambda kbo}} s$ by by transitivity of $>$ on natural numbers and the induction hypothesis. If $t \succsim_{\mathsf{m\lambda kbo}} s$ was derived by rule 4c, then rule 3c yields $u \succ_{\mathsf{m\lambda kbo}} s$.

If $u \succsim_{\mathsf{m\lambda kbo}} t$ was derived by rule 4c, then $t \succsim_{\mathsf{m\lambda kbo}} s$ must be derived by rule 5. Then rule 3c yields $u \succ_{\mathsf{m\lambda kbo}} s$.

If $u \succsim_{\mathsf{m\lambda kbo}} t$ was derived by rule 5, then $t \succsim_{\mathsf{m\lambda kbo}} s$ must be derived by rule 5, too. Then rule 5 also yields $u \succsim_{\mathsf{m\lambda kbo}} s$ by transitivity of the precedence $>$, by transitivity of $\succ_{\mathsf{ty}}$ and its lexicographic extension, and by the induction hypothesis, which implies transitivity of $\succsim^{\mathsf{lex}}_{\mathsf{m\lambda kbo}}$ on the relevant preterms. If in addition $u \succ_{\mathsf{m\lambda kbo}} t$ or $t \succ_{\mathsf{m\lambda kbo}} s$, then this must be by rule 4. Then rule 4 yields $u \succ_{\mathsf{m\lambda kbo}} s$ by transitivity of the precedence $>$, by transitivity of $\succ_{\mathsf{ty}}$ and its lexicographic extension, and by the induction hypothesis. □

Theorem 4.15. *Let $t \succ_{\mathsf{m\lambda kbo}} s$. Let θ be a substitution such that all variables in $t\theta$ and $s\theta$ are nonfunctional. Let $s\theta$ contain a nonfunctional variable x outside of parameters. Then $t\theta$ must also contain x outside of parameters.*

Proof. Since $t \succ_{\mathsf{m\lambda kbo}} s$, we have $\mathcal{W}_{\mathsf{m}}(t) \geq \mathcal{W}_{\mathsf{m}}(s)$. By Lemma 4.9, we have $\mathcal{W}_{\mathsf{m}}(t\theta) \geq \mathcal{W}_{\mathsf{m}}(s\theta)$. By definition of $\mathcal{W}_{\mathsf{m}}$, since all variables in $s\theta$ are nonfunctional, $W_{\mathsf{m}}(s\theta)$ must contain $\mathbf{w}_x$ with a nonzero coefficient. Since $\mathcal{W}_{\mathsf{m}}(t\theta) \geq \mathcal{W}_{\mathsf{m}}(s\theta)$, $t\theta$ must also contain $\mathbf{w}_x$ with a nonzero coefficient. Therefore, x must occur outside of parameters in $t\theta$. □

4.2. **λLPO.**

Definition 4.16. Let $>^{\mathsf{ty}}$ be a precedence on Σ_{ty}. Let $\succ_{\mathsf{ty}}$ be the strict first-order LPO on $\mathcal{T}(\Sigma_{\mathsf{ty}}, \emptyset)$ induced by $>^{\mathsf{ty}}$. Let $>$ be a precedence on Σ. Let $\mathsf{ws} \in \Sigma$ be the watershed.

The *strict monomorphic λLPO* $\succ_{\mathsf{m\lambda lpo}}$ and the *nonstrict monomorphic λLPO* $\succsim_{\mathsf{m\lambda lpo}}$ induced by $>^{\mathsf{ty}}, >$ on $\mathcal{T}^{\infty}_{\mathrm{pre}}(\Sigma_{\mathsf{ty}}, \Sigma, \emptyset, X)$ are defined by mutual induction. The strict relation is defined so that $t \succ_{\mathsf{m\lambda lpo}} s$ if any of these conditions is met:

(1) t is of the form $\mathsf{g}\langle\bar{\upsilon}\rangle(\bar{v})\,\bar{t}_k$ and any of these conditions is met:
  (a) $t_i \succsim_{\mathsf{m\lambda lpo}} s$ for some $i \in \{1, \ldots, k\}$, or
  (b) $s = \mathsf{f}(\bar{u})\,\bar{s}$, $\mathsf{g} > \mathsf{f}$, and $\mathit{chkargs}(t, \bar{s})$, or
  (c) $s = \mathsf{g}\langle\bar{\tau}\rangle(\bar{u})\,\bar{s}$, $\bar{\upsilon} \succ^{\mathsf{lex}}_{\mathsf{ty}} \bar{\tau}$, and $\mathit{chkargs}(t, \bar{s})$, or

   (d) $s = \mathsf{g}\langle\bar{\upsilon}\rangle(\bar{u})\,\bar{s}$, $(\bar{\upsilon}, \bar{t}) \succsim^{\mathsf{lex}}_{\mathsf{m\lambda lpo}} (\bar{u}, \bar{s})$, and $\mathit{chkargs}(t, \bar{s})$, or
   (e) $\mathsf{g} > \mathsf{ws}$ and s is either of the form $m\,\bar{s}$ and $\mathit{chkargs}(t, \bar{s})$ or of the form $\lambda\,s'$ and $\mathit{chkargs}(t, [s'])$;

(2) t is of the form $n\,\bar{t}_k$ and any of these conditions is met:
   (a) $t_i \succsim_{\mathsf{m\lambda lpo}} s$ for some $i \in \{1, \ldots, k\}$, or
   (b) $s = m\,\bar{s}$, $n > m$, and $\mathit{chkargs}(t, \bar{s})$, or
   (c) $s = n\,\bar{s}$, $\bar{t} \succsim^{\mathsf{lex}}_{\mathsf{m\lambda lpo}} \bar{s}$, and $\mathit{chkargs}(t, \bar{s})$, or
   (d) s is of the form $\lambda\,s'$ and $\mathit{chkargs}(t, [s'])$ or of the form or $\mathsf{f}(\bar{u})\,\bar{s}$, where $\mathsf{f} \le \mathsf{ws}$, and $\mathit{chkargs}(t, \bar{s})$;

(3) t is of the form $\lambda\langle\upsilon\rangle\,t'$ and any of these conditions is met:
   (a) $t' \succsim_{\mathsf{m\lambda lpo}} s$, or
   (b) $s = \lambda\langle\tau\rangle\,s'$, $\upsilon \succ_{\mathsf{ty}} \tau$, and $\mathit{chkargs}(t, [s'])$, or
   (c) $s = \lambda\langle\upsilon\rangle\,s'$ and $t' \succ_{\mathsf{m\lambda lpo}} s'$, or
   (d) s is of the form $\mathsf{f}(\bar{u})\,\bar{s}$, where $\mathsf{f} \le \mathsf{ws}$, and $\mathit{chkargs}(t, \bar{s})$

where $\mathit{chkargs}(t, \bar{s}_k)$ if and only if $t \succ_{\mathsf{m\lambda lpo}} s_i$ for every $i \in \{1, \ldots, k\}$. The nonstrict relation is defined so that $t \succsim_{\mathsf{m\lambda lpo}} s$ if any of these conditions is met:

(1) t is of the form $y\,\bar{t}$, s is of the form $y\,\bar{s}$, and for every i, t_i is steady and $t_i \succsim_{\mathsf{m\lambda lpo}} s_i$;

(2) t is of the form $\mathsf{g}\langle\bar{\upsilon}\rangle(\bar{v})\,\bar{t}_k$ and any of these conditions is met:
   (a) $t_i \succsim_{\mathsf{m\lambda lpo}} s$ for some $i \in \{1, \ldots, k\}$, or
   (b) $s = \mathsf{f}(\bar{u})\,\bar{s}$, $\mathsf{g} > \mathsf{f}$, and $\mathit{chkargs}(t, \bar{s})$, or
   (c) $s = \mathsf{g}\langle\bar{\tau}\rangle(\bar{u})\,\bar{s}$, $\bar{\upsilon} \succ^{\mathsf{lex}}_{\mathsf{ty}} \bar{\tau}$, and $\mathit{chkargs}(t, \bar{s})$, or
   (d) $s = \mathsf{g}\langle\bar{\upsilon}\rangle(\bar{u})\,\bar{s}$, $(\bar{v}, \bar{t}) \gtrapprox^{\mathsf{lex}}_{\mathsf{m\lambda lpo}} (\bar{u}, \bar{s})$, and $\mathit{chkargs}(t, \bar{s})$, or
   (e) $\mathsf{g} > \mathsf{ws}$ and s is either of the form $m\,\bar{s}$ and $\mathit{chkargs}(t, \bar{s})$ or of the form $\lambda\,s'$ and $\mathit{chkargs}(t, [s'])$;

(3) t is of the form $n\,\bar{t}_k$ and any of these conditions is met:
   (a) $t_i \succsim_{\mathsf{m\lambda lpo}} s$ for some $i \in \{1, \ldots, k\}$, or
   (b) $s = m\,\bar{s}$, $n > m$, and $\mathit{chkargs}(t, \bar{s})$, or
   (c) $s = n\,\bar{s}$, $\bar{t} \gtrapprox^{\mathsf{lex}}_{\mathsf{m\lambda lpo}} \bar{s}$, and $\mathit{chkargs}(t, \bar{s})$, or
   (d) s is of the form $\lambda\,s'$ and $\mathit{chkargs}(t, [s'])$ or of the form or $\mathsf{f}(\bar{u})\,\bar{s}$, where $\mathsf{f} \le \mathsf{ws}$, and $\mathit{chkargs}(t, \bar{s})$;

(4) t is of the form $\lambda\langle\upsilon\rangle\,t'$ and any of these conditions is met:
   (a) $t' \succsim_{\mathsf{m\lambda lpo}} s$, or
   (b) $s = \lambda\langle\tau\rangle\,s'$, $\upsilon \succ_{\mathsf{ty}} \tau$, and $\mathit{chkargs}(t, [s'])$, or
   (c) $s = \lambda\langle\upsilon\rangle\,s'$ and $t' \succsim_{\mathsf{m\lambda lpo}} s'$, or
   (d) s is of the form $\mathsf{f}(\bar{u})\,\bar{s}$, where $\mathsf{f} \le \mathsf{ws}$, and $\mathit{chkargs}(t, \bar{s})$

where $\mathit{chkargs}(t, \bar{s}_k)$ is defined as above.

The only syntactic differences between the definitions of $\succ_{\mathsf{g\lambda lpo}}$ and $\succ_{\mathsf{m\lambda lpo}}$ are that $\succ_{\mathsf{m\lambda lpo}}$ uses $\succsim_{\mathsf{m\lambda lpo}}$ instead of $\succeq_{\mathsf{g\lambda lpo}}$ and $\succsim^{\mathsf{lex}}_{\mathsf{m\lambda lpo}}$ instead of $\succ^{\mathsf{lex}}_{\mathsf{g\lambda lpo}}$. Moreover, rule 1 of $\succsim_{\mathsf{m\lambda lpo}}$ is analogous to rule 2 in the definition of $\succsim_{\mathsf{m\lambda kbo}}$. As for rules 2–4 of $\succsim_{\mathsf{m\lambda lpo}}$, they are nearly identical to the rules defining the strict orders $\succ_{\mathsf{g\lambda lpo}}$ and $\succ_{\mathsf{m\lambda lpo}}$.

Analogous theorems to those about $\succ_{\mathsf{m\lambda kbo}}$ and $\succsim_{\mathsf{m\lambda kbo}}$ also hold about $\succ_{\mathsf{m\lambda lpo}}$ and $\succsim_{\mathsf{m\lambda lpo}}$.

Lemma 4.17. *$s \succsim_{\mathsf{m\lambda lpo}} s$ for every monomorphic preterm s.*

Proof. By straightforward induction on s. □

Lemma 4.18. *If $t \succ_{\mathsf{m\lambda lpo}} s$, then $t \succsim_{\mathsf{m\lambda lpo}} s$.*

Proof. By induction on the derivation of $t \succ_{\mathsf{m\lambda lpo}} s$. For each rule, there is a clearly corresponding rule of $\succsim_{\mathsf{m\lambda lpo}}$ to apply. The induction hypothesis is only required for applying rule 4c. A crucial observation is that $\succsim^{\mathsf{lex}}_{\mathsf{m\lambda lpo}}$ implies $\succapprox^{\mathsf{lex}}_{\mathsf{m\lambda lpo}}$ by definition. □

Theorem 4.19. *If $t \succ_{\mathsf{m\lambda lpo}} s$, then $t\theta \succ_{\mathsf{g\lambda lpo}} s\theta$ for any grounding substitution θ. If $t \succsim_{\mathsf{m\lambda lpo}} s$, then $t\theta \succeq_{\mathsf{g\lambda lpo}} s\theta$ for any grounding substitution θ.*

Proof. The proof of the two claims is by induction on the shape of the derivation of $t \succ_{\mathsf{m\lambda lpo}} s$ and $t \succsim_{\mathsf{m\lambda lpo}} s$. As in the proof of Theorem 4.10, we observe that

– $\bar{t} \succsim^{\mathsf{lex}}_{\mathsf{m\lambda lpo}} \bar{s}$ implies $\bar{t}\theta \succ^{\mathsf{lex}}_{\mathsf{g\lambda lpo}} \bar{s}\theta$, and

– $\bar{t} \succapprox^{\mathsf{lex}}_{\mathsf{m\lambda lpo}} \bar{s}$ implies $\bar{t}\theta \succ^{\mathsf{lex}}_{\mathsf{g\lambda lpo}} \bar{s}\theta$ or $\bar{t}\theta = \bar{s}\theta$.

For the first claim, we make a case distinction on the rule deriving $t \succ_{\mathsf{m\lambda lpo}} s$. In each case, the corresponding rule of $\succ_{\mathsf{g\lambda lpo}}$ applies. The only mismatches between the two definitions are the use of $\succ_{\mathsf{m\lambda kbo}}$ versus $\succ_{\mathsf{g\lambda kbo}}$ and $\succsim^{\mathsf{lex}}_{\mathsf{m\lambda kbo}}$ versus $\succ^{\mathsf{lex}}_{\mathsf{g\lambda kbo}}$. These are repaired by the induction hypothesis and our observation above.

For the second claim, we make a case distinction on the rule deriving $t \succsim_{\mathsf{m\lambda lpo}} s$:

Rule 1: Analogous to the case for rule 2 of $\succsim_{\mathsf{m\lambda kbo}}$ in the proof of Theorem 4.10.

Rules 2, 3, 4: If $t\theta = s\theta$, there is nothing to prove. Otherwise, the corresponding rule 1, 2, or 3 applies. The rest of the proof is as for $\succ_{\mathsf{m\lambda lpo}}$. □

Theorem 4.20. *The relation $\succ_{\mathsf{m\lambda lpo}}$ coincides with $\succ_{\mathsf{g\lambda lpo}}$ on ground preterms.*

Proof. One direction of the equivalence follows by Theorem 4.19. It remains to show that $t \succ_{\mathsf{g\lambda lpo}} s$ implies $t \succ_{\mathsf{m\lambda lpo}} s$. The proof is by induction on the definition of $\succ_{\mathsf{g\lambda lpo}}$. It is easy to see that to every case in the definition of $\succ_{\mathsf{g\lambda lpo}}$ corresponds a case in the definition of $\succ_{\mathsf{m\lambda lpo}}$. To account for the mismatches between $\succeq_{\mathsf{g\lambda kbo}}$ and $\succsim_{\mathsf{m\lambda kbo}}$ and and between $\succ^{\mathsf{lex}}_{\mathsf{g\lambda lpo}}$ and $\succsim^{\mathsf{lex}}_{\mathsf{m\lambda lpo}}$, we apply Lemmas 4.17 and 4.18. □

Theorem 4.21. *The relation $\succsim_{\mathsf{m\lambda lpo}}$ coincides with $\succeq_{\mathsf{g\lambda lpo}}$ on ground preterms.*

Proof. One direction of the equivalence follows by Theorem 4.19. It remains to show that $t \succ_{\mathsf{g\lambda lpo}} s$ implies $t \succsim_{\mathsf{m\lambda lpo}} s$. If $t = s$, Lemma 4.17 applies. Otherwise, we appeal to Theorem 4.20 to obtain $t \succ_{\mathsf{m\lambda lpo}} s$. By Lemma 4.18, this implies $t \succsim_{\mathsf{m\lambda lpo}} s$. □

Theorem 4.22. *If $u \succsim_{\mathsf{m\lambda lpo}} t$ and $t \succsim_{\mathsf{m\lambda lpo}} s$, then $u \succsim_{\mathsf{m\lambda lpo}} s$. If in addition $u \succ_{\mathsf{m\lambda lpo}} t$ or $t \succ_{\mathsf{m\lambda lpo}} s$, then even $u \succ_{\mathsf{m\lambda lpo}} s$.*

Proof. We proceed by well-founded induction on the multiset $\{|u|, |t|, |s|\}$.

If $u \succsim_{\mathsf{m\lambda lpo}} t$ was derived by rule 1, then $t \succsim_{\mathsf{m\lambda lpo}} s$ must have been derived by rule 1, too. Then rule 1 also yields $u \succsim_{\mathsf{m\lambda lpo}} s$ by the induction hypothesis.

If $u \succsim_{\mathsf{m\lambda lpo}} t$ was derived by rule 2a, 3a, or 4a, then $u \succ_{\mathsf{m\lambda lpo}} s$ by rule 1a, 2a, or 3a and the induction hypothesis.

If $u \succsim_{\mathsf{m\lambda lpo}} t$ was derived by rule 2b, 2c, or 2d, then $t \succsim_{\mathsf{m\lambda lpo}} s$ must have been derived by rule 2, too. If $t \succsim_{\mathsf{m\lambda lpo}} s$ was derived by rule 2a, then the *chkargs*-condition and the induction hypothesis yield $u \succ_{\mathsf{m\lambda lpo}} s$. If $t \succsim_{\mathsf{m\lambda lpo}} s$ was derived by rule 2b, 2c, or 2d, then rule 2b, 2c, or 2d also yield $u \succsim_{\mathsf{m\lambda lpo}} s$ by transitivity of the precedence $>$, by transitivity of $\succ_{\mathsf{ty}}$ and its lexicographic extension, and by the induction hypothesis, which implies transitivity of $\succapprox^{\mathsf{lex}}_{\mathsf{m\lambda lpo}}$ on the relevant preterms and the required *chkargs*-condition. If moreover $u \succ_{\mathsf{m\lambda lpo}} t$

or $t \succ_{\mathsf{m\lambda lpo}} s$, we can similarly derive $u \succ_{\mathsf{m\lambda lpo}} s$ by rule 1b, 1c, or 1d. If $t \gtrsim_{\mathsf{m\lambda lpo}} s$ was derived by rule 2e, then $u \succ_{\mathsf{m\lambda lpo}} s$ by rule 1e, using transitivity of the precedence $>$ and the induction hypothesis.

If $u \gtrsim_{\mathsf{m\lambda lpo}} t$ was derived by rule 2e, then $t \gtrsim_{\mathsf{m\lambda lpo}} s$ must have been derived by rule 3 or 4. If $t \gtrsim_{\mathsf{m\lambda lpo}} s$ was derived by rule 3a or 4a, then the *chkargs*-condition and the induction hypothesis yield $u \succ_{\mathsf{m\lambda lpo}} s$. If $t \gtrsim_{\mathsf{m\lambda lpo}} s$ was derived by rule 3b 3c, 3d, 4b, 4c, or 4d, then s is of the form $m\,\bar{u}$, $\lambda\, u'$ or $\mathsf{f}(\bar{u})\,\bar{v}$, where $\mathsf{f} \leq \mathsf{ws}$. If it is of the form $m\,\bar{u}$ or $\lambda\, u'$, then rule 1e yields $u \succ_{\mathsf{m\lambda lpo}} s$, where the *chkargs*-condition is satisfied by the induction hypothesis. If it is of the form $\mathsf{f}(\bar{u})\,\bar{v}$ with $\mathsf{f} \leq \mathsf{ws}$, rule 1b yields $u \succ_{\mathsf{m\lambda lpo}} s$, using transitivity of the precedence $>$ and the induction hypothesis.

If $u \gtrsim_{\mathsf{m\lambda lpo}} t$ was derived by rule 3b or 3c, then $t \gtrsim_{\mathsf{m\lambda lpo}} s$ must have been derived by rule 3. If $t \gtrsim_{\mathsf{m\lambda lpo}} s$ was derived by rule 3a, then the *chkargs*-condition and the induction hypothesis yield $u \succ_{\mathsf{m\lambda lpo}} s$. If $t \gtrsim_{\mathsf{m\lambda lpo}} s$ was derived by rule 3b or 3c, then rule 3b or 3c also yield $u \gtrsim_{\mathsf{m\lambda lpo}} s$, using transitivity of $>$ on natural numbers and the induction hypothesis, which implies transitivity of $\gtrsim^{\mathsf{lex}}_{\mathsf{m\lambda lpo}}$ on the relevant preterms and the required *chkargs*-condition. If moreover $u \succ_{\mathsf{m\lambda lpo}} t$ or $t \succ_{\mathsf{m\lambda lpo}} s$, we can similarly derive $u \succ_{\mathsf{m\lambda lpo}} s$ by rule 2b or 2c. If $t \gtrsim_{\mathsf{m\lambda lpo}} s$ was derived by rule 3d, then rule 2d yields $u \succ_{\mathsf{m\lambda lpo}} s$, using the induction hypothesis to discharge the *chkargs*-condition.

If $u \gtrsim_{\mathsf{m\lambda lpo}} t$ was derived by rule 3d, then $t \gtrsim_{\mathsf{m\lambda lpo}} s$ must have been derived by rule 2 (but not rule 2e) or 4. If $t \gtrsim_{\mathsf{m\lambda lpo}} s$ was derived by rule 2a or 4a, then the *chkargs*-condition and the induction hypothesis yield $u \succ_{\mathsf{m\lambda lpo}} s$. If $t \gtrsim_{\mathsf{m\lambda lpo}} s$ was derived by rule 2b, 2c, 2d, 4b, 4c, or 4d, then rule 2d yields $u \succ_{\mathsf{m\lambda lpo}} s$, using transitivity of the precedence $>$ and the induction hypothesis.

If $u \gtrsim_{\mathsf{m\lambda lpo}} t$ was derived by rule 4b or 4c, then $t \gtrsim_{\mathsf{m\lambda lpo}} s$ must have been derived by rule 4. If $t \gtrsim_{\mathsf{m\lambda lpo}} s$ was derived by rule 4a, then the *chkargs*-condition and the induction hypothesis yield $u \succ_{\mathsf{m\lambda lpo}} s$. If $t \gtrsim_{\mathsf{m\lambda lpo}} s$ was derived by rule 4b or 4c then rule 4b or 4c yield $u \gtrsim_{\mathsf{m\lambda lpo}} s$, using transitivity of $\succ_{\mathsf{ty}}$ and the induction hypothesis. If moreover $u \succ_{\mathsf{m\lambda lpo}} t$ or $t \succ_{\mathsf{m\lambda lpo}} s$, we can similarly derive $u \succ_{\mathsf{m\lambda lpo}} s$ by rule 3b or 3c. If $t \gtrsim_{\mathsf{m\lambda lpo}} s$ was derived by rule 4d, then rule 3d yields $u \succ_{\mathsf{m\lambda lpo}} s$, using the induction hypothesis to discharge the *chkargs*-condition.

If $u \gtrsim_{\mathsf{m\lambda lpo}} t$ was derived by rule 4d, then $t \gtrsim_{\mathsf{m\lambda lpo}} s$ must have been derived by rule 2 (but not rule 2e). If $t \gtrsim_{\mathsf{m\lambda lpo}} s$ was derived by rule 2a, then the *chkargs*-condition and the induction hypothesis yield $u \succ_{\mathsf{m\lambda lpo}} s$. If $t \gtrsim_{\mathsf{m\lambda lpo}} s$ was derived by rule 2b, 2c, or 2d, then rule 3d yields $u \succ_{\mathsf{m\lambda lpo}} s$, using transitivity of the precedence $>$ and the induction hypothesis. □

Theorem 4.23. *Let $t \succ_{\mathsf{m\lambda lpo}} s$ or $t \gtrsim_{\mathsf{m\lambda lpo}} s$. Let θ be a substitution such that all variables in $t\theta$ and $s\theta$ are nonfunctional. Let $s\theta$ contain a nonfunctional variable x outside of parameters. Then $t\theta$ must also contain x outside of parameters.*

Proof. The proof of the two claims is by induction on the shape of the derivation of $t \succ_{\mathsf{m\lambda lpo}} s$ or $t \gtrsim_{\mathsf{m\lambda lpo}} s$. In most cases, the claims follow directly form the induction hypothesis. For case 1, we have $t = y\,\bar{t}$ and $s = y\,\bar{s}$ and for every i, t_i is steady and $t_i \gtrsim_{\mathsf{m\lambda lpo}} s_i$. Our assumption is that $s\theta$ contains a nonfunctional variable x outside of parameters. The x could originate from $y\theta$ or from $s_i\theta$ for some i. If it originates from $y\theta$, then x must also occur outside of parameters in $t\theta = y\theta\,\bar{t}\theta$ because t_i is steady for all i. If it originates from $s_i\theta$, then x must also occur in $t_i\theta$ outside of parameters by the induction hypothesis because $t_i \gtrsim_{\mathsf{m\lambda lpo}} s_i$. Since t_i is steady, x must also occur in $t\theta$ outside of parameters. □

5. The Polymorphic Level

In a third and final step, we generalize the definition of λKBO and λLPO to polymorphic nonground preterms. The resulting orders coincide with the monomorphic nonground λKBO and λLPO on monomorphic preterms while supporting type variables.

Type variables, in conjunction with the η-long β-normal form, lead to substantial complications. Instantiating a type variable with a functional type causes η-expansion to take place, transforming for instance $y\langle\alpha\rangle$ into $\lambda y\langle\beta \to \gamma\rangle 0$ or even $\lambda\lambda y\langle\beta \to (\beta \to \beta) \to \gamma\rangle 1(\lambda 10)$. This affects the weight calculation of λKBO, since each η-expansion increases the weight by $\mathcal{w}_\lambda + \mathcal{w}_{\mathsf{db}}$. Our solution is to add a term to the polynomial to account for possible η-expansion. This also affects the shape comparison of λKBO and λLPO, since the shape of any preterm whose type is a type variable α can change radically as a result of instantiating α.

If we used the η-short β-normal form instead, we would be out of the frying pan into the fire. Applying a λ-abstraction to an argument makes not only the λ but also De Bruijn indices disappear. Applying an η-reduced functional term t to an argument, however, makes neither a λ nor De Bruijn indices disappear, resulting in a weight discrepancy of at least $\mathcal{w}_\lambda$ compared with a λ-abstraction of the same type and weight. Moreover, the η-short normal form makes it more difficult, if not impossible, to achieve another of our goals, namely, the order requirement for the diff symbol of the optimistic λ-superposition calculus.

5.1. $\boldsymbol{\lambda}$KBO.

Definition 5.1. Let $(\Sigma_{\mathsf{ty}}, \Sigma)$ be a higher-order signature. We denote by $\mathbf{P}$ the set of $\mathbf{O}$-valued polynomials of the indeterminates $\mathbf{w}_{y\,\bar{t}}$, $\mathbf{k}_{y\,\bar{t},i}$, and $\mathbf{h}_\alpha$. The first two are the same as in Definition 4.1, except that preterms are now polymorphic. The last one is as follows, where $\alpha \in X_{\mathsf{ty}}$:

- $\mathbf{h}_\alpha$, ranging over $\mathbb{N}$, represents the number of η-expansions incurred as a result of instantiating α for one preterm of type α (excluding any subterms).

Auxiliary concepts are defined as in Definition 4.1.

For example, if $\mathsf{c} : \alpha$ and $\alpha\theta = (\kappa \to \kappa) \to \kappa$, then $\mathsf{c}\theta = \lambda\, \mathsf{c}\, (\lambda\, 1\, 0)$. In this case, instantiation caused two η-expansions, including one to a De Bruijn index.

Definition 5.2. Let $\Sigma' = \Sigma \uplus \{k \mid k \in \mathbf{O}_{>0}\}$ with $k : \Pi\alpha.\, \alpha$. Define the normalization function $[\;] : \mathcal{T}^{\infty}_{\mathrm{pre}}(\Sigma_{\mathsf{ty}}, \Sigma, X_{\mathsf{ty}}, X) \to \mathcal{T}^{\infty}_{\mathrm{pre}}(\Sigma_{\mathsf{ty}}, \Sigma', X_{\mathsf{ty}}, X)$ recursively by

$$\begin{aligned}
[y\,\bar{t}] &= y\,[\bar{t}] \\
[\mathsf{f}\langle\bar{u}\rangle\,\bar{t}] &= \begin{cases} k\langle\tau\rangle\,[\bar{t}] & \text{if } \mathcal{k}(\mathsf{f}, i) = 1 \text{ for every } i, \text{ with } \mathcal{w}(\mathsf{f}) = k \text{ and } \mathsf{f}\langle\bar{u}\rangle : \tau \\ \mathsf{f}\langle\bar{u}\rangle\,[\bar{t}] & \text{otherwise} \end{cases} \\
[m\langle\tau\rangle\,\bar{t}] &= \mathcal{w}_{\mathsf{db}}\langle\tau\rangle\,[\bar{t}] \\
[\lambda\, t] &= \lambda\,[t]
\end{aligned}$$

Definition 5.3. Define the η-expansion polynomial $\mathcal{H} : \mathcal{Ty}(\Sigma_{\mathsf{ty}}, X_{\mathsf{ty}}) \to \mathbf{P}$ by

$$\mathcal{H}(\alpha) = (\mathcal{w}_\lambda + \mathcal{w}_{\mathsf{db}})\mathbf{h}_\alpha \qquad\qquad \mathcal{H}(\kappa(\bar{\tau})) = 0$$

Definition 5.4. Let $\mathcal{w} : \Sigma \to \mathbf{O}_{>0}$, $\mathcal{w}_\lambda, \mathcal{w}_{\mathsf{db}} \in \mathbf{O}_{>0}$, and $\mathcal{k} : \Sigma \times \mathbb{N}_{>0} \to \mathbf{O}_{>0}$. For every $\mathsf{f} \in \Sigma$ and $i > \mathit{arity}(\mathsf{f})$, we require (K) $\mathcal{k}(\mathsf{f}, i) = 1$. Given a list of preterms $\bar{t}$, let $\bar{t}^{\star\star}$ denote the longest suffix consisting of steady preterms, and let $\bar{t}^{\star}$ denote the complementary prefix. Define the polymorphic weight function $\mathcal{W} : \mathcal{T}^{\infty}_{\mathrm{pre}}(\Sigma_{\mathsf{ty}}, \Sigma, X_{\mathsf{ty}}, X) \to \mathbf{P}$ recursively by

$$\begin{aligned}
\mathcal{W}(y\,\bar{t}) &= 1 + \mathbf{w}_{y\,[\bar{t}^{\star}]} + \sum\nolimits_{i=1}^{|\bar{t}^{\star\star}|} \mathbf{k}_{y\,[\bar{t}^{\star}],i}(\mathcal{W}(\bar{t}_i^{\star\star}) - \mathcal{w}_{\mathsf{db}}) + \mathcal{H}(\tau) \\
&\quad \text{if } y\,\bar{t} : \tau \\
\mathcal{W}(\mathsf{f}(\bar{u})\,\bar{t}_n) &= \mathcal{w}(\mathsf{f}) + \sum\nolimits_{i=1}^{n} \mathcal{k}(\mathsf{f}, i)\,\mathcal{W}(t_i) + \mathcal{H}(\tau) \\
&\quad \text{if } \mathsf{f}(\bar{u})\,\bar{t}_n : \tau \\
\mathcal{W}(m\,\bar{t}_n) &= \mathcal{w}_{\mathsf{db}} + \sum\nolimits_{i=1}^{n} \mathcal{W}(t_i) + \mathcal{H}(\tau) \\
&\quad \text{if } m\,\bar{t}_n : \tau \\
\mathcal{W}(\lambda\,t) &= \mathcal{w}_{\lambda} + \mathcal{W}(t)
\end{aligned}$$

Notice, in the definition above, the presence of $\mathcal{H}(\tau)$ monomials to account for η-expansion caused by type variable instantiation.

Definition 5.5. Let $\tau, \upsilon \in \mathcal{Ty}(\Sigma_{\mathsf{ty}}, X_{\mathsf{ty}})$. The polymorphism comparison $\upsilon \trianglerighteq \tau$ holds if τ is not a type variable or if $\upsilon = \tau$. Moreover, let $s, t \in \mathcal{T}^{\infty}_{\mathrm{pre}}(\Sigma, \Sigma_{\mathsf{ty}}, X, X_{\mathsf{ty}})$ such that $s : \tau$, $t : \upsilon$. We write $t \trianglerighteq s$ if $\upsilon \trianglerighteq \tau$.

Definition 5.6. Let $\mathcal{w}_{\mathsf{ty}}, \mathcal{w}, \mathcal{w}_{\lambda}, \mathcal{w}_{\mathsf{db}}, \mathcal{k}, \mathcal{W}$ be as in Definition 5.4. Let $>^{\mathsf{ty}}$ be a precedence on Σ_{ty}. Let $\succ_{\mathsf{ty}}$ be the strict first-order KBO on $\mathcal{T}(\Sigma_{\mathsf{ty}}, X_{\mathsf{ty}})$ induced by $\mathcal{w}_{\mathsf{ty}}$ and $>^{\mathsf{ty}}$. Let $>$ be a precedence on Σ.

The *strict polymorphic λKBO* $\succ_{\mathsf{\lambda kbo}}$ induced by $\mathcal{w}_{\mathsf{ty}}, \mathcal{w}, \mathcal{w}_{\lambda}, \mathcal{w}_{\mathsf{db}}, \mathcal{k}, >^{\mathsf{ty}}, >$ on $\mathcal{T}^{\infty}_{\mathrm{pre}}(\Sigma_{\mathsf{ty}}, \Sigma, X_{\mathsf{ty}}, X)$ is defined inductively so that $t \succ_{\mathsf{\lambda kbo}} s$ if

(1) the rule 1, 2a, 2b, 3b, or 4c of the definition of $\succ_{\mathsf{m\lambda kbo}}$ applies mutatis mutandis, or
(2) the rule 2c, 3a, 3c, 4a, or 4b of the definition of $\succ_{\mathsf{m\lambda kbo}}$ applies mutatis mutandis and $t \trianglerighteq s$ holds.

The *nonstrict polymorphic λKBO* $\succsim_{\mathsf{\lambda kbo}}$ induced by $\mathcal{w}_{\mathsf{ty}}, \mathcal{w}, \mathcal{w}_{\lambda}, \mathcal{w}_{\mathsf{db}}, \mathcal{k}, >^{\mathsf{ty}}, >$ on $\mathcal{T}^{\infty}_{\mathrm{pre}}(\Sigma_{\mathsf{ty}}, \Sigma, X_{\mathsf{ty}}, X)$ is defined inductively so that $t \succsim_{\mathsf{\lambda kbo}} s$ if

(1) the rule 1, 2, 3a, 3b, 4b, or 5c of the definition of $\succsim_{\mathsf{m\lambda kbo}}$ applies mutatis mutandis, or
(2) the rule 3c, 4a, 4c, 5a, or 5b of the definition of $\succsim_{\mathsf{m\lambda kbo}}$ applies mutatis mutandis and $t \trianglerighteq s$ holds.

For some of the rules, the condition $t \trianglerighteq s$ is necessary to compare t and s. For the other rules, the condition can be derived from the types of t and s, either because both are of nonvariable type or because they are of the same type.

Below we will connect the polymorphic λKBO with its monomorphic counterpart to lift its properties, which in turn were lifted from the ground λKBO.

Definition 5.7. The polynomial substitution $\mathcal{poly}(\theta)$ associated with a monomorphizing type substitution θ maps indeterminate $\mathbf{w}_{y\,\bar{t}}$ to $\mathbf{w}_{(y\,\bar{t})\theta}$, indeterminate $\mathbf{k}_{y\,\bar{t},i}$ to $\mathbf{k}_{(y\,\bar{t})\theta,i}$, and indeterminate $\mathbf{h}_{\alpha}$ to the number of η-expansions incurred as a result of instantiating α for one preterm of type α (excluding any subterms).

Lemma 5.8. *Given a monomorphizing type substitution θ, we have $\mathcal{W}(t)\big|_{\mathcal{poly}(\theta)} = \mathcal{W}_{\mathsf{m}}(t\theta)$.*

Proof. Let $\sigma = \mathcal{poly}(\theta)$. Let $t : \tau$, and let k be the number of η-expansions incurred as a result of applying θ on a term of type τ (excluding any subterms). The proof is by induction on the definition of $\mathcal{W}$.

CASE $t = y\,\bar{t}$: We have

$$
\begin{aligned}
&\mathcal{W}(y\,\bar{t})\big|_\sigma \\
&= 1 + \mathbf{w}_{y\,\bar{t}^{\,\star}}\big|_\sigma + \sum_{i=1}^{|\bar{t}^{\,\star\star}|} \mathbf{k}_{y\,\bar{t}^{\,\star},i}\big|_\sigma (\mathcal{W}(\bar{t}_i^{\star\star})\big|_\sigma - \mathcal{w}_{\mathsf{db}}) + \mathcal{H}(\tau)\big|_\sigma \\
&\qquad \text{by definition of } \mathcal{W} \\
&= 1 + \mathbf{w}_{y\,\bar{t}^{\,\star}}\big|_\sigma + \sum_{i=1}^{|\bar{t}^{\,\star\star}|} \mathbf{k}_{y\,\bar{t}^{\,\star},i}\big|_\sigma (\mathcal{W}_{\mathsf{m}}(\bar{t}_i^{\star\star}\theta) - \mathcal{w}_{\mathsf{db}}) + \mathcal{H}(\tau)\big|_\sigma \\
&\qquad \text{by the induction hypothesis} \\
&= 1 + \mathbf{w}_{y\,\bar{t}^{\,\star}}\big|_\sigma + \sum_{i=1}^{|\bar{t}^{\,\star\star}|} \mathbf{k}_{y\,\bar{t}^{\,\star},i}\big|_\sigma (\mathcal{W}_{\mathsf{m}}(\bar{t}_i^{\star\star}\theta) - \mathcal{w}_{\mathsf{db}}) + (\mathcal{w}_\lambda + \mathcal{w}_{\mathsf{db}})k \\
&\qquad \text{by definition of } \mathcal{H} \text{ and the semantics of } \mathbf{h} \\
&= 1 + \mathbf{w}_{(y\theta)\,(\bar{t}\theta)^{\,\star}} + \sum_{i=1}^{|(\bar{t}\theta)^{\,\star\star}|} \mathbf{k}_{(y\theta)\,(\bar{t}\theta)^{\,\star},i} (\mathcal{W}_{\mathsf{m}}(\bar{t}_i^{\star\star}\theta) - \mathcal{w}_{\mathsf{db}}) + (\mathcal{w}_\lambda + \mathcal{w}_{\mathsf{db}})k \\
&\qquad \text{by definition of } \sigma \\
&= \mathcal{W}_{\mathsf{m}}((y\,\bar{t})\theta) \\
&\qquad \text{by definition of } \mathcal{W}_{\mathsf{m}} \text{ and substitution}
\end{aligned}
$$

CASE $t = \mathsf{f}\langle\bar{u}\rangle\,\bar{t}_n$: We have

$$
\begin{aligned}
&\mathcal{W}(\mathsf{f}\langle\bar{u}\rangle\,\bar{t}_n)\big|_\sigma \\
&= \mathcal{w}(\mathsf{f}) + \sum_{i=1}^{n} \mathcal{k}(\mathsf{f},i)\,\mathcal{W}(t_i)\big|_\sigma + \mathcal{H}(\tau)\big|_\sigma && \text{by definition of } \mathcal{W} \\
&= \mathcal{w}(\mathsf{f}) + \sum_{i=1}^{n} \mathcal{k}(\mathsf{f},i)\,\mathcal{W}_{\mathsf{m}}(t_i\theta) + \mathcal{H}(\tau)\big|_\sigma && \text{by the induction hypothesis} \\
&= \mathcal{w}(\mathsf{f}) + \sum_{i=1}^{n} \mathcal{k}(\mathsf{f},i)\,\mathcal{W}_{\mathsf{m}}(t_i\theta) + (\mathcal{w}_\lambda + \mathcal{w}_{\mathsf{db}})k && \text{by definition of } \mathcal{H} \\
&= \mathcal{W}_{\mathsf{m}}((\mathsf{f}\langle\bar{u}\rangle\,\bar{t}_n)\theta) && \text{by definition of } \mathcal{W}_{\mathsf{m}}\text{, substitution, and hypothesis (K)}
\end{aligned}
$$

CASE $t = m\,\bar{t}_n$: Similar to the previous case.

CASE $t = \lambda\,t$: We have

$$
\begin{aligned}
&\mathcal{W}(\lambda\,t)\big|_\sigma \\
&= \mathcal{w}_\lambda + \mathcal{W}(t)\big|_\sigma && \text{by definition of } \mathcal{W} \\
&= \mathcal{w}_\lambda + \mathcal{W}_{\mathsf{m}}(t\theta) && \text{by the induction hypothesis} \\
&= \mathcal{W}_{\mathsf{m}}(\lambda\,(t\theta)) && \text{by definition of } \mathcal{W}_{\mathsf{m}} \\
&= \mathcal{W}_{\mathsf{m}}((\lambda\,t)\theta) && \text{by definition of substitution}
\end{aligned}
$$

□

Definition 5.9. Let σ be a substitution. Given a preterm t, let $t\sigma?$ denote its *truncating substitution*, in which any outermost λs introduced due to η-expansion as a result of applying σ to t are omitted. (In contrast, any introduced De Bruijn indices are kept.)

For example, if $\mathsf{c} : \alpha$ and $\alpha\sigma = \kappa \to \kappa \to \kappa$, then $\mathsf{c}\sigma? = \mathsf{c}\,1\,0$, whereas $\mathsf{c}\sigma = \lambda\,\lambda\,\mathsf{c}\,1\,0$.

Theorem 5.10. *If $t \succ_{\lambda\mathsf{kbo}} s$, then $t\theta \succ_{\mathsf{m}\lambda\mathsf{kbo}} s\theta$ for any monomorphizing type substitution θ. If $t \succsim_{\lambda\mathsf{kbo}} s$, then $t\theta \succsim_{\mathsf{m}\lambda\mathsf{kbo}} s\theta$ for any monomorphizing type substitution θ.*

Proof. The proof of the two claims is by induction on the shape of the derivation of $t \succ_{\lambda\mathsf{kbo}} s$ and $t \succsim_{\lambda\mathsf{kbo}} s$.

For the first claim, we proceed by case distinction on the rule deriving $t \succ_{\mathsf{\lambda kbo}} s$:
RULE 1: From $\mathcal{W}(t) > \mathcal{W}(s)$, by Lemma 5.8, we have $\mathcal{W}_{\mathsf{m}}(t\theta_1) > \mathcal{W}_{\mathsf{m}}(s\theta_1)$. Thus, rule 1 of $\succ_{\mathsf{m\lambda kbo}}$ applies.

RULES 2, 3, 4: We have $\mathcal{W}(t) \geq \mathcal{W}(s)$. We also have $t \trianglerighteq s$, either because the rule requires it or because it follows from the types of t and s. We perform a case analysis on $t \trianglerighteq s$.

SUBCASE 1, where t and s are of nonvariable types: The corresponding rule 2, 3, or 4 for $\succ_{\mathsf{m\lambda kbo}}$ applies. The only mismatch between the two definitions is the use of $\succ_{\mathsf{\lambda kbo}}$ versus $\succ_{\mathsf{m\lambda kbo}}$, and it is repaired by the induction hypothesis.

SUBCASE 2, where t has some variable type α but not s: If $\alpha\theta$ is a function type, then applying θ to t results in some η-expansion, which leads to a heavier weight; rule 1 then applies. Otherwise, $\alpha\theta$ is not a function type, and the reasoning is as for subcase 1.

SUBCASE 3, where t and s have some variable type α: Let k be the number of curried arguments expected by values of type $\alpha\theta$. This means that we have

$$t\theta = \underbrace{\lambda \ldots \lambda}_{k \text{ times}} (t\theta){\uparrow}^k \; (k-1){\uparrow}_\eta \; \ldots \; 0{\uparrow}_\eta \qquad\qquad s\theta = \underbrace{\lambda \ldots \lambda}_{k \text{ times}} (s\theta){\uparrow}^k \; (k-1){\uparrow}_\eta \; \ldots \; 0{\uparrow}_\eta$$

First, we apply rule 2b k times to remove the leading λs on both sides. It remains to show that $t\theta? \succ_{\mathsf{m\lambda kbo}} s\theta?$. By inspection of the rules 3a and 3b of $\succ_{\mathsf{\lambda kbo}}$, we find that $t{\uparrow}^k \succ_{\mathsf{\lambda kbo}} s{\uparrow}^k$. For each of the $\succ_{\mathsf{\lambda kbo}}$ rules that could have been used to establish this, we can check that the corresponding $\succ_{\mathsf{m\lambda kbo}}$ rule is applicable. The additional De Bruijn indices $(k-1){\uparrow}_\eta \; \ldots \; 0{\uparrow}_\eta$ on both sides are harmless.

The proof of the second claim is analogous. □

Theorem 5.11. *The relation $\succ_{\mathsf{\lambda kbo}}$ coincides with $\succ_{\mathsf{m\lambda kbo}}$ on monomorphic preterms. The relation $\succsim_{\mathsf{\lambda kbo}}$ coincides with $\succsim_{\mathsf{m\lambda kbo}}$ on monomorphic preterms.*

Proof. One direction of the equivalences follows by Theorem 5.10. It remains to show that $t \succ_{\mathsf{m\lambda kbo}} s$ implies $t \succ_{\mathsf{\lambda kbo}} s$ and that $t \succsim_{\mathsf{m\lambda kbo}} s$ implies $t \succsim_{\mathsf{\lambda kbo}} s$. The proof is by induction on the definition of $\succ_{\mathsf{m\lambda kbo}}$ and $\succsim_{\mathsf{m\lambda kbo}}$. It is easy to see that to every case in the definition of $\succ_{\mathsf{m\lambda kbo}}$ corresponds a case in the definition of $\succ_{\mathsf{\lambda kbo}}$ and every case in the definition of $\succsim_{\mathsf{m\lambda kbo}}$ corresponds a case in the definition of $\succsim_{\mathsf{\lambda kbo}}$. For the weights, $\mathcal{W}_{\mathsf{m}}$ and $\mathcal{W}$ coincide. In particular, for a monomorphic preterm, the polynomial returned by $\mathcal{W}$ contains no $\mathbf{h}_\alpha$ indeterminates. □

Lemma 5.12. *If $t \succ_{\mathsf{\lambda kbo}} s$, then $t \succsim_{\mathsf{\lambda kbo}} s$.*

Proof. Analogous to Lemma 4.7. □

Theorem 5.13. *If $t \succsim_{\mathsf{\lambda kbo}} u$ and $u \succsim_{\mathsf{\lambda kbo}} s$, then $t \succsim_{\mathsf{\lambda kbo}} s$. If moreover $t \succ_{\mathsf{\lambda kbo}} u$ or $u \succ_{\mathsf{\lambda kbo}} s$, then $t \succ_{\mathsf{\lambda kbo}} s$.*

Proof. Analogous to Theorem 4.14, using the fact that $\trianglerighteq$ is transitive. □

Theorem 5.14. *Let $t \succ_{\mathsf{\lambda kbo}} s$. Let θ be a substitution such that all variables in $t\theta$ and $s\theta$ are nonfunctional term variables. Let $s\theta$ contain a nonfunctional variable x outside of parameters. Then $t\theta$ must also contain x outside of parameters.*

Proof. By Theorems 4.15 and 5.10. □

5.2. λLPO.

Definition 5.15. Let $>^{\mathsf{ty}}$ be a precedence on Σ_{ty}. Let $\succ_{\mathsf{ty}}$ be the strict first-order LPO on $\mathcal{T}(\Sigma_{\mathsf{ty}}, X_{\mathsf{ty}})$ induced by $>^{\mathsf{ty}}$. Let $>$ be a precedence on Σ. Let $\mathsf{ws} \in \Sigma$ be the watershed.

The *strict polymorphic λLPO* $\succ_{\mathsf{\lambda lpo}}$ and the *nonstrict polymorphic λLPO* $\succsim_{\mathsf{\lambda lpo}}$ induced by $>^{\mathsf{ty}}, >$ on $\mathcal{T}^{\infty}_{\mathrm{pre}}(\Sigma_{\mathsf{ty}}, \Sigma, X_{\mathsf{ty}}, X)$ are defined by mutual induction. The strict relation is defined so that $t \succ_{\mathsf{\lambda lpo}} s$ if

(1) the rule 1a, 1d, 1e, 2a, 2c, 3a, 3b, or 3c of the definition of $\succ_{\mathsf{\lambda lpo}}$ applies mutatis mutandi, or
(2) the rule 1b or 1c of the definition of $\succ_{\mathsf{\lambda lpo}}$ applies mutatis mutandi and either $\mathsf{g} > \mathsf{ws}$ or $t \trianglerighteq s$ holds, or
(3) the rule 2b, 2d, or 3d of the definition of $\succ_{\mathsf{\lambda lpo}}$ applies mutatis mutandi and $t \trianglerighteq s$ holds.

The nonstrict relation is defined so that $t \succsim_{\mathsf{\lambda lpo}} s$ if

(1) the rule 1, 2a, 2d, 2e, 3a, 3c, 4a, 4b, or 4c of the definition of $\succsim_{\mathsf{\lambda lpo}}$ applies mutatis mutandi, or
(2) the rule 2b or 2c of the definition of $\succ_{\mathsf{\lambda lpo}}$ applies mutatis mutandi and $\mathsf{g} > \mathsf{ws}$ or $t \trianglerighteq s$ holds, or
(3) the rule 3b, 3d, or 4d of the definition of $\succsim_{\mathsf{\lambda lpo}}$ applies mutatis mutandi and $t \trianglerighteq s$ holds.

Like for λKBO, the conditions $t \trianglerighteq s$ are sometimes necessary to guard against η-expansion on the right-hand side of $\succsim_{\mathsf{\lambda lpo}}$. However, they are not necessary in most cases. Consider the precedence $\mathsf{h} > \mathsf{f} > \mathsf{a}$, with $\mathsf{h} > \mathsf{ws}$, and suppose $\mathsf{a} : \alpha$, $\mathsf{h} : \kappa$. We allow the polymorphic comparison $\mathsf{h} \succ_{\mathsf{\lambda lpo}} \mathsf{a}$ even though instantiating α may lead to η-expansion of a:

$$\mathsf{h} \succ_{\mathsf{\lambda lpo}} \underbrace{\lambda \ldots \lambda}_{k \text{ times}} \mathsf{a}\ (k-1)\ \ldots\ 0$$

The key for this to work is that symbols above the watershed—here, h—are considered larger than both λs and De Bruijn indices.

Theorem 5.16. *If $t \succ_{\mathsf{\lambda lpo}} s$, then $t\theta \succsim_{\mathsf{m\lambda lpo}} t\theta? \succ_{\mathsf{m\lambda lpo}} s\theta$ for any monomorphizing type substitution θ. If $t \succsim_{\mathsf{\lambda lpo}} s$, then $t\theta \succsim_{\mathsf{m\lambda lpo}} s\theta$ for any monomorphizing type substitution θ.*

Proof. As an induction hypothesis, the inequality $t\theta? \succ_{\mathsf{m\lambda lpo}} s\theta$ will be useful to apply rules that have a $\mathit{chkargs}$ condition.

First, we show $t\theta \succsim_{\mathsf{m\lambda lpo}} t\theta?$. The only difference between the two preterms is the presence of k additional λs on the left. If $k = 0$, Lemma 4.17 can be used to establish $t\theta \succsim_{\mathsf{m\lambda lpo}} t\theta?$. Otherwise, the property can be established by applying rule 3a k times.

The proof of the two remaining inequalities is by induction on $|t| + |s|$.

CASES 1A, 2A, 3A OF $\succ_{\mathsf{\lambda lpo}}$: These cases all correspond to "subterm" rules. If t is of nonvariable type, we apply the corresponding rule of $\succ_{\mathsf{m\lambda lpo}}$, relying on the induction hypothesis for the recursive comparison with $\succsim_{\mathsf{m\lambda lpo}}$. Otherwise, suppose t is of type α. Let k be the number of curried arguments expected by values of type $\alpha\theta$. This means that $t\theta?$ is of the form

$$t'{\uparrow^k}\ (k-1){\uparrow_\eta}\ \ldots\ 0{\uparrow_\eta}$$

We apply the rule of $\succ_{\mathsf{m\lambda lpo}}$ corresponding to the rule that was used to establish $t \succ_{\mathsf{\lambda lpo}} s$ in the first place.

For this to work, a recursive comparison must be possible. We show how it can be done for the case of rule 1a of $\succ_{\mathsf{\lambda lpo}}$, where $t = \mathsf{f}(\bar{u})\,\bar{t}$; the other two cases are similar. For rule 1a to have been applicable to establish $\mathsf{f}(\bar{u})\,\bar{t} \succ_{\mathsf{\lambda lpo}} s$, we must have $t_i \succsim_{\mathsf{\lambda lpo}} s$ for some i. Now, to apply rule 1a to derive

$$\mathsf{f}(\bar{u}\theta{\uparrow}^k)\;\bar{t}\theta{\uparrow}^k\;(k-1){\uparrow}_\eta\;\ldots\;0{\uparrow}_\eta \succ_{\mathsf{m\lambda lpo}} s\theta$$

we must show that $t_i\theta{\uparrow}^k \succsim_{\mathsf{m\lambda lpo}} s\theta$. From $t_i \succsim_{\mathsf{\lambda lpo}} s$, the induction hypothesis, and by inspection of rules 3b and 3c of $\succsim_{\mathsf{\lambda lpo}}$, we get $t_i\theta{\uparrow}^k \succsim_{\mathsf{m\lambda lpo}} t_i\theta \succsim_{\mathsf{m\lambda lpo}} s\theta$, as desired.

Rules 1b, 1c, 1d, 1e of $\succ_{\mathsf{\lambda lpo}}$: These cases have a symbol as the head on the left-hand side. We will focus on the case of rule 1b; the other three cases are similar. For rule 1b, $t = \mathsf{g}(\bar{v})\,\bar{t}$ and $s = \mathsf{f}(\bar{u})\,\bar{s}$, with $t \succ_{\mathsf{m\lambda lpo}} s_i$ for every i. We also have either $\mathsf{g} > \mathsf{ws}$ or $t \trianglerighteq s$. We focus on the case where $\mathsf{g} > \mathsf{ws}$; the other case is similar to that of rule 2b, below.

Let $t : \upsilon$ and $s : \tau$. Let l and k be the number of curried arguments expected by values of type $\upsilon\theta$ and $\tau\theta$, respectively. This means that we have

$$t\theta? = \mathsf{g}(\bar{v}\theta{\uparrow}^l)\;\bar{t}\theta{\uparrow}^l\;(l-1){\uparrow}_\eta\;\ldots\;0{\uparrow}_\eta \qquad s\theta = \underbrace{\lambda\ldots\lambda}_{k\ \text{times}}\;\mathsf{f}(\bar{u}\theta{\uparrow}^k)\;\bar{s}\theta{\uparrow}^k\;(k-1){\uparrow}_\eta\;\ldots\;0{\uparrow}_\eta$$

To show $t\theta? \succ_{\mathsf{\lambda lpo}} s\theta$, we apply rule 1e k times to remove the λs on the right. It then suffices to prove $t\theta? \succ_{\mathsf{\lambda lpo}} s\theta?$. We apply rule 1b. For the rule to be applicable, due to the *chkargs* condition we need $t\theta? \succ_{\mathsf{m\lambda lpo}} s_i\theta$ to hold for every i. This follows from $t \succ_{\mathsf{\lambda lpo}} s_i$ and the induction hypothesis. In addition, we need $t\theta? \succ_{\mathsf{m\lambda lpo}} j{\uparrow}_\eta$ for $j \in \{0, \ldots, k-1\}$. This follows from rule 1e.

Rules 2b, 2c of $\succ_{\mathsf{\lambda lpo}}$: These cases compare applied De Bruijn indices $t = n\,\bar{t}$ and $s = m\,\bar{s}$, where $n \geq m$. We also know that $t \succ_{\mathsf{\lambda lpo}} s_i$ for every i. We perform a case analysis on $t \trianglerighteq s$.

Subcase 1, where t and s are of nonvariable types: Rule 2b or 2c of $\succ_{\mathsf{m\lambda lpo}}$ applies. For the rule to be applicable, due to the *chkargs* condition we need $t\theta = t\theta? \succ_{\mathsf{m\lambda lpo}} s_i\theta$ to hold for every i. This follows from $t \succ_{\mathsf{\lambda lpo}} s_i$ and the induction hypothesis.

Subcase 2, where t has some variable type α but not s: If $\alpha\theta$ is nonfunctional, the reasoning is as for subcase 1. Otherwise, $\alpha\theta$ is a function type, and applying θ to t results in $k > 0$ η-expansions. This means we have

$$t\theta? = (n+k)\;(\bar{t}\theta){\uparrow}^k\;(k-1){\uparrow}_\eta\;\ldots\;0{\uparrow}_\eta \qquad\qquad s\theta = m\;\bar{s}\theta$$

Since $n + k > m$, we apply rule 2b to establish $t\theta? \succ_{\mathsf{m\lambda lpo}} s\theta$. For the rule to be applicable, due to the *chkargs* condition $t\theta? \succ_{\mathsf{m\lambda lpo}} s_i\theta$ must hold for every i. This follows from $t \succ_{\mathsf{\lambda lpo}} s_i$ and the induction hypothesis.

Subcase 3, where t and s has some variable type α: Let k be the number of curried arguments expected by values of type $\alpha\theta$. This means that we have

$$t\theta? = (n+k)\;(\bar{t}\theta){\uparrow}^k\;(k-1){\uparrow}_\eta\;\ldots\;0{\uparrow}_\eta \qquad s\theta = \underbrace{\lambda\ldots\lambda}_{k\ \text{times}}\;(m+k)\;(s\theta){\uparrow}^k\;(k-1){\uparrow}_\eta\;\ldots\;0{\uparrow}_\eta$$

and must show $t\theta? \succ_{\mathsf{m\lambda lpo}} s\theta$. First, we apply rule 2d k times to remove the λs on the right. For the rule to be applicable, due to the *chkargs* condition $t\theta? \succ_{\mathsf{m\lambda lpo}} s\theta?$ must hold. To prove it, we apply rule 2b or 2c, depending on whether $n > m$ or $n = m$. For the tuple comparison in rule 2c, it is easy to see that the additional De Bruijn arguments are harmless. For either rule to be applicable, due to the *chkargs* condition we also need $t\theta? \succ_{\mathsf{m\lambda lpo}} s_i\theta$ for every i and $t\theta? \succ_{\mathsf{m\lambda lpo}} j{\uparrow}_\eta$ for every $j \in \{0, \ldots, k-1\}$. The first inequality follows from $t \succ_{\mathsf{\lambda lpo}} s_i$ and the induction hypothesis. The second inequality follows from rule 2a, since one of the arguments in $t\theta?$ is $j{\uparrow}_\eta$.

RULE 2D OF $\succ_{\mathsf{\lambda lpo}}$: This case compares an applied De Bruijn index $t = n\,\bar{t}$ and either a λ-abstraction $\lambda\, s'$ or an applied symbol $\mathsf{f}(\bar{u})\,\bar{s}$ below the watershed. In the λ subcase, we have $t \succ_{\mathsf{m\lambda lpo}} s'$. We apply rule 2d to derive $t\theta? \succ_{\mathsf{m\lambda lpo}} s\theta$. This requires us to prove $t\theta? \succ_{\mathsf{m\lambda lpo}} s'\theta$, which follows from $t \succ_{\mathsf{m\lambda lpo}} s'$ and the induction hypothesis. In the other subcase, the proof is similar to as in cases 2b, 2c of $\succ_{\mathsf{m\lambda lpo}}$ above.

RULE 3B OF $\succ_{\mathsf{\lambda lpo}}$: This case compares two λ-abstractions $t = \lambda\langle\upsilon\rangle\, t'$ and $s = \lambda\langle\tau\rangle\, s'$. We have $t \succ_{\mathsf{\lambda lpo}} s'$. To derive the desired inequality $t\theta? = t\theta \succ_{\mathsf{m\lambda lpo}} s\theta$, we apply rule 3b, which requires us to prove $\upsilon\theta \succ_{\mathsf{ty}} \tau\theta$ and $t\theta \succ_{\mathsf{m\lambda lpo}} s'\theta$. The first inequality follows from $\upsilon \succ_{\mathsf{ty}} \tau$ by stability under substitution of the standard LPO. The second inequality follows from $t \succ_{\mathsf{\lambda lpo}} s'$ and the induction hypothesis.

RULES 3C OF $\succ_{\mathsf{\lambda lpo}}$: These cases compare two λ-abstractions $t = \lambda\langle\upsilon\rangle\, t'$ and $s = \lambda\langle\upsilon\rangle\, s'$. We have $t' \succ_{\mathsf{\lambda lpo}} s'$. By the induction hypothesis, $t'\theta \succ_{\mathsf{\lambda lpo}} s'\theta$. By rule 3c, we get $t\theta = \lambda\langle\upsilon\theta\rangle\, t'\theta \succ_{\mathsf{m\lambda lpo}} \lambda\langle\upsilon\theta\rangle\, s'\theta = s\theta$, as desired.

RULE 3D OF $\succ_{\mathsf{\lambda lpo}}$: This case compares a λ-abstraction $t = \lambda\langle\upsilon\rangle\, t'$ and an applied symbol $\mathsf{f}(\bar{u})\,\bar{s}$ below the watershed. The proof is similar to as in cases 2b, 2c of $\succ_{\mathsf{m\lambda lpo}}$ above.

RULE 1 OF $\succsim_{\mathsf{\lambda lpo}}$: This case compares two preterms $y\,\bar{t}$ and $y\,\bar{s}$ headed by the same variable and of the same type τ. Let k be the number of curried arguments expected by values of type $\tau\theta$. This means that we have

$$t\theta = \underbrace{\lambda \ldots \lambda}_{k \text{ times}}\ \underbrace{(y\,\bar{t})\theta{\uparrow}^k\ (k-1){\uparrow}_\eta \ldots 0{\uparrow}_\eta}_{(y\,\bar{t})\theta?} \qquad s\theta = \underbrace{\lambda \ldots \lambda}_{k \text{ times}}\ \underbrace{(y\,\bar{s})\theta{\uparrow}^k\ (k-1){\uparrow}_\eta \ldots 0{\uparrow}_\eta}_{(y\,\bar{s})\theta?}$$

To show $t\theta \succsim_{\mathsf{m\lambda lpo}} s\theta$, we apply rule 4c k times. The rule is applicable if $(y\bar{t})\theta? \succsim_{\mathsf{m\lambda lpo}} (y\,\bar{s})\theta?$. It is easy to see that this last inequality can be established using rule 1 given that $y\bar{t} \succsim_{\mathsf{m\lambda lpo}} y\bar{s}$, using the induction hypothesis to compare pairs t_i, s_i and using Lemma 4.17 for the pairs $j{\uparrow}_\eta, j{\uparrow}_\eta$ of (identical) De Bruijn indices introduced by η-expansion.

CASES 2A, 2B, 2C, 2E, 3A, 3B, 3D, 4A, 4B, 4D OF $\succsim_{\mathsf{\lambda lpo}}$: These cases are similar to cases 1a, 1b, 1c, 1e, 2a, 2b, 2d, 3a, 3b, 3d of $\succ_{\mathsf{\lambda lpo}}$. These cases correspond to strict inequalities. We first establish $t\theta \succsim_{\mathsf{m\lambda lpo}} t\theta?$ by applying rule 4a repeatedly. Then we show $t\theta? \succsim_{\mathsf{m\lambda lpo}} s\theta$ in the same way as in the corresponding case of $\succ_{\mathsf{\lambda lpo}}$.

RULES 2D, 3C OF $\succsim_{\mathsf{\lambda lpo}}$: These cases may correspond to nonstrict comparisons—for example, if the argument tuples are equal. We apply rule 4c repeatedly to eliminate any λs on both sides. If there are any λs remaining on the left, proceed as in the previous case (2a, 2b, etc.). Otherwise, the rest of the proof is similar to case 1d or 2c of $\succ_{\mathsf{\lambda lpo}}$.

RULES 4C OF $\succsim_{\mathsf{\lambda lpo}}$: Analogous to case 3c of $\succ_{\mathsf{\lambda lpo}}$. □

Theorem 5.17. *The relation $\succ_{\mathsf{\lambda lpo}}$ coincides with $\succ_{\mathsf{m\lambda lpo}}$ on monomorphic preterms. The relation $\succsim_{\mathsf{\lambda lpo}}$ coincides with $\succsim_{\mathsf{m\lambda lpo}}$ on monomorphic preterms.*

Proof. One direction of the equivalences follows by Theorem 5.16. It remains to show that $t \succ_{\mathsf{m\lambda lpo}} s$ implies $t \succ_{\mathsf{\lambda lpo}} s$ and that $t \succsim_{\mathsf{m\lambda lpo}} s$ implies $t \succsim_{\mathsf{\lambda lpo}} s$. The proof is by induction on the definition of $\succ_{\mathsf{m\lambda lpo}}$ and $\succsim_{\mathsf{m\lambda lpo}}$. It is easy to see that every case in the definition of $\succ_{\mathsf{m\lambda lpo}}$ corresponds a case in the definition of $\succ_{\mathsf{\lambda lpo}}$ and every case in the definition of $\succsim_{\mathsf{m\lambda lpo}}$ corresponds a case in the definition of $\succsim_{\mathsf{\lambda lpo}}$. □

Lemma 5.18. *If $t \succ_{\mathsf{\lambda lpo}} s$, then $t \succsim_{\mathsf{\lambda lpo}} s$.*

Proof. Analogous to Lemma 4.18. □

Theorem 5.19. *If $t \succsim_{\mathsf{\lambda lpo}} u$ and $u \succsim_{\mathsf{\lambda lpo}} s$, then $t \succsim_{\mathsf{\lambda lpo}} s$. If moreover $t \succ_{\mathsf{\lambda lpo}} u$ or $u \succ_{\mathsf{\lambda lpo}} s$, then $t \succ_{\mathsf{\lambda lpo}} s$.*

Proof. Analogous to Theorem 4.22, using the fact that $\trianglerighteq$ is transitive. □

Theorem 5.20. *Let $t \succ_{\mathsf{\lambda lpo}} s$. Let θ be a substitution such that all variables in $t\theta$ and $s\theta$ are nonfunctional term variables. Let $s\theta$ contain a nonfunctional variable x outside of parameters. Then $t\theta$ must also contain x outside of parameters.*

Proof. By Theorems 4.23 and 5.16. □

6. Examples

Let us see how λKBO and λLPO work on some realistic examples. All the examples below are beyond the reach of the derived higher-order KBO and LPO presented by Bentkamp et al. [3] and implemented in Zipperposition.

Example 6.1. Consider the following clause: $\mathsf{p}\,(\lambda\,\mathsf{f}\,(y\,0)) \vee \mathsf{p}\,(\lambda\,y\,0)$. We would like to orient the two literals. The orientation $\mathsf{p}\,(\lambda\,\mathsf{f}\,(y\,0)) \succ \mathsf{p}\,(\lambda\,y\,0)$ appears more promising. With λKBO, assuming a weight of 1 for f, p, λ, and 0 and argument coefficients of 1, via a mechanical application of Definition 5.4 we get the polynomial inequality $1+1+1+1+\mathbf{w}_y+\mathbf{k}_{y,1}(1-1) > 1+1+1+\mathbf{w}_y+\mathbf{k}_{y,1}(1-1)$. In other words, $1 > 0$. With λLPO, the desired orientation is easy to derive since $y\,0$ is a subterm of $\mathsf{f}\,(y\,0)$.

Example 6.2. The following equation defines the transitivity of a relation r, encoded as a binary predicate: $\mathsf{trans}\,(\lambda\,\lambda\,r\,1\,0) \approx \boldsymbol{\forall}\,(\lambda\,\boldsymbol{\forall}\,(\lambda\,\boldsymbol{\forall}\,(\lambda\,r\,2\,1 \mathbin{\boldsymbol{\wedge}} r\,1\,0 \mathbin{\boldsymbol{\rightarrow}} r\,2\,0)))$. We start with λKBO. Assume a weight of 1 for trans, $\boldsymbol{\forall}$, $\boldsymbol{\wedge}$, $\boldsymbol{\rightarrow}$, λ, 0, … and argument coefficients of 1. After simplification, the polynomial inequality for a right-to-left orientation becomes $\mathbf{w}_r + 4 < 3\mathbf{w}_r + 14$, which clearly holds. Is there a way to orient the equation from left to right instead? There is if we make trans heavier and set a higher weight coefficient on its argument. Take $\mathcal{w}(\mathsf{trans}) = 5$ and $\mathcal{k}(\mathsf{trans}, 1) = 3$. Then we get $3\mathbf{w}_r + 15 > 3\mathbf{w}_r + 14$. In contrast, λLPO cannot orient the equation from left to right; among the proof obligations that emerge are $r\,1\,0 \succsim^?_{\mathsf{\lambda lpo}} r\,2\,1$ and $r\,1\,0 \succsim^?_{\mathsf{\lambda lpo}} r\,2\,0$, and these cannot be discharged.

Example 6.3. In the clause $y\,(\lambda\,\mathsf{a}\,0) \vee \neg\,y\,(\lambda\,\mathsf{f}\,(\mathsf{sk}(y)\,0)) \vee \neg\,y\,(\lambda\,0)$, we would like to make $y\,(\lambda\,\mathsf{a}\,0)$ the maximal literal. The apparent difficulty is the presence of the variable y deep inside the second literal. Fortunately, since it occurs in a parameter, it has no impact on the λKBO weight. We are then free to make the symbol a as heavy as we want to ensure that $\lambda\,\mathsf{a}\,0$ is heavier than $\lambda\,\mathsf{f}\,(\mathsf{sk}(y)\,0)$ and $\lambda\,0$, both of which have constant weights. A similar approach can be taken with λLPO, using the precedence instead of weights.

Example 6.4. In functional programming, the map function on lists is defined recursively by

$$\mathsf{map}\,(\lambda\,f\,0)\,\mathsf{nil} \approx \mathsf{nil} \qquad \mathsf{map}\,(\lambda\,f\,0)\,(\mathsf{cons}\,x\,xs) \approx \mathsf{cons}\,(f\,x)\,(\mathsf{map}\,(\lambda\,f\,0)\,xs)$$

The first equation is easy to orient from left to right. Not so for the second equation. With λKBO, to compensate for the two occurrences of f on the right-hand side, we would need to set a coefficient of at least 2 on map's first argument; this would make the left-hand side heavier but would also make the right-hand side even heavier. In general, KBO is rather ineffective at orienting recursive equations from left to right. With λLPO, the issue is the undischargeable proof obligation $f\,0 \succsim^?_{\mathsf{\lambda lpo}} f\,x$.

With both orders, a right-to-left orientation is also problematic, because f might ignore its argument, resulting in an x on the left-hand side with no matching x on the right-hand side. On the positive side, superposition provers rarely need to orient recursive equations in their full generality. Instead, the calculus considers instances of the equations where higher-order variables are replaced by concrete functions. These equation instances are often orientable from right to left.

7. Naive Algorithms

The definitions given in Sect. 5 provide a sound theoretical basis for an implementation, but they should not be followed naively. Our algorithms below perform the comparisons $t \succ s$, $t \succsim s$, $t = s$, $t \precsim s$, and $t \prec s$ simultaneously, reusing subcomputations. Typically, given terms s, t, a superposition prover might need to check both $t \succsim s$ and $t \precsim s$.

As our programming language, we use a functional programming notation inspired by Standard ML, OCaml, and Haskell. First, we need a type to represent the result of a comparison:

$\quad$ **datatype** $\mathsf{cmp} = \mathsf{G} \mid \mathsf{GE} \mid \mathsf{E} \mid \mathsf{LE} \mid \mathsf{L} \mid \mathsf{U}$

The six values represent $\succ$, $\succsim$, $=$, $\precsim$, $\prec$, and "unknown" or "incomparable," respectively.

7.1. $\boldsymbol{\lambda}$KBO.

Our first algorithm will perform λKBO comparisons in both directions simultaneously. The following auxiliary functions are used to combine an imprecise comparison result GE or LE with another result, using U on mismatch:

```
function mergeWithGE cmp :=
  match cmp with
    L | LE ⇒ U
  | E ⇒ GE
  | _ ⇒ cmp
  end
function mergeWithLE cmp :=
  match cmp with
    G | GE ⇒ U
  | E ⇒ LE
  | _ ⇒ cmp
  end
```

The lexicographic extension of a comparison operator working with our comparison type is defined as follows:

```
function lexExt op b̄ ā :=
  match b̄, ā with
    [], [] ⇒ E
  | b :: b̄′, a :: ā′ ⇒
    match op b a with
      G ⇒ G
    | GE ⇒ mergeWithGE (lexExt op b̄′ ā′)
    | E ⇒ lexExt op b̄′ ā′
    | LE ⇒ mergeWithLE (lexExt op b̄′ ā′)
```

```
      | L ⇒ L
      | U ⇒ U
    end
  end
```

We need to support only the case in which both lists have the same length.

Next to the lexicographic extension, we also define a form of componentwise extension of a comparison operator:

```
function smooth cmp :=
  match cmp with
    G ⇒ GE
  | L ⇒ LE
  | _ ⇒ cmp
  end

function cwExt op :=
  lexExt (fun b a ⇒ smooth (op b a))
```

We need to support only the case in which both lists have the same length.

Next, we need a function that checks whether $\trianglerighteq$ or its inverse $\trianglelefteq$ holds and that adjusts the comparison result accordingly:

```
function considerPoly t s cmp :=
  match cmp with
    G | GE ⇒ if t ⊵ s then cmp else U
    L | LE ⇒ if t ⊴ s then cmp else U
  | _ ⇒ cmp
  end
```

The function for checking inequalities is very simple:

```
function surelyNonneg w :=
  all coefficients in the standard form of w are ≥ 0
```

It returns true if all the polynomials in the list are certainly nonnegative for any values of the indeterminates and false if this is not known to be the case, either because there exists a counterexample or because the approach is too imprecise to tell.

Polynomials in standard form have at most one *constant monomial*: a monomial consisting of only a coefficient with no indeterminates. If absent, it is taken to be 0. The weight comparison can be refined by considering the sign of the constant monomial in the difference $\mathcal{W}(t) - \mathcal{W}(s)$. If the sign is positive, $\mathcal{W}(t) \geq \mathcal{W}(s)$ actually means $\mathcal{W}(t) > \mathcal{W}(s)$. If the sign is negative, $\mathcal{W}(t) \leq \mathcal{W}(s)$ actually means $W(t) < \mathcal{W}(s)$.

```
function analyzeWeightDiff w :=
  match surelyNonneg w, surelyNonneg (−w) with
    false, false ⇒ U
  | true, false ⇒ if the constant monomial of w is > 0 then G else GE
  | false, true ⇒ if the constant monomial of w is < 0 then L else LE
  | true, true ⇒ E
  end
```

For preterms with possibly equal weights, a lexicographic comparison implemented by the compareShapes function below breaks the tie. We assume the existence of a function compareSyms g f based on $>$ that returns G, E, or L and of a function compareTypes υ τ

based on $\succ_{\mathsf{ty}}$ that returns G, E, L, or U. The $\mathsf{compareShapes}$ function is mutually recursive with the main comparison function, $\mathsf{compareTerms}$.

$$
\begin{array}{l}
\textbf{function } \mathsf{compareShapes}\ t\ s\ := \\
\quad \textbf{match } t,\ s \textbf{ with} \\
\qquad y\,\bar{t},\ y\,\bar{s} \Rightarrow \textbf{if } \bar{t} \text{ are steady } \textbf{then } \mathsf{cwExt}\ \mathsf{compareTerms}\ \bar{t}\ \bar{s} \textbf{ else } \mathsf{U} \\
\quad\ \mid y\ \_,\ \_ \mid \_,\ x\ \_ \Rightarrow \mathsf{U} \\
\quad\ \mid \lambda\langle\upsilon\rangle\, t',\ \lambda\langle\tau\rangle\, s' \Rightarrow \\
\qquad \textbf{match } \mathsf{compareTypes}\ \upsilon\ \tau \textbf{ with} \\
\qquad\quad \mathsf{E} \Rightarrow \mathsf{compareShapes}\ t'\ s' \\
\qquad\ \mid \mathit{cmp} \Rightarrow \mathit{cmp} \\
\qquad \textbf{end} \\
\quad\ \mid \lambda\ \_,\ \_ \Rightarrow \mathsf{considerPoly}\ t\ s\ \mathsf{G} \\
\quad\ \mid n\ \_,\ \lambda\ \_ \Rightarrow \mathsf{considerPoly}\ t\ s\ \mathsf{L} \\
\quad\ \mid n\,\bar{t},\ m\,\bar{s} \Rightarrow \\
\qquad \textbf{if } n > m \textbf{ then } \mathsf{considerPoly}\ t\ s\ \mathsf{G} \\
\qquad \textbf{else if } n < m \textbf{ then } \mathsf{considerPoly}\ t\ s\ \mathsf{L} \\
\qquad \textbf{else } \mathsf{lexExt}\ \mathsf{compareTerms}\ \bar{t}\ \bar{s} \\
\quad\ \mid n\ \_,\ \mathsf{f}(\_)\ \_ \Rightarrow \mathsf{considerPoly}\ t\ s\ \mathsf{G} \\
\quad\ \mid \mathsf{g}\langle\bar{\upsilon}\rangle(\bar{v})\,\bar{t},\ \mathsf{f}\langle\bar{\tau}\rangle(\bar{u})\,\bar{s} \Rightarrow \\
\qquad \textbf{match } \mathsf{compareSyms}\ \mathsf{g}\ \mathsf{f} \textbf{ with} \\
\qquad\quad \mathsf{E} \Rightarrow \\
\qquad\quad \textbf{match } \mathsf{lexExt}\ \mathsf{compareTypes}\ \bar{\upsilon}\ \bar{\tau} \textbf{ with} \\
\qquad\qquad \mathsf{E} \Rightarrow \mathsf{lexExt}\ \mathsf{compareTerms}\ (\bar{v}\cdot\bar{t})\ (\bar{u}\cdot\bar{s}) \\
\qquad\quad\ \mid \mathit{cmp} \Rightarrow \mathsf{considerPoly}\ t\ s\ \mathit{cmp} \\
\qquad\quad \textbf{end} \\
\qquad\ \mid \mathit{cmp} \Rightarrow \mathsf{considerPoly}\ t\ s\ \mathit{cmp} \\
\qquad \textbf{end} \\
\quad\ \mid \mathsf{g}(\_)\ \_,\ \_ \Rightarrow \mathsf{considerPoly}\ t\ s\ \mathsf{L} \\
\quad \textbf{end}
\end{array}
$$

In the above, the operator $\cdot$ denotes list concatenation.

The main function implementing λKBO invokes $\mathsf{analyzeWeightDiff}$, falling back on $\mathsf{compareShapes}$ to break ties:

$$
\begin{array}{l}
\textbf{function } \mathsf{compareTerms}\ t\ s\ := \\
\quad \textbf{match } \mathsf{analyzeWeightDiff}\ (\mathcal{W}(t) - \mathcal{W}(s)) \textbf{ with} \\
\qquad \mathsf{G} \Rightarrow \mathsf{G} \\
\quad\ \mid \mathsf{GE} \Rightarrow \mathsf{mergeWithGE}\ (\mathsf{compareShapes}\ t\ s) \\
\quad\ \mid \mathsf{E} \Rightarrow \mathsf{compareShapes}\ t\ s \\
\quad\ \mid \mathsf{LE} \Rightarrow \mathsf{mergeWithLE}\ (\mathsf{compareShapes}\ t\ s) \\
\quad\ \mid \mathsf{L} \Rightarrow \mathsf{L} \\
\quad\ \mid \mathsf{U} \Rightarrow \mathsf{U} \\
\quad \textbf{end}
\end{array}
$$

7.2. **$\boldsymbol{\lambda}$LPO.** The following algorithm performs λLPO comparisons in both directions simultaneously. The main comparison function, $\mathsf{compareTerms}$, is accompanied by four mutually recursive auxiliary functions.

```
function considerPolyBelowWS g t s cmp :=
  if g > ws then cmp else considerPoly t s cmp

  function checkSubs t̄ s :=
  ∃i. compareTerms t_i s ∈ {G, GE, E}

  function checkArgs t s̄ :=
  ∀i. compareTerms t s_i = G

  function compareArgs t v̄ t̄ s ū s̄ :=
  match lexExt compareTerms (v̄ · t̄) (ū · s̄) with
    G ⇒ if checkArgs t s̄ then G else U
  | GE ⇒ if checkArgs t s̄ then GE else U
  | E ⇒ E
  | LE ⇒ if checkArgs s t̄ then LE else U
  | L ⇒ if checkArgs s t̄ then L else U
  | U ⇒ U
  end

  function compareTerms t s :=
  match t with
    y t̄ ⇒
    match s with
      x s̄ ⇒ if y = x ∧ t̄ are steady then cwExt compareTerms t̄ s̄ else U
     | f(_) s̄ | m s̄ ⇒ if checkSubs s̄ t then L else U
     | λ s′ ⇒ if checkSubs [s′] t then L else U
    end
  | g(v̄) t̄ ⇒
    if checkSubs t̄ s then
      G
    else match s with
      x _ ⇒ U
     | f(ū) s̄ ⇒
      if checkSubs s̄ t then
        L
      else match compareSyms g f with
        G ⇒ if checkArgs t s̄ then considerPolyBelowWS g G else U
       | E ⇒
        match lexExt compareTypes ῡ τ̄ with
          G ⇒ if checkArgs t s̄ then considerPolyBelowWS g G else U
         | E ⇒ compareArgs t v̄ t̄ s ū s̄
         | L ⇒ if checkArgs s t̄ then considerPolyBelowWS g L else U
         | U ⇒ U
        end
       | L ⇒ if checkArgs s t̄ then considerPolyBelowWS g L else U
       | U ⇒ U
      end
     | m s̄ ⇒
      if checkSubs s̄ t then L
```

```
    else if g > ws ∧ checkArgs t s̄ then G
    else if g ≤ ws ∧ checkArgs s t̄ then considerPoly t s L
    else U
  | λ s′ ⇒
    if checkSubs [s′] t then L
    else if g > ws ∧ checkArgs t [s′] then G
    else if g ≤ ws ∧ checkArgs s t̄ then considerPoly t s L
    else U
  end
| n t̄ ⇒
  if checkSubs t̄ s then
    G
  else match s with
    x _ ⇒ U
  | f(_) s̄ ⇒
    if checkSubs s̄ t then L
    else if f > ws ∧ checkArgs s t̄ then L
    else if f ≤ ws ∧ checkArgs t s̄ then considerPoly t s G
    else U
  | m s̄ ⇒
    if checkSubs s̄ t then
      L
    else if n > m then
      if checkArgs t s̄ then considerPoly t s G else U
    else if n = m then
      compareArgs t [] t̄ s [] s̄
    else
      if checkArgs s t̄ then considerPoly t s L else U
  | λ s′ ⇒
    if checkSubs [s′] t then L
    else if checkArgs t [s′] then G
    else U
  end
| λ⟨υ⟩ t′ ⇒
  if checkSubs [t′] s then
    G
  else match s with
    x _ ⇒ U
  | f(_) s̄ ⇒
    if checkSubs s̄ t then L
    else if f > ws ∧ checkArgs s [t′] then L
    else if f ≤ ws ∧ checkArgs t s̄ then considerPoly t s G
    else U
  | m s̄ ⇒
    if checkSubs s̄ t then L
    else if checkArgs s [t′] then L
```

```
        else U
      | λ⟨τ⟩ s′ ⇒
        if checkSubs [s′] t then
          L
        else match compareTypes υ τ with
          G ⇒ if checkArgs t [s′] then G else U
         | E ⇒ compareTerms t′ s′
         | L ⇒ if checkArgs s [t′] then L else U
         | U ⇒ U
        end
      end
    end
```

8. Optimized Algorithms

Another improvement, embodied by a separate pair of algorithms, consists of following Löchner's refinement approach [8, 9]. For the standard KBO and LPO, his comparison algorithms are respectively linear and quadratic in the size of the input terms. The use of polynomials instead of integers in the λKBO makes the computation slightly more expensive, but we can nonetheless benefit from tupling.

8.1. **λKBO.**

The naive bidirectional algorithm for λKBO is wasteful because it recursively recomputes preterm weights. If $t = \mathsf{f}(\bar{t})$, the subterm t_i's weight is computed first in the main function by the call to analyzeWeightDiff and then possibly again in compareShapes, when compareTerms is called to break ties. Although a factor of 2 might not sound particularly expensive, the factor is higher for the subterms' subterms, their subsubterms, and so on. Thus, the native algorithm is quadratic in the size of the input preterms [8].

Our solution, inspired by Löchner [8], consists of interleaving the two passes: computing the weights and comparing the shapes. The information for the passes is stored in a tuple. In this way, the subterms' weights can be shared between the passes. At the end of the combined pass, we can look at the tuple and determine what result to return.

First, we need to extend the lexicographic and componentwise extension functions to thread through additional information—in our case, weights—returned by the operator *op* as the first component of a pair, the second component being the comparison result.

```
function lexExtData op b̄ ā :=
  match b̄, ā with
    [], [] ⇒ ([], E)
  | b :: b̄′, a :: ā′ ⇒
    match op b a with
      (w, G) ⇒ ([w], G)
     | (w, GE) ⇒
      let (w̄, cmp) := lexExtData op b̄′ ā′ in
      (w :: w̄, mergeWithGE cmp)
     | (w, E) ⇒
      let (w̄, cmp) := lexExtData op b̄′ ā′ in
      (w :: w̄, cmp)
```

```
    | (w, LE) ⇒
      let (w̄, cmp) := lexExtData op b̄′ ā′ in
      (w :: w̄, mergeWithLE cmp)
    | (w, L) ⇒ ([w], L)
    | (w, U) ⇒ ([w], U)
  end
end

function cwExtData op :=
lexExtData (fun b a ⇒
  let (w, cmp) = op b a in
  (w, smooth cmp))
```

In the above, the operator :: ("cons") prepends an element to a list.

The auxiliary function considerWeight resembles the unoptimized compareTerms, but it uses its arguments w and cmp instead of recomputing them, where cmp is the result of a shape comparison.

```
function considerWeight w cmp :=
  (w, match analyzeWeightDiff w with
        G ⇒ G
      | GE ⇒ mergeWithGE cmp
      | E ⇒ cmp
      | LE ⇒ mergeWithLE cmp
      | L ⇒ L
      | U ⇒ U
      end)
```

The core of the code consists of two mutually recursive functions: processArgs and processTerms. They compute weights and compare shapes, returning pairs of the form $(w,\ cmp)$, where cmp takes both the preterms' weights and their shapes into account. The code for processTerms follows the structure of the unoptimized compareShape but is instrumented to also compute weights. It calls processArgs to compare argument lists. In processArgs, the weights computed as part of the lexicographic comparison are reused and extended with any missing weights if the comparison ended before the end of the lists (i.e., if $m < n$).

function processArgs $\bar{t}_n\ \bar{s}_n$:=
  let $(\bar{w}_m,\ cmp)$:= lexExtData processTerms $\bar{t}_n\ \bar{s}_n$ **in**
  considerWeight $(\sum_{i=1}^{m} w_i + \sum_{i=m+1}^{n}(\mathcal{W}(t_i) - \mathcal{W}(s_i)))\ cmp$

function processTerms $t\ s$:=
match $t,\ s$ **with**
  $y\,\bar{t},\ x\,\bar{s} \Rightarrow$
  if $y = x$ **then**
    if $\bar{t}$ are steady **then**
      let $(\bar{w},\ cmp)$:= cwExtData processTerms $\bar{t}\ \bar{s}$ **in**
      considerWeight $\left(\sum_{i=1}^{|\bar{t}|} \mathbf{k}_{y,i}\, w_i\right)\ cmp$
    else
      considerWeight $(\mathcal{W}(t) - \mathcal{W}(s))$ U
  else

$\quad\quad \mathsf{considerWeight}\ (\mathcal{W}(t) - \mathcal{W}(s))\ \mathsf{U}$
$\mid y\ \_,\ \_ \mid \_,\ x\ \_ \Rightarrow \mathsf{considerWeight}\ (\mathcal{W}(t) - \mathcal{W}(s))\ \mathsf{U}$
$\mid \lambda\langle\upsilon\rangle\, t',\ \lambda\langle\tau\rangle\, s' \Rightarrow$
$\quad$ **match** $\mathsf{compareTypes}\ \upsilon\ \tau$ **with**
$\quad\quad \mathsf{E} \Rightarrow \mathsf{processTerms}\ t'\ s'$
$\quad \mid cmp \Rightarrow \mathsf{considerWeight}\ (\mathcal{W}(t') - \mathcal{W}(s'))\ cmp$
$\quad$ **end**
$\mid \lambda\ \_,\ \_ \Rightarrow \mathsf{considerWeight}\ (\mathcal{W}(t) - \mathcal{W}(s))\ (\mathsf{considerPoly}\ t\ s\ \mathsf{G})$
$\mid n\ \_,\ \lambda\ \_ \Rightarrow \mathsf{considerWeight}\ (\mathcal{W}(t) - \mathcal{W}(s))\ (\mathsf{considerPoly}\ t\ s\ \mathsf{L})$
$\mid n\,\bar{t},\ m\,\bar{s} \Rightarrow$
$\quad$ **if** $n > m$ **then** $\mathsf{considerWeight}\ (\mathcal{W}(t) - \mathcal{W}(s))\ (\mathsf{considerPoly}\ t\ s\ \mathsf{G})$
$\quad$ **else if** $n < m$ **then** $\mathsf{considerWeight}\ (\mathcal{W}(t) - \mathcal{W}(s))\ (\mathsf{considerPoly}\ t\ s\ \mathsf{L})$
$\quad$ **else** $\mathsf{processArgs}\ \bar{t}\ \bar{s}$
$\mid n\ \_,\ \mathsf{f}(\_)\ \_ \Rightarrow \mathsf{considerWeight}\ (\mathcal{W}(t) - \mathcal{W}(s))\ (\mathsf{considerPoly}\ t\ s\ \mathsf{G})$
$\mid \mathsf{g}\langle\bar{\upsilon}\rangle(\bar{v})\,\bar{t},\ \mathsf{f}\langle\bar{\tau}\rangle(\bar{u})\,\bar{s} \Rightarrow$
$\quad$ **match** $\mathsf{compareSyms}\ \mathsf{g}\ \mathsf{f}$ **with**
$\quad\quad \mathsf{E} \Rightarrow$
$\quad\quad$ **match** $\mathsf{lexExt}\ \mathsf{compareTypes}\ \bar{\upsilon}\ \bar{\tau}$ **with**
$\quad\quad\quad \mathsf{E} \Rightarrow \mathsf{processArgs}\ (\bar{v}\cdot\bar{t})\ (\bar{u}\cdot\bar{s})$
$\quad\quad \mid cmp \Rightarrow \mathsf{considerWeight}\ (\mathcal{W}(t) - \mathcal{W}(s))\ (\mathsf{considerPoly}\ t\ s\ cmp)$
$\quad\quad$ **end**
$\quad \mid cmp \Rightarrow \mathsf{considerWeight}\ (\mathcal{W}(t) - \mathcal{W}(s))\ (\mathsf{considerPoly}\ t\ s\ cmp)$
$\quad$ **end**
$\mid \mathsf{g}(\_)\ \_,\ \_ \Rightarrow \mathsf{considerWeight}\ (\mathcal{W}(t) - \mathcal{W}(s))\ (\mathsf{considerPoly}\ t\ s\ \mathsf{L})$
end

When calling processTerms to compare two preterms, we would normally ignore the w component of the result and only consider cmp, which should be equal to what the untupled compareTerms would return.

One last point to discuss is the representation of polynomials. In the standard KBO, multisets of variables must be compared. These can be seen as polynomials of degree 1. Löchner's approach for the KBO variable check is to use an array indexed by a finite variable set X. Clearly, this technique does not scale to polynomials of arbitrarily high degrees. Instead of arrays, we can use maps or hash tables indexed by sorted lists of indeterminates. With a reasonable map implementation, this would replace an $\mathrm{O}(n)$ complexity with $\mathrm{O}(n \log n)$, where $n = |s| + |t|$, the size of the input preterms.

One of Löchner's ideas that also applies in our setting is to maintain two counters indicating how many monomials are nonnegative or nonpositive in the current polynomial expressed in standard form. These counters must be updated whenever the map or hash table is modified. The two calls to surelyNonneg in analyzeWeightDiff can then be replaced by two conditions that each check whether a counter is 0.

8.2. **λLPO.** The naive bidirectional algorithm has exponential complexity because of the overlapping computations of checkArgs and checkSubs. Our solution, again inspired by Löchner [9], consists of postponing the checks and avoiding redundant comparisons. Specifically, our algorithm below draws inspiration from Löchner's clpo_6.

We start with a simple auxiliary function:

```
function flip cmp :=
  match cmp with
    G ⇒ L
  | GE ⇒ LE
  | E ⇒ E
  | LE ⇒ GE
  | L ⇒ G
  | U ⇒ U
  end
```

The following six functions are mutually recursive. The main function is called compareTerms, as in Section 7.

```
function checkSubs t̄ s :=
  ∃i. compareTerms t_i s ∈ {G, GE, E}

function compareSubsBothWays t t̄ s s̄ :=
  if checkSubs t̄ s then G
  else if checkSubs s̄ t then L
  else U

function compareRest t s̄ :=
  match s̄ with
    [] ⇒ G
  | s :: s̄′ ⇒
    match compareTerms t s with
      G ⇒ compareRest t s̄′
    | E | LE | L ⇒ L
    | GE | U ⇒ if checkSubs s̄′ t then L else U
    end
  end

function compareRegularArgs t t̄ s s̄ :=
  match t̄, s̄ with
    [], [] ⇒ E
  | t_1 :: t̄′, s_1 :: s̄′ ⇒
    match compareTerms t_1 s_1 with
      G ⇒ compareRest t s̄′
    | GE ⇒ mergeWithGE (compareRegularArgs t t̄′ s s̄′)
    | E ⇒ compareRegularArgs t t̄′ s s̄′
    | LE ⇒ mergeWithLE (compareRegularArgs t t̄′ s s̄′)
    | L ⇒ flip (compareRest s t̄′)
    | U ⇒ compareSubsBothWays t t̄′ s s̄′
    end
  end

function compareArgs t v̄ t̄ s ū s̄ :=
  match v̄, ū with
    [], [] ⇒ compareRegularArgs t t̄ s s̄
  | v_1 :: v̄′, u_1 :: ū′ ⇒
    match compareTerms v_1 u_1 with
```

G $\Rightarrow$ compareRest $t\ \bar{s}$
| GE $\Rightarrow$ mergeWithGE (compareArgs $t\ \bar{v}'\ \bar{t}\ s\ \bar{u}'\ \bar{s}$)
| E $\Rightarrow$ compareArgs $t\ \bar{v}'\ \bar{t}\ s\ \bar{u}'\ \bar{s}$
| LE $\Rightarrow$ mergeWithLE (compareArgs $t\ \bar{v}'\ \bar{t}\ s\ \bar{u}'\ \bar{s}$)
| L $\Rightarrow$ flip (compareRest $s\ \bar{t}$)
| U $\Rightarrow$ compareSubsBothWays $t\ \bar{t}\ s\ \bar{s}$
end
end

function compareTerms $t\ s$:=
match t, s **with**
$y\,\bar{t}$, $x\,\bar{s}$ $\Rightarrow$ **if** $y = x \wedge \bar{t}$ are steady **then** cwExt compareTerms $\bar{t}\ \bar{s}$ **else** U
| y _, f(_) $\bar{s}$ | y _, $m\,\bar{s}$ $\Rightarrow$ **if** checkSubs $\bar{s}\ t$ **then** L **else** U
| y _, $\lambda\, s'$ $\Rightarrow$ **if** checkSubs $[s']\ t$ **then** L **else** U
| g(_) $\bar{t}$, x _ $\Rightarrow$ **if** checkSubs $\bar{t}\ s$ **then** G **else** U
| g($\bar{v}$) $\bar{t}$, f($\bar{u}$) $\bar{s}$ $\Rightarrow$
match compareSyms g f **with**
G $\Rightarrow$ considerPolyBelowWS g $t\ s$ (compareRest $t\ \bar{s}$)
| E $\Rightarrow$
match lexExt compareTypes $\bar{\upsilon}\ \bar{\tau}$ **with**
G $\Rightarrow$ considerPolyBelowWS g $t\ s$ (compareRest $t\ \bar{s}$)
| E $\Rightarrow$ compareArgs $t\ \bar{v}\ \bar{t}\ s\ \bar{u}\ \bar{s}$
| L $\Rightarrow$ considerPolyBelowWS g $t\ s$ (flip (compareRest $s\ \bar{t}$))
| U $\Rightarrow$ compareSubsBothWays $t\ \bar{t}\ s\ \bar{s}$
end
| L $\Rightarrow$ considerPolyBelowWS g $t\ s$ (flip (compareRest $t\ \bar{s}$))
| U $\Rightarrow$ compareSubsBothWays $t\ \bar{t}\ s\ \bar{s}$
end
| g(_) $\bar{t}$, $m\,\bar{s}$ $\Rightarrow$
if f $>$ w **then** compareRest $t\ \bar{s}$
else considerPoly $t\ s$ (flip (compareRest $s\ \bar{t}$))
| g(_) $\bar{t}$, $\lambda\, s'$ $\Rightarrow$
if f $>$ w **then** compareRest $t\ [s']$
else considerPoly $t\ s$ (flip (compareRest $s\ \bar{t}$))
| $n\,\bar{t}$, x _ $\Rightarrow$ **if** checkSubs $\bar{t}\ s$ **then** G **else** U
| $n\,\bar{t}$, f(_) $\bar{s}$ $\Rightarrow$
if f $>$ w **then** flip (compareRest $s\ \bar{t}$)
else considerPoly $t\ s$ (compareRest $t\ \bar{s}$)
| $n\,\bar{t}$, $m\,\bar{s}$ $\Rightarrow$
if $n > m$ **then** considerPoly $t\ s$ (compareRest $t\ \bar{s}$)
else if $n = m$ **then** compareRegularArgs $t\ \bar{t}\ s\ \bar{s}$
else considerPoly $t\ s$ (flip (compareRest $s\ \bar{t}$))
| n _, $\lambda\, s'$ $\Rightarrow$ compareRest $t\ [s']$
| $\lambda\, t'$, x _ $\Rightarrow$ **if** checkSubs $[t']\ s$ **then** G **else** U
| $\lambda\, t'$, f(_) $\bar{s}$ $\Rightarrow$
if f $>$ w **then** flip (compareRest $s\ [t']$)
else considerPoly $t\ s$ (compareRest $t\ \bar{s}$)

$\mid \lambda\, t',\, m\ \_ \Rightarrow \mathsf{flip}\ (\mathsf{compareRest}\ s\ t')$
$\mid \lambda\langle\upsilon\rangle\, t',\ \lambda\langle\tau\rangle\, s' \Rightarrow$
  match $\mathsf{compareTypes}\ \upsilon\ \tau$ **with**
    $\mathsf{G} \Rightarrow \mathsf{compareRest}\ t\ [s']$
  $\mid \mathsf{E} \Rightarrow \mathsf{compareTerms}\ t'\ s'$
  $\mid \mathsf{L} \Rightarrow \mathsf{flip}\ (\mathsf{compareRest}\ s\ [t'])$
  $\mid \mathsf{U} \Rightarrow \mathsf{compareSubsBothWays}\ t\ [t']\ s\ [s']$
  end
end

9. Conclusion

We defined two new term orders, λKBO and λLPO, for use with λ-superposition. We expect these new order to improve Zipperposition's performance, measured as both proving time and success rate. Some of the ideas might also apply to λ-free superposition [2] and combinatory superposition [4]: Despite working on logics devoid of λ-abstractions, these proof calculi contain axiom (Ext) and could benefit from the implicit η-expansion that makes its positive literal maximal.

Acknowledgment. Petar Vukmirović discussed ideas with us and provided some of the examples in Sect. 6.

Bentkamp and Blanchette's research has received funding from the European Research Council (ERC, Matryoshka, 713999 and Nekoka, 101083038) Blanchette's research has received funding from the Netherlands Organization for Scientific Research (NWO) under the Vidi program (project No. 016.Vidi.189.037, Lean Forward). Hetzenberger's research has received funding from the European Research Council (ERC, ARTIST, 101002685).

Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

We have used artificial intelligence tools for textual editing.

References

[1] Becker, H., Blanchette, J.C., Waldmann, U., Wand, D.: A transfinite Knuth–Bendix order for lambda-free higher-order terms. In: de Moura, L. (ed.) CADE-26. LNCS, vol. 10395, pp. 432–453. Springer (2017)
[2] Bentkamp, A., Blanchette, J., Cruanes, S., Waldmann, U.: Superposition for lambda-free higher-order logic. Log. Meth. Comput. Sci. **17**(2), 1:1–1:38 (2021)
[3] Bentkamp, A., Blanchette, J., Tourret, S., Vukmirović, P.: Superposition for full higher-order logic. In: Platzer, A., Sutcliffe, G. (eds.) CADE-28. LNCS, vol. 12699, pp. 396–412. Springer (2021)
[4] Bhayat, A., Reger, G.: A combinator-based superposition calculus for higher-order logic. In: Peltier, N., Sofronie-Stokkermans, V. (eds.) IJCAR 2020, Part I. LNCS, vol. 12166, pp. 278–296. Springer (2020)
[5] de Bruijn, N.G.: Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church–Rosser theorem. Indag. Math **75**(5), 381–392 (1972)
[6] Charguéraud, A.: The locally nameless representation. J. Autom. Reason. **49**(3), 363–408 (2012)
[7] Knuth, D.E., Bendix, P.B.: Simple word problems in universal algebras. In: Leech, J. (ed.) Computational Problems in Abstract Algebra. pp. 263–297. Pergamon Press (1970)
[8] Löchner, B.: Things to know when implementing KBO. J. Autom. Reasoning **36**(4), 289–310 (2006)
[9] Löchner, B.: Things to know when implementing lpo. Int. J. Artif. Intell. Tools **15**(1), 53–80 (2006)

[10] Ludwig, M., Waldmann, U.: An extension of the Knuth-Bendix ordering with LPO-like properties. In: Dershowitz, N., Voronkov, A. (eds.) LPAR-14. LNCS, vol. 4790, pp. 348–362. Springer (2007)

[11] Sternagel, C., Thiemann, R.: Formalizing Knuth-Bendix orders and Knuth-Bendix completion. In: van Raamsdonk, F. (ed.) Rewriting Techniques and Applications (RTA 2013). LIPIcs, vol. 21, pp. 287–302. Schloss Dagstuhl—Leibniz-Zentrum für Informatik (2013)

[12] Sutcliffe, G.: The 10th IJCAR automated theorem proving system competition—CASC-J10. AI Commun. pp. 1–15 (2021). https://doi.org/10.3233/AIC-201566

[13] 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)