

Formalization of Knuth–Bendix Orders for Lambda-Free Higher-Order Terms

Heiko Becker, Jasmin Christian Blanchette, Uwe Waldmann, and Daniel Wand

March 17, 2025

Abstract

This Isabelle/HOL formalization defines Knuth–Bendix orders for higher-order terms without λ -abstraction and proves many useful properties about them. The main order fully coincides with the standard transfinite KBO with subterm coefficients on first-order terms. It appears promising as the basis of a higher-order superposition calculus.

Contents

| | | |
|----------|-----------------------------------------------------------------------------------------------------------------|-----------|
| 1 | Introduction | 1 |
| 2 | Utilities for Knuth–Bendix Orders for Lambda-Free Higher-Order Terms | 1 |
| 3 | The Applicative Knuth–Bendix Order for Lambda-Free Higher-Order Terms | 3 |
| 4 | The Graceful Standard Knuth–Bendix Order for Lambda-Free Higher-Order Terms | 4 |
| 4.1 | Setup | 4 |
| 4.2 | Weights | 4 |
| 4.3 | Inductive Definitions | 5 |
| 4.4 | Irreflexivity | 5 |
| 4.5 | Transitivity | 5 |
| 4.6 | Subterm Property | 6 |
| 4.7 | Compatibility with Functions | 6 |
| 4.8 | Compatibility with Arguments | 6 |
| 4.9 | Stability under Substitution | 6 |
| 4.10 | Totality on Ground Terms | 7 |
| 4.11 | Well-foundedness | 7 |
| 5 | The Graceful Basic Knuth–Bendix Order for Lambda-Free Higher-Order Terms | 7 |
| 6 | The Graceful Transfinite Knuth–Bendix Order with Subterm Coefficients for Lambda-Free Higher-Order Terms | 8 |
| 6.1 | Setup | 8 |
| 6.2 | Weights and Subterm Coefficients | 10 |
| 6.3 | Inductive Definitions | 14 |
| 6.4 | Irreflexivity | 14 |
| 6.5 | Transitivity | 14 |
| 6.6 | Subterm Property | 15 |
| 6.7 | Compatibility with Functions | 15 |
| 6.8 | Compatibility with Arguments | 15 |
| 6.9 | Stability under Substitution | 15 |
| 6.10 | Totality on Ground Terms | 16 |
| 6.11 | Well-foundedness | 16 |
| 7 | Properties of Lambda-Free KBO on the Lambda Encoding | 16 |
| 8 | Knuth–Bendix Orders for Lambda-Free Higher-Order Terms | 17 |

1 Introduction

This Isabelle/HOL formalization defines Knuth–Bendix orders for higher-order terms without λ -abstraction and proves many useful properties about them. The main order fully coincides with the standard transfinite KBO with subterm coefficients on first-order terms. It appears promising as the basis of a higher-order superposition calculus.

We refer to the following conference paper for details:

Heiko Becker, Jasmin Christian Blanchette, Uwe Waldmann, Daniel Wand:
A Transfinite Knuth–Bendix Order for Lambda-Free Higher-Order Terms.
CADE 2017: 432-453
https://www21.in.tum.de/~blanchet/lambda_free_kbo_conf.pdf

2 Utilities for Knuth–Bendix Orders for Lambda-Free Higher-Order Terms

```
theory Lambda_Free_KBO_Util
imports Lambda_Free_RPOs.Lambda_Free_Term Lambda_Free_RPOs.Extension_Orders Polynomials.Polynomials
begin

locale kbo_basic_basis = gt_sym (>s)
  for gt_sym :: 's ⇒ 's ⇒ bool (infix <>s> 50) +
  fixes
    wt_sym :: 's ⇒ nat and
    ε :: nat and
    ground_heads_var :: 'v ⇒ 's set and
    extf :: 's ⇒ (('s, 'v) tm ⇒ ('s, 'v) tm ⇒ bool) ⇒ ('s, 'v) tm list ⇒ ('s, 'v) tm list ⇒
      bool
  assumes
    ε_gt_0: ε > 0 and
    wt_sym_ge_ε: wt_sym f ≥ ε and
    ground_heads_var_nonempty: ground_heads_var x ≠ {} and
    extf_ext_irrefl_before_trans: ext_irrefl_before_trans (extf f) and
    extf_ext_compat_list_strong: ext_compat_list_strong (extf f) and
    extf_ext_hd_or_tl: ext_hd_or_tl (extf f)
begin

lemma wt_sym_gt_0: wt_sym f > 0
  ⟨proof⟩

end

locale kbo_std_basis = ground_heads (>s) arity_sym arity_var
  for
    gt_sym :: 's ⇒ 's ⇒ bool (infix <>s> 50) and
    arity_sym :: 's ⇒ enat and
    arity_var :: 'v ⇒ enat +
  fixes
    wt_sym :: 's ⇒ 'n::{ord,semiring_1} and
    ε :: nat and
    δ :: nat and
    extf :: 's ⇒ (('s, 'v) tm ⇒ ('s, 'v) tm ⇒ bool) ⇒ ('s, 'v) tm list ⇒ ('s, 'v) tm list ⇒
      bool
  assumes
    ε_gt_0: ε > 0 and
    δ_le_ε: δ ≤ ε and
    arity_hd_ne_infinity_if_δ_gt_0: δ > 0 ⇒ arity_hd ζ ≠ ∞ and
    wt_sym_ge: wt_sym f ≥ of_nat (ε - the_enat (of_nat δ * arity_sym f)) and
    unary_wt_sym_0_gt: arity_sym f = 1 ⇒ wt_sym f = 0 ⇒ f >s g ∨ g = f and
    unary_wt_sym_0_imp_δ_eq_ε: arity_sym f = 1 ⇒ wt_sym f = 0 ⇒ δ = ε and
    extf_ext_irrefl_before_trans: ext_irrefl_before_trans (extf f) and
```

```

    extf_ext_compat_list_strong: ext_compat_list_strong (extf f) and
    extf_ext_hd_or_tl: ext_hd_or_tl (extf f) and
    extf_ext_snoc_if_delta_eq_epsilon: delta = epsilon ==> ext_snoc (extf f)
begin

lemma arity_sym_ne_infinity_if_delta_gt_0: delta > 0 ==> arity_sym f != infinity
<proof>

lemma arity_var_ne_infinity_if_delta_gt_0: delta > 0 ==> arity_var x != infinity
<proof>

lemma arity_ne_infinity_if_delta_gt_0: delta > 0 ==> arity s != infinity
<proof>

lemma extf_ext_irrefl: ext_irrefl (extf f)
<proof>

lemma extf_ext: ext (extf f)
<proof>

lemma
  extf_ext_compat_cons: ext_compat_cons (extf f) and
  extf_ext_compat_snoc: ext_compat_snoc (extf f) and
  extf_ext_singleton: ext_singleton (extf f)
<proof>

lemma extf_ext_compat_list: ext_compat_list (extf f)
<proof>

lemma extf_ext_wf_bounded: ext_wf_bounded (extf f)
<proof>

lemmas extf_mono_strong = ext.mono_strong[OF extf_ext]
lemmas extf_mono = ext.mono[OF extf_ext, mono]
lemmas extf_map = ext.map[OF extf_ext]
lemmas extf_irrefl = ext_irrefl.irrefl[OF extf_ext_irrefl]
lemmas extf_trans_from_irrefl =
  ext_irrefl_before_trans.trans_from_irrefl[OF extf_ext_irrefl_before_trans]
lemmas extf_compat_cons = ext_compat_cons.compat_cons[OF extf_ext_compat_cons]
lemmas extf_compat_append_left = ext_compat_cons.compat_append_left[OF extf_ext_compat_cons]
lemmas extf_compat_append_right = ext_compat_snoc.compat_append_right[OF extf_ext_compat_snoc]
lemmas extf_compat_list = ext_compat_list.compat_list[OF extf_ext_compat_list]
lemmas extf_singleton = ext_singleton.singleton[OF extf_ext_singleton]
lemmas extf_wf_bounded = ext_wf_bounded.wf_bounded[OF extf_ext_wf_bounded]

lemmas extf_snoc_if_delta_eq_epsilon = ext_snoc.snoc[OF extf_ext_snoc_if_delta_eq_epsilon]

lemma extf_singleton_nil_if_delta_eq_epsilon: delta = epsilon ==> extf f gt [s] []
<proof>

end

sublocale kbo_basic_basis < kbo_std_basis __ lambda_. infinity lambda_. infinity __ 0
<proof>

end

```

3 The Applicative Knuth–Bendix Order for Lambda-Free Higher-Order Terms

```

theory Lambda_Free_KBO_App
imports Lambda_Free_KBO_Util

```

```

abbrevs >t = >t
and ≥t = ≥t
begin

```

This theory defines the applicative Knuth–Bendix order, a variant of KBO for λ -free higher-order terms. It corresponds to the order obtained by applying the standard first-order KBO on the applicative encoding of higher-order terms and assigning the lowest precedence to the application symbol.

```

locale kbo_app = gt_sym (>s)
  for gt_sym :: 's ⇒ 's ⇒ bool (infix <>s> 50) +
  fixes
    wt_sym :: 's ⇒ nat and
    ε :: nat and
    ext :: (('s, 'v) tm ⇒ ('s, 'v) tm ⇒ bool) ⇒ ('s, 'v) tm list ⇒ ('s, 'v) tm list ⇒ bool
  assumes
    ε_gt_0: ε > 0 and
    wt_sym_ge_ε: wt_sym f ≥ ε and
    ext_ext_irrefl_before_trans: ext_irrefl_before_trans ext and
    ext_ext_compat_list: ext_compat_list ext and
    ext_ext_hd_or_tl: ext_hd_or_tl ext
begin

```

```

lemma ext_mono[mono]: gt ≤ gt' ⇒ ext gt ≤ ext gt'
  <proof>

```

```

fun wt :: ('s, 'v) tm ⇒ nat where
  wt (Hd (Var x)) = ε
| wt (Hd (Sym f)) = wt_sym f
| wt (App s t) = wt s + wt t

```

```

inductive gt :: ('s, 'v) tm ⇒ ('s, 'v) tm ⇒ bool (infix <>t> 50) where
  gt_wt: vars_mset t ⊇# vars_mset s ⇒ wt t > wt s ⇒ t >t s
| gt_sym_sym: wt_sym g = wt_sym f ⇒ g >s f ⇒ Hd (Sym g) >t Hd (Sym f)
| gt_sym_app: vars s = {} ⇒ wt t = wt s ⇒ t = Hd (Sym g) ⇒ is_App s ⇒ t >t s
| gt_app_app: vars_mset t ⊇# vars_mset s ⇒ wt t = wt s ⇒ t = App t1 t2 ⇒ s = App s1 s2 ⇒
  ext (>t) [t1, t2] [s1, s2] ⇒ t >t s

```

```

abbreviation ge :: ('s, 'v) tm ⇒ ('s, 'v) tm ⇒ bool (infix <≥t> 50) where
  t ≥t s ≡ t >t s ∨ t = s

```

```

end

```

```

end

```

4 The Graceful Standard Knuth–Bendix Order for Lambda-Free Higher-Order Terms

```

theory Lambda_Free_KBO_Std
imports Lambda_Free_KBO_Util Nested_Multisets_Ordinals.Multiset_More
abbrevs >t = >t
and ≥t = ≥t
begin

```

This theory defines the standard version of the graceful Knuth–Bendix order for λ -free higher-order terms. Standard means that one symbol is allowed to have a weight of 0.

4.1 Setup

```

locale kbo_std = kbo_std_basis __ arity_sym arity_var wt_sym
  for
    arity_sym :: 's ⇒ enat and
    arity_var :: 'v ⇒ enat and
    wt_sym :: 's ⇒ nat

```

begin

4.2 Weights

primrec $wt :: ('s, 'v) tm \Rightarrow nat$ **where**

$wt (Hd \zeta) = (LEAST w. \exists f \in ground_heads \zeta. w = wt_sym f + the_enat (\delta * arity_sym f))$
 $| wt (App s t) = (wt s - \delta) + wt t$

lemma $wt_Hd_Sym: wt (Hd (Sym f)) = wt_sym f + the_enat (\delta * arity_sym f)$
 $\langle proof \rangle$

lemma $exists_wt_sym: \exists f \in ground_heads \zeta. wt (Hd \zeta) = wt_sym f + the_enat (\delta * arity_sym f)$
 $\langle proof \rangle$

lemma $wt_le_wt_sym: f \in ground_heads \zeta \Longrightarrow wt (Hd \zeta) \leq wt_sym f + the_enat (\delta * arity_sym f)$
 $\langle proof \rangle$

lemma $enat_the_enat_delta_times_arity_sym[simp]: enat (the_enat (\delta * arity_sym f)) = \delta * arity_sym f$
 $\langle proof \rangle$

lemma $wt_arg_le: wt (arg s) \leq wt s$
 $\langle proof \rangle$

lemma $wt_ge_epsilon: wt s \geq \epsilon$
 $\langle proof \rangle$

lemma $wt_ge_delta: wt s \geq \delta$
 $\langle proof \rangle$

lemma $wt_gt_delta_if_superunary: arity_hd (head s) > 1 \Longrightarrow wt s > \delta$
 $\langle proof \rangle$

lemma $wt_App_delta: wt (App s t) = wt t \Longrightarrow wt s = \delta$
 $\langle proof \rangle$

lemma $wt_App_ge_fun: wt (App s t) \geq wt s$
 $\langle proof \rangle$

lemma $wt_hd_le: wt (Hd (head s)) \leq wt s$
 $\langle proof \rangle$

lemma $wt_delta_imp_delta_eq_epsilon: wt s = \delta \Longrightarrow \delta = \epsilon$
 $\langle proof \rangle$

lemma $wt_ge_arity_head_if_delta_gt_0:$
assumes $\delta_gt_0: \delta > 0$
shows $wt s \geq arity_hd (head s)$
 $\langle proof \rangle$

lemma $wt_ge_num_args_if_delta_eq_0:$
assumes $\delta_eq_0: \delta = 0$
shows $wt s \geq num_args s$
 $\langle proof \rangle$

lemma $wt_ge_num_args: wary s \Longrightarrow wt s \geq num_args s$
 $\langle proof \rangle$

4.3 Inductive Definitions

inductive $gt :: ('s, 'v) tm \Rightarrow ('s, 'v) tm \Rightarrow bool$ (**infix** $\langle >_t \rangle$ 50) **where**

$gt_wt: vars_mset t \supseteq \# vars_mset s \Longrightarrow wt t > wt s \Longrightarrow t >_t s$
 $| gt_unary: wt t = wt s \Longrightarrow \neg head t \leq_{hd} head s \Longrightarrow num_args t = 1 \Longrightarrow$
 $(\exists f \in ground_heads (head t). arity_sym f = 1 \wedge wt_sym f = 0) \Longrightarrow arg t >_t s \vee arg t = s \Longrightarrow$
 $t >_t s$

| *gt_diff*: $\text{vars_mset } t \supseteq\# \text{vars_mset } s \implies \text{wt } t = \text{wt } s \implies \text{head } t >_{hd} \text{head } s \implies t >_t s$
| *gt_same*: $\text{vars_mset } t \supseteq\# \text{vars_mset } s \implies \text{wt } t = \text{wt } s \implies \text{head } t = \text{head } s \implies$
 $(\forall f \in \text{ground_heads } (\text{head } t). \text{extf } f (>_t) (\text{args } t) (\text{args } s)) \implies t >_t s$

abbreviation *ge* :: $(s, v) \text{ tm} \Rightarrow (s, v) \text{ tm} \Rightarrow \text{bool}$ (**infix** $\langle \geq_t \rangle$ 50) **where**
 $t \geq_t s \equiv t >_t s \vee t = s$

inductive *gt_wt* :: $(s, v) \text{ tm} \Rightarrow (s, v) \text{ tm} \Rightarrow \text{bool}$ **where**
gt_wtI: $\text{vars_mset } t \supseteq\# \text{vars_mset } s \implies \text{wt } t > \text{wt } s \implies \text{gt_wt } t s$

inductive *gt_diff* :: $(s, v) \text{ tm} \Rightarrow (s, v) \text{ tm} \Rightarrow \text{bool}$ **where**
gt_diffI: $\text{vars_mset } t \supseteq\# \text{vars_mset } s \implies \text{wt } t = \text{wt } s \implies \text{head } t >_{hd} \text{head } s \implies \text{gt_diff } t s$

inductive *gt_unary* :: $(s, v) \text{ tm} \Rightarrow (s, v) \text{ tm} \Rightarrow \text{bool}$ **where**
gt_unaryI: $\text{wt } t = \text{wt } s \implies \neg \text{head } t \leq_{hd} \text{head } s \implies \text{num_args } t = 1 \implies$
 $(\exists f \in \text{ground_heads } (\text{head } t). \text{arity_sym } f = 1 \wedge \text{wt_sym } f = 0) \implies \text{arg } t \geq_t s \implies \text{gt_unary } t s$

inductive *gt_same* :: $(s, v) \text{ tm} \Rightarrow (s, v) \text{ tm} \Rightarrow \text{bool}$ **where**
gt_sameI: $\text{vars_mset } t \supseteq\# \text{vars_mset } s \implies \text{wt } t = \text{wt } s \implies \text{head } t = \text{head } s \implies$
 $(\forall f \in \text{ground_heads } (\text{head } t). \text{extf } f (>_t) (\text{args } t) (\text{args } s)) \implies \text{gt_same } t s$

lemma *gt_iff_wt_unary_diff_same*: $t >_t s \iff \text{gt_wt } t s \vee \text{gt_unary } t s \vee \text{gt_diff } t s \vee \text{gt_same } t s$
 $\langle \text{proof} \rangle$

lemma *gt_imp_vars_mset*: $t >_t s \implies \text{vars_mset } t \supseteq\# \text{vars_mset } s$
 $\langle \text{proof} \rangle$

lemma *gt_imp_vars*: $t >_t s \implies \text{vars } t \supseteq \text{vars } s$
 $\langle \text{proof} \rangle$

4.4 Irreflexivity

theorem *gt_irrefl*: $\text{wary } s \implies \neg s >_t s$
 $\langle \text{proof} \rangle$

4.5 Transitivity

lemma *gt_imp_wt_ge*: $t >_t s \implies \text{wt } t \geq \text{wt } s$
 $\langle \text{proof} \rangle$

lemma *not_extf_gt_nil_singleton_if_delta_eq_epsilon*:
assumes *wary_s*: $\text{wary } s$ **and** $\delta_eq_epsilon$: $\delta = \epsilon$
shows $\neg \text{extf } f (>_t) [] [s]$
 $\langle \text{proof} \rangle$

lemma *gt_sub_arg*: $\text{wary } (\text{App } s t) \implies \text{App } s t >_t t$
 $\langle \text{proof} \rangle$

lemma *gt_arg*: $\text{wary } s \implies \text{is_App } s \implies s >_t \text{arg } s$
 $\langle \text{proof} \rangle$

theorem *gt_trans*: $\text{wary } u \implies \text{wary } t \implies \text{wary } s \implies u >_t t \implies t >_t s \implies u >_t s$
 $\langle \text{proof} \rangle$

lemma *gt_antisym*: $\text{wary } s \implies \text{wary } t \implies t >_t s \implies \neg s >_t t$
 $\langle \text{proof} \rangle$

4.6 Subterm Property

lemma *gt_sub_fun*: $\text{App } s t >_t s$
 $\langle \text{proof} \rangle$

theorem *gt_proper_sub*: $\text{wary } t \implies \text{proper_sub } s t \implies t >_t s$
 $\langle \text{proof} \rangle$

4.7 Compatibility with Functions

theorem *gt_compat_fun*:

assumes

wary_t: *wary t* **and**

t'_gt_t: $t' >_t t$

shows $App\ s\ t' >_t App\ s\ t$

<proof>

4.8 Compatibility with Arguments

theorem *gt_compat_arg*:

assumes *wary_s't*: *wary (App s' t)* **and** *s'_gt_s*: $s' >_t s$

shows $App\ s'\ t >_t App\ s\ t$

<proof>

4.9 Stability under Substitution

definition *extra_wt* :: $('v \Rightarrow ('s, 'v)\ tm) \Rightarrow ('s, 'v)\ tm \Rightarrow nat$ **where**

$extra_wt\ \rho\ s = (\sum x \in \# vars_mset\ s. wt\ (\rho\ x) - wt\ (Hd\ (Var\ x)))$

lemma

extra_wt_Var[simp]: $extra_wt\ \rho\ (Hd\ (Var\ x)) = wt\ (\rho\ x) - wt\ (Hd\ (Var\ x))$ **and**

extra_wt_Sym[simp]: $extra_wt\ \rho\ (Hd\ (Sym\ f)) = 0$ **and**

extra_wt_App[simp]: $extra_wt\ \rho\ (App\ s\ t) = extra_wt\ \rho\ s + extra_wt\ \rho\ t$

<proof>

lemma *extra_wt_subseteq*:

assumes *vars_s*: $vars_mset\ t \supseteq \# vars_mset\ s$

shows $extra_wt\ \rho\ t \geq extra_wt\ \rho\ s$

<proof>

lemma *wt_subst*:

assumes *wary_rho*: *wary_subst rho* **and** *wary_s*: *wary s*

shows $wt\ (subst\ \rho\ s) = wt\ s + extra_wt\ \rho\ s$

<proof>

theorem *gt_subst*:

assumes *wary_rho*: *wary_subst rho*

shows $wary\ t \implies wary\ s \implies t >_t s \implies subst\ \rho\ t >_t subst\ \rho\ s$

<proof>

4.10 Totality on Ground Terms

theorem *gt_total_ground*:

assumes *extf_total*: $\bigwedge f. ext_total\ (extf\ f)$

shows $ground\ t \implies ground\ s \implies t >_t s \vee s >_t t \vee t = s$

<proof>

4.11 Well-foundedness

abbreviation *gtw* :: $('s, 'v)\ tm \Rightarrow ('s, 'v)\ tm \Rightarrow bool$ (**infix** $\langle >_{tw} \rangle$ 50) **where**

$\langle >_{tw} \rangle \equiv \lambda t\ s. wary\ t \wedge wary\ s \wedge t >_t s$

abbreviation *gtwg* :: $('s, 'v)\ tm \Rightarrow ('s, 'v)\ tm \Rightarrow bool$ (**infix** $\langle >_{twg} \rangle$ 50) **where**

$\langle >_{twg} \rangle \equiv \lambda t\ s. ground\ t \wedge t >_{tw}\ s$

lemma *ground_gt_unary*:

assumes *gr_t*: *ground t*

shows $\neg gt_unary\ t\ s$

<proof>

theorem *gt_wf*: *wfP* $(\lambda s\ t. t >_{tw}\ s)$

<proof>

end

end

5 The Graceful Basic Knuth–Bendix Order for Lambda-Free Higher-Order Terms

```
theory Lambda_Free_KBO_Basic
imports Lambda_Free_KBO_Std
begin
```

This theory defines the basic version of the graceful Knuth–Bendix order (KBO) for λ -free higher-order terms. Basic means that all symbols must have a positive weight. The results are lifted from the standard KBO.

```
locale kbo_basic = kbo_basic_basis _ _ _ ground_heads_var
  for ground_heads_var :: 'v  $\Rightarrow$  's set
begin
```

```
sublocale kbo_std: kbo_std _ _ _ 0 _  $\lambda$ _.  $\infty$   $\lambda$ _.  $\infty$ 
  <proof>
```

```
fun wt :: ('s, 'v) tm  $\Rightarrow$  nat where
  wt (Hd  $\zeta$ ) = (LEAST w.  $\exists f \in$  ground_heads  $\zeta$ .  $w =$  wt_sym f)
| wt (App s t) = wt s + wt t
```

```
inductive gt :: ('s, 'v) tm  $\Rightarrow$  ('s, 'v) tm  $\Rightarrow$  bool (infix <math>\langle \rangle_t</math> 50) where
  gt_wt: vars_mset t  $\supseteq\#$  vars_mset s  $\Longrightarrow$  wt t  $>$  wt s  $\Longrightarrow$  t  $>_t$  s
| gt_diff: vars_mset t  $\supseteq\#$  vars_mset s  $\Longrightarrow$  wt t = wt s  $\Longrightarrow$  head t  $>_{hd}$  head s  $\Longrightarrow$  t  $>_t$  s
| gt_same: vars_mset t  $\supseteq\#$  vars_mset s  $\Longrightarrow$  wt t = wt s  $\Longrightarrow$  head t = head s  $\Longrightarrow$ 
  ( $\forall f \in$  ground_heads (head s). extf f ( $>_t$ ) (args t) (args s))  $\Longrightarrow$  t  $>_t$  s
```

```
lemma arity_hd_eq_inf[simp]: arity_hd  $\zeta = \infty$ 
  <proof>
```

```
lemma waryI[intro, simp]: wary s
  <proof>
```

```
lemma basic_wt_eq_wt: wt s = kbo_std.wt s
  <proof>
```

```
lemma
  basic_gt_and_gt_le_gt: ( $\lambda t s$ . t  $>_t$  s  $\wedge$  local.kbo_std.gt t s)  $\leq$  kbo_std.gt and
  gt_and_basic_gt_le_basic_gt: ( $\lambda t s$ . local.kbo_std.gt t s  $\wedge$  t  $>_t$  s)  $\leq$  ( $>_t$ )
  <proof>
```

```
lemma basic_gt_iff_gt: t  $>_t$  s  $\longleftrightarrow$  kbo_std.gt t s
  <proof>
```

```
theorem gt_irrefl:  $\neg$  s  $>_t$  s
  <proof>
```

```
theorem gt_trans: u  $>_t$  t  $\Longrightarrow$  t  $>_t$  s  $\Longrightarrow$  u  $>_t$  s
  <proof>
```

```
theorem gt_proper_sub: proper_sub s t  $\Longrightarrow$  t  $>_t$  s
  <proof>
```

```
theorem gt_compat_fun: t'  $>_t$  t  $\Longrightarrow$  App s t'  $>_t$  App s t
  <proof>
```

```
theorem gt_compat_arg: s'  $>_t$  s  $\Longrightarrow$  App s' t  $>_t$  App s t
```

<proof>

lemma *wt_subst*: *wary_subst* $\rho \implies wt (subst \rho s) = wt s + kbo_std.extra_wt \rho s$
<proof>

theorem *gt_subst*: *wary_subst* $\rho \implies t >_t s \implies subst \rho t >_t subst \rho s$
<proof>

theorem *gt_wf*: *wfP* ($\lambda s t. t >_t s$)
<proof>

end

end

6 The Graceful Transfinite Knuth–Bendix Order with Subterm Coefficients for Lambda-Free Higher-Order Terms

theory *Lambda_Free_TKBO_Coefs*
imports *Lambda_Free_KBO_Util Nested_Multisets_Ordinals.Signed_Syntactic_Ordinal*
abbrevs $=_p =_{=p}$
 and $>_p = >_p$
 and $\geq_p = \geq_p$
 and $>_t = >_t$
 and $\geq_t = \geq_t$
 and $!h =_h$
begin

This theory defines the graceful transfinite Knuth–Bendix order (KBO) with subterm coefficients for λ -free higher-order terms. The proof was developed by copying that of the standard KBO and generalizing it along two axes: subterm coefficients and ordinals. Both features complicate the definitions and proofs substantially.

6.1 Setup

locale *tkbo_coefs* = *kbo_std_basis* $_{_}$ $_{_}$ *arity_sym* *arity_var* *wt_sym*
 for
 arity_sym $:: 's \Rightarrow enat$ **and**
 arity_var $:: 'v \Rightarrow enat$ **and**
 wt_sym $:: 's \Rightarrow hmultiset +$
 fixes *coef_sym* $:: 's \Rightarrow nat \Rightarrow hmultiset$
 assumes *coef_sym_gt_0*: *coef_sym* $f i > 0$
begin

abbreviation $\delta_h :: hmultiset$ **where**
 $\delta_h \equiv of_nat \delta$

abbreviation $\varepsilon_h :: hmultiset$ **where**
 $\varepsilon_h \equiv of_nat \varepsilon$

abbreviation *arity_sym_h* $:: 's \Rightarrow hmultiset$ **where**
arity_sym_h $f \equiv hmsset_of_enat (arity_sym f)$

abbreviation *arity_var_h* $:: 'v \Rightarrow hmultiset$ **where**
arity_var_h $f \equiv hmsset_of_enat (arity_var f)$

abbreviation *arity_hd_h* $:: ('s, 'v) hd \Rightarrow hmultiset$ **where**
arity_hd_h $f \equiv hmsset_of_enat (arity_hd f)$

abbreviation *arity_h* $:: ('s, 'v) tm \Rightarrow hmultiset$ **where**
arity_h $s \equiv hmsset_of_enat (arity s)$

lemma *arity_h_conv*: *arity_h* $s = arity_hd_h (head s) - of_nat (num_args s)$

<proof>

lemma *arity_h_App[simp]*: $\text{arity}_h (\text{App } s \ t) = \text{arity}_h \ s - 1$
<proof>

lemmas *wary_App_h[intro]* = *wary_App[folded of nat_lt_hmset_of_enat_iff]*

lemmas *wary_AppE_h* = *wary_AppE[folded of nat_lt_hmset_of_enat_iff]*

lemmas *wary_num_args_le_arity_head_h* =

wary_num_args_le_arity_head[folded of nat_le_hmset_of_enat_iff]

lemmas *wary_apps_h* = *wary_apps[folded of nat_le_hmset_of_enat_iff]*

lemmas *wary_cases_apps_h[consumes 1, case_names apps]* =
wary_cases_apps[folded of nat_le_hmset_of_enat_iff]

lemmas *ground_heads_arity_h* = *ground_heads_arity[folded hmset_of_enat_le]*

lemmas *some_ground_head_arity_h* = *some_ground_head_arity[folded hmset_of_enat_le]*

lemmas $\varepsilon_h \text{_gt } 0 = \varepsilon \text{_gt } 0$ [folded of nat_less_hmset, unfolded of nat_0]

lemmas $\delta_h \text{_le } \varepsilon_h = \delta \text{_le } \varepsilon$ [folded of nat_le_hmset]

lemmas *arity_hd_h_lt_ω_if_δ_h_gt_0* = *arity_hd_ne_infinity_if_δ_gt_0*
[folded of nat_less_hmset, unfolded of nat_0, folded hmset_of_enat_lt_iff_ne_infinity]

lemma *wt_sym_ge_h*: $\text{wt_sym } f \geq \varepsilon_h - \delta_h * \text{arity_sym}_h \ f$
<proof>

lemmas *unary_wt_sym_0_gt_h* = *unary_wt_sym_0_gt[folded hmset_of_enat_inject, unfolded hmset_of_enat_1]*

lemmas *unary_wt_sym_0_imp_δ_h_eq_ε_h* = *unary_wt_sym_0_imp_δ_eq_ε*
[folded of nat_inject_hmset, unfolded of nat_0]

lemmas *extf_ext_snoc_if_δ_h_eq_ε_h* = *extf_ext_snoc_if_δ_eq_ε* [folded of nat_inject_hmset]

lemmas *extf_snoc_if_δ_h_eq_ε_h* = *ext_snoc.snoc[OF extf_ext_snoc_if_δ_h_eq_ε_h]*

lemmas *arity_sym_h_lt_ω_if_δ_h_gt_0* = *arity_sym_ne_infinity_if_δ_gt_0*
[folded of nat_less_hmset hmset_of_enat_lt_iff_ne_infinity, unfolded of nat_0]

lemmas *arity_var_h_lt_ω_if_δ_h_gt_0* = *arity_var_ne_infinity_if_δ_gt_0*
[folded of nat_less_hmset hmset_of_enat_lt_iff_ne_infinity, unfolded of nat_0]

lemmas *arity_h_lt_ω_if_δ_h_gt_0* = *arity_ne_infinity_if_δ_gt_0*
[folded of nat_less_hmset hmset_of_enat_lt_iff_ne_infinity, unfolded of nat_0]

lemmas *warywary_subst_subst_h_conv* = *wary_subst_def* [folded hmset_of_enat_le]

lemmas *extf_singleton_nil_if_δ_h_eq_ε_h* = *extf_singleton_nil_if_δ_eq_ε* [folded of nat_inject_hmset]

lemma *arity_sym_h_if_δ_h_gt_0_E*:
assumes $\delta \text{_gt } 0$: $\delta_h > 0$
obtains n **where** $\text{arity_sym}_h \ f = \text{of_nat } n$
<proof>

lemma *arity_var_h_if_δ_h_gt_0_E*:
assumes $\delta \text{_gt } 0$: $\delta_h > 0$
obtains n **where** $\text{arity_var}_h \ f = \text{of_nat } n$
<proof>

6.2 Weights and Subterm Coefficients

abbreviation *zhmset_of_tpoly* :: $('a, \text{hmultiset}) \text{tpoly} \Rightarrow ('a, \text{zhmultiset}) \text{tpoly}$ **where**
 $\text{zhmset_of_tpoly} \equiv \text{map_tpoly } (\lambda x. x) \ \text{zhmset_of}$

abbreviation *eval_ztpoly* :: $('a \Rightarrow \text{zhmultiset}) \Rightarrow ('a, \text{hmultiset}) \text{tpoly} \Rightarrow \text{zhmultiset}$ **where**
 $\text{eval_ztpoly } A \ p \equiv \text{eval_tpoly } A \ (\text{zhmset_of_tpoly } p)$

lemma *eval_tpoly_eq_eval_ztpoly[simp]*:
 $\text{zhmset_of } (\text{eval_tpoly } A \ p) = \text{eval_ztpoly } (\lambda v. \text{zhmset_of } (A \ v)) \ p$
<proof>

definition *min_ground_head* :: $('s, 'v) \text{hd} \Rightarrow 's$ **where**
 $\text{min_ground_head } \zeta =$
 $(\text{SOME } f. f \in \text{ground_heads } \zeta \wedge$
 $(\forall g \in \text{ground_heads } \zeta. \text{wt_sym } g + \delta_h * \text{arity_sym}_h \ g \geq \text{wt_sym } f + \delta_h * \text{arity_sym}_h \ f))$

datatype 'va pvar =
 PWt 'va
 | PCoef 'va nat

primrec min_passign :: 'v pvar \Rightarrow hmultiset **where**
 min_passign (PWt x) = wt_sym (min_ground_head (Var x))
 | min_passign (PCoef _ _) = 1

abbreviation min_zpassign :: 'v pvar \Rightarrow zhmultiset **where**
 min_zpassign v \equiv zhmsset_of (min_passign v)

lemma min_zpassign_simps[simp]:
 min_zpassign (PWt x) = zhmsset_of (wt_sym (min_ground_head (Var x)))
 min_zpassign (PCoef x i) = 1
 <proof>

definition legal_passign :: ('v pvar \Rightarrow hmultiset) \Rightarrow bool **where**
 legal_passign A \longleftrightarrow ($\forall x. A x \geq$ min_passign x)

definition legal_zpassign :: ('v pvar \Rightarrow zhmultiset) \Rightarrow bool **where**
 legal_zpassign A \longleftrightarrow ($\forall x. A x \geq$ min_zpassign x)

lemma legal_min_passign: legal_passign min_passign
 <proof>

lemma legal_min_zpassign: legal_zpassign min_zpassign
 <proof>

lemma assign_ge_0[intro]: legal_zpassign A \Longrightarrow A x \geq 0
 <proof>

definition
 eq_tpoly :: ('v pvar, hmultiset) tpoly \Rightarrow ('v pvar, hmultiset) tpoly \Rightarrow bool (**infix** $\langle =_p \rangle$ 50)
where
 q $=_p$ p \longleftrightarrow ($\forall A. legal_zpassign A \longrightarrow eval_ztpoly A q = eval_ztpoly A p$)

definition
 ge_tpoly :: ('v pvar, hmultiset) tpoly \Rightarrow ('v pvar, hmultiset) tpoly \Rightarrow bool (**infix** $\langle \geq_p \rangle$ 50)
where
 q \geq_p p \longleftrightarrow ($\forall A. legal_zpassign A \longrightarrow eval_ztpoly A q \geq eval_ztpoly A p$)

definition
 gt_tpoly :: ('v pvar, hmultiset) tpoly \Rightarrow ('v pvar, hmultiset) tpoly \Rightarrow bool (**infix** $\langle >_p \rangle$ 50)
where
 q $>_p$ p \longleftrightarrow ($\forall A. legal_zpassign A \longrightarrow eval_ztpoly A q > eval_ztpoly A p$)

lemma gt_tpoly_imp_ge[intro]: q $>_p$ p \Longrightarrow q \geq_p p
 <proof>

lemma eq_tpoly_refl[simp]: p $=_p$ p
 <proof>

lemma ge_tpoly_refl[simp]: p \geq_p p
 <proof>

lemma gt_tpoly_irrefl: \neg p $>_p$ p
 <proof>

lemma
 eq_eq_tpoly_trans: r $=_p$ q \Longrightarrow q $=_p$ p \Longrightarrow r $=_p$ p **and**
 eq_ge_tpoly_trans: r $=_p$ q \Longrightarrow q \geq_p p \Longrightarrow r \geq_p p **and**
 eq_gt_tpoly_trans: r $=_p$ q \Longrightarrow q $>_p$ p \Longrightarrow r $>_p$ p **and**
 ge_eq_tpoly_trans: r \geq_p q \Longrightarrow q $=_p$ p \Longrightarrow r \geq_p p **and**

$ge_ge_tpoly_trans: r \geq_p q \implies q \geq_p p \implies r \geq_p p$ **and**
 $ge_gt_tpoly_trans: r \geq_p q \implies q >_p p \implies r >_p p$ **and**
 $gt_eq_tpoly_trans: r >_p q \implies q =_p p \implies r >_p p$ **and**
 $gt_ge_tpoly_trans: r >_p q \implies q \geq_p p \implies r >_p p$ **and**
 $gt_gt_tpoly_trans: r >_p q \implies q >_p p \implies r >_p p$
 ⟨proof⟩

primrec $coef_hd :: ('s, 'v) hd \Rightarrow nat \Rightarrow ('v\ pvar, hmultiset) tpoly$ **where**
 $coef_hd (Var\ x)\ i = PVar\ (PCoef\ x\ i)$
 $| coef_hd (Sym\ f)\ i = PNum\ (coef_sym\ f\ i)$

lemma $coef_hd_gt_0:$
assumes $legal: legal_zpassign\ A$
shows $eval_ztpoly\ A\ (coef_hd\ \zeta\ i) > 0$
 ⟨proof⟩

primrec $coef :: ('s, 'v) tm \Rightarrow nat \Rightarrow ('v\ pvar, hmultiset) tpoly$ **where**
 $coef (Hd\ \zeta)\ i = coef_hd\ \zeta\ i$
 $| coef (App\ s\ _)\ i = coef\ s\ (i + 1)$

lemma $coef_apps[simp]: coef\ (apps\ s\ ss)\ i = coef\ s\ (i + length\ ss)$
 ⟨proof⟩

lemma $coef_gt_0: legal_zpassign\ A \implies eval_ztpoly\ A\ (coef\ s\ i) > 0$
 ⟨proof⟩

lemma $exists_min_ground_head:$
 $\exists f. f \in ground_heads\ \zeta \wedge$
 $(\forall g \in ground_heads\ \zeta. wt_sym\ g + \delta_h * arity_sym_h\ g \geq wt_sym\ f + \delta_h * arity_sym_h\ f)$
 ⟨proof⟩

lemma $min_ground_head_Sym[simp]: min_ground_head\ (Sym\ f) = f$
 ⟨proof⟩

lemma $min_ground_head_in_ground_heads: min_ground_head\ \zeta \in ground_heads\ \zeta$
 ⟨proof⟩

lemma $min_ground_head_min:$
 $f \in ground_heads\ \zeta \implies$
 $wt_sym\ f + \delta_h * arity_sym_h\ f \geq wt_sym\ (min_ground_head\ \zeta) + \delta_h * arity_sym_h\ (min_ground_head\ \zeta)$
 ⟨proof⟩

lemma $min_ground_head_antimono:$
 $ground_heads\ \zeta \subseteq ground_heads\ \xi \implies$
 $wt_sym\ (min_ground_head\ \zeta) + \delta_h * arity_sym_h\ (min_ground_head\ \zeta)$
 $\geq wt_sym\ (min_ground_head\ \xi) + \delta_h * arity_sym_h\ (min_ground_head\ \xi)$
 ⟨proof⟩

primrec $wt0 :: ('s, 'v) hd \Rightarrow ('v\ pvar, hmultiset) tpoly$ **where**
 $wt0 (Var\ x) = PVar\ (PWt\ x)$
 $| wt0 (Sym\ f) = PNum\ (wt_sym\ f)$

lemma $wt0_ge_min_ground_head:$
 $legal_zpassign\ A \implies eval_ztpoly\ A\ (wt0\ \zeta) \geq zhmsset_of\ (wt_sym\ (min_ground_head\ \zeta))$
 ⟨proof⟩

lemma $eval_ztpoly_nonneg: legal_zpassign\ A \implies eval_ztpoly\ A\ p \geq 0$
 ⟨proof⟩

lemma $in_zip_imp_size_lt_apps: (s, y) \in set\ (zip\ ss\ ys) \implies size\ s < size\ (apps\ (Hd\ \zeta)\ ss)$
 ⟨proof⟩

function $wt :: ('s, 'v) tm \Rightarrow ('v\ pvar, hmultiset) tpoly$ **where**

$wt (apps (Hd \zeta) ss) =$
 $PSum ([wt0 \zeta, PNum (\delta_h * (arity_sym_h (\min_ground_head \zeta) - of_nat (length ss)))] @$
 $map (\lambda(s, i). PMult [coef_hd \zeta i, wt s]) (zip ss [0..<length ss]))$
 <proof>

termination

<proof>

definition

$wt_args :: nat \Rightarrow ('v \text{ pvar} \Rightarrow zhmultiset) \Rightarrow ('s, 'v) \text{ hd} \Rightarrow ('s, 'v) \text{ tm list} \Rightarrow zhmultiset$

where

$wt_args i A \zeta ss = sum_list$
 $(map (eval_ztpoly A \circ (\lambda(s, i). PMult [coef_hd \zeta i, wt s])) (zip ss [i..<i + length ss]))$

lemma $wt_Hd[simp]$: $wt (Hd \zeta) = PSum [wt0 \zeta, PNum (\delta_h * arity_sym_h (\min_ground_head \zeta))]$

<proof>

lemma $coef_hd_cong$:

$(\forall x \in vars_hd \zeta. \forall i. A (PCoef x i) = B (PCoef x i)) \implies$
 $eval_ztpoly A (coef_hd \zeta i) = eval_ztpoly B (coef_hd \zeta i)$
 <proof>

lemma $wt0_cong$:

assumes $pwt_eq: \forall x \in vars_hd \zeta. A (PWt x) = B (PWt x)$
shows $eval_ztpoly A (wt0 \zeta) = eval_ztpoly B (wt0 \zeta)$
 <proof>

lemma wt_cong :

assumes
 $\forall x \in vars s. A (PWt x) = B (PWt x)$ **and**
 $\forall x \in vars s. \forall i. A (PCoef x i) = B (PCoef x i)$
shows $eval_ztpoly A (wt s) = eval_ztpoly B (wt s)$
 <proof>

lemma $ground_eval_ztpoly_wt_eq$: $ground s \implies eval_ztpoly A (wt s) = eval_ztpoly B (wt s)$

<proof>

lemma $exists_wt_sym$:

assumes $legal: legal_zpassign A$
shows $\exists f \in ground_heads \zeta. eval_ztpoly A (wt (Hd \zeta)) \geq zhmsset_of (wt_sym f + \delta_h * arity_sym_h f)$
 <proof>

lemma $wt_ge_e_h$:

assumes $legal: legal_zpassign A$
shows $eval_ztpoly A (wt s) \geq zhmsset_of e_h$
 <proof>

lemma $wt_args_ge_length_times_e_h$:

assumes $legal: legal_zpassign A$
shows $wt_args i A \zeta ss \geq of_nat (length ss) * zhmsset_of e_h$
 <proof>

lemma $wt_ge_delta_h$: $legal_zpassign A \implies eval_ztpoly A (wt s) \geq zhmsset_of \delta_h$

<proof>

lemma wt_gt_0 : $legal_zpassign A \implies eval_ztpoly A (wt s) > 0$

<proof>

lemma $wt_gt_delta_h_if_superunary$:

assumes
 $legal: legal_zpassign A$ **and**
 $superunary: arity_hd_h (head s) > 1$
shows $eval_ztpoly A (wt s) > zhmsset_of \delta_h$
 <proof>

lemma *wt_App_plus_δ_h_ge*:
 $eval_ztpoly\ A\ (wt\ (App\ s\ t)) + zhmsset_of\ \delta_h$
 $\geq eval_ztpoly\ A\ (wt\ s) + eval_ztpoly\ A\ (coef\ s\ 0) * eval_ztpoly\ A\ (wt\ t)$
 ⟨proof⟩

lemma *wt_App_fun_δ_h*:
assumes
legal: *legal_zpassign* *A* **and**
wt_st: $eval_ztpoly\ A\ (wt\ (App\ s\ t)) = eval_ztpoly\ A\ (wt\ t)$
shows $eval_ztpoly\ A\ (wt\ s) = zhmsset_of\ \delta_h$
 ⟨proof⟩

lemma *wt_App_arg_δ_h*:
assumes
legal: *legal_zpassign* *A* **and**
wt_st: $eval_ztpoly\ A\ (wt\ (App\ s\ t)) = eval_ztpoly\ A\ (wt\ s)$
shows $eval_ztpoly\ A\ (wt\ t) = zhmsset_of\ \delta_h$
 ⟨proof⟩

lemma *wt_App_ge_fun*: $wt\ (App\ s\ t) \geq_p\ wt\ s$
 ⟨proof⟩

lemma *wt_App_ge_arg*: $wt\ (App\ s\ t) \geq_p\ wt\ t$
 ⟨proof⟩

lemma *wt_δ_h_imp_δ_h_eq_ε_h*:
assumes
legal: *legal_zpassign* *A* **and**
wt_s_eq_δ: $eval_ztpoly\ A\ (wt\ s) = zhmsset_of\ \delta_h$
shows $\delta_h = \varepsilon_h$
 ⟨proof⟩

lemma *wt_ge_vars*: $wt\ t \geq_p\ wt\ s \implies vars\ t \supseteq vars\ s$
 ⟨proof⟩

lemma *sum_coefs_ge_num_args_if_δ_h_eq_0*:
assumes
legal: *legal_passign* *A* **and**
δ_eq_0: $\delta_h = 0$ **and**
wary_s: *wary* *s*
shows $sum_coefs\ (eval_tpoly\ A\ (wt\ s)) \geq num_args\ s$
 ⟨proof⟩

6.3 Inductive Definitions

inductive *gt* :: (*'s*, *'v*) *tm* ⇒ (*'s*, *'v*) *tm* ⇒ bool (**infix** <>_{*t*} 50) **where**
gt_wt: $wt\ t >_p\ wt\ s \implies t >_t\ s$
| *gt_unary*: $wt\ t \geq_p\ wt\ s \implies \neg head\ t \leq_{hd}\ head\ s \implies num_args\ t = 1 \implies$
 $(\exists f \in ground_heads\ (head\ t).\ arity_sym\ f = 1 \wedge wt_sym\ f = 0) \implies arg\ t >_t\ s \vee arg\ t = s \implies$
 $t >_t\ s$
| *gt_diff*: $wt\ t \geq_p\ wt\ s \implies head\ t >_{hd}\ head\ s \implies t >_t\ s$
| *gt_same*: $wt\ t \geq_p\ wt\ s \implies head\ t = head\ s \implies$
 $(\forall f \in ground_heads\ (head\ t).\ extf\ f\ (>_t)\ (args\ t)\ (args\ s)) \implies t >_t\ s$

abbreviation *ge* :: (*'s*, *'v*) *tm* ⇒ (*'s*, *'v*) *tm* ⇒ bool (**infix** <>_{*t*} 50) **where**
 $t \geq_t\ s \equiv t >_t\ s \vee t = s$

inductive *gt_wt* :: (*'s*, *'v*) *tm* ⇒ (*'s*, *'v*) *tm* ⇒ bool **where**
gt_wtI: $wt\ t >_p\ wt\ s \implies gt_wt\ t\ s$

inductive *gt_unary* :: (*'s*, *'v*) *tm* ⇒ (*'s*, *'v*) *tm* ⇒ bool **where**
gt_unaryI: $wt\ t \geq_p\ wt\ s \implies \neg head\ t \leq_{hd}\ head\ s \implies num_args\ t = 1 \implies$
 $(\exists f \in ground_heads\ (head\ t).\ arity_sym\ f = 1 \wedge wt_sym\ f = 0) \implies arg\ t \geq_t\ s \implies gt_unary\ t\ s$

inductive $gt_diff :: ('s, 'v) tm \Rightarrow ('s, 'v) tm \Rightarrow bool$ **where**
 $gt_diffI: wt\ t \geq_p wt\ s \Longrightarrow head\ t >_{hd}\ head\ s \Longrightarrow gt_diff\ t\ s$

inductive $gt_same :: ('s, 'v) tm \Rightarrow ('s, 'v) tm \Rightarrow bool$ **where**
 $gt_sameI: wt\ t \geq_p wt\ s \Longrightarrow head\ t = head\ s \Longrightarrow$
 $(\forall f \in ground_heads\ (head\ t).\ extf\ f\ (>_t)\ (args\ t)\ (args\ s)) \Longrightarrow gt_same\ t\ s$

lemma $gt_iff_wt_unary_diff_same: t >_t s \longleftrightarrow gt_wt\ t\ s \vee gt_unary\ t\ s \vee gt_diff\ t\ s \vee gt_same\ t\ s$
 $\langle proof \rangle$

lemma $gt_imp_wt: t >_t s \Longrightarrow wt\ t \geq_p wt\ s$
 $\langle proof \rangle$

lemma $gt_imp_vars: t >_t s \Longrightarrow vars\ t \supseteq vars\ s$
 $\langle proof \rangle$

6.4 Irreflexivity

theorem $gt_irrefl: wary\ s \Longrightarrow \neg s >_t s$
 $\langle proof \rangle$

6.5 Transitivity

lemma $not_extf_gt_nil_singleton_if\ \delta_h\ eq\ \varepsilon_h:$
assumes $wary_s: wary\ s$ **and** $\delta_eq\ \varepsilon: \delta_h = \varepsilon_h$
shows $\neg extf\ f\ (>_t)\ []\ [s]$
 $\langle proof \rangle$

lemma $gt_sub_arg: wary\ (App\ s\ t) \Longrightarrow App\ s\ t >_t t$
 $\langle proof \rangle$

lemma $gt_arg: wary\ s \Longrightarrow is_App\ s \Longrightarrow s >_t arg\ s$
 $\langle proof \rangle$

theorem $gt_trans: wary\ u \Longrightarrow wary\ t \Longrightarrow wary\ s \Longrightarrow u >_t t \Longrightarrow t >_t s \Longrightarrow u >_t s$
 $\langle proof \rangle$

lemma $gt_antisym: wary\ s \Longrightarrow wary\ t \Longrightarrow t >_t s \Longrightarrow \neg s >_t t$
 $\langle proof \rangle$

6.6 Subterm Property

lemma $gt_sub_fun: App\ s\ t >_t s$
 $\langle proof \rangle$

theorem $gt_proper_sub: wary\ t \Longrightarrow proper_sub\ s\ t \Longrightarrow t >_t s$
 $\langle proof \rangle$

6.7 Compatibility with Functions

lemma $gt_compat_fun:$
assumes
 $wary_t: wary\ t$ **and**
 $t'_gt_t: t' >_t t$
shows $App\ s\ t' >_t App\ s\ t$
 $\langle proof \rangle$

theorem $gt_compat_fun_strong:$
assumes
 $wary_t: wary\ t$ **and**
 $t'_gt_t: t' >_t t$
shows $apps\ s\ (t' \# us) >_t apps\ s\ (t \# us)$
 $\langle proof \rangle$

6.8 Compatibility with Arguments

theorem *gt_compat_arg_weak*:

assumes

wary_st: *wary* (*App* *s* *t*) **and**

wary_s't: *wary* (*App* *s'* *t*) **and**

coef_s'_0_ge_s: *coef* *s'* 0 \geq_p *coef* *s* 0 **and**

s'_gt_s: *s'* $>_t$ *s*

shows *App* *s'* *t* $>_t$ *App* *s* *t*

<proof>

6.9 Stability under Substitution

primrec

subst_zpassign :: (*'v* \Rightarrow (*'s*, *'v*) *tm*) \Rightarrow (*'v* *pvar* \Rightarrow *zhmultiset*) \Rightarrow *'v* *pvar* \Rightarrow *zhmultiset*

where

subst_zpassign ϱ *A* (*PWt* *x*) =

eval_ztpoly *A* (*wt* (ϱ *x*)) - *zhmset_of* (δ_h * *arity_sym_h* (*min_ground_head* (*Var* *x*)))

| *subst_zpassign* ϱ *A* (*PCoef* *x* *i*) = *eval_ztpoly* *A* (*coef* (ϱ *x*) *i*)

lemma *legal_subst_zpassign*:

assumes

legal: *legal_zpassign* *A* **and**

wary_ρ: *wary_subst* ϱ

shows *legal_zpassign* (*subst_zpassign* ϱ *A*)

<proof>

lemma *wt_subst*:

assumes

legal: *legal_zpassign* *A* **and**

wary_ρ: *wary_subst* ϱ

shows *wary* *s* \Longrightarrow *eval_ztpoly* *A* (*wt* (*subst* ϱ *s*)) = *eval_ztpoly* (*subst_zpassign* ϱ *A*) (*wt* *s*)

<proof>

theorem *gt_subst*:

assumes *wary_ρ*: *wary_subst* ϱ

shows *wary* *t* \Longrightarrow *wary* *s* \Longrightarrow *t* $>_t$ *s* \Longrightarrow *subst* ϱ *t* $>_t$ *subst* ϱ *s*

<proof>

6.10 Totality on Ground Terms

lemma *wt_total_ground*:

assumes

gr_t: *ground* *t* **and**

gr_s: *ground* *s*

shows *wt* *t* $>_p$ *wt* *s* \vee *wt* *s* $>_p$ *wt* *t* \vee *wt* *t* $=_p$ *wt* *s*

<proof>

theorem *gt_total_ground*:

assumes *extf_total*: $\bigwedge f. \text{ext_total } (extf f)$

shows *ground* *t* \Longrightarrow *ground* *s* \Longrightarrow *t* $>_t$ *s* \vee *s* $>_t$ *t* \vee *t* $=$ *s*

<proof>

6.11 Well-foundedness

abbreviation *gtw* :: (*'s*, *'v*) *tm* \Rightarrow (*'s*, *'v*) *tm* \Rightarrow *bool* (**infix** $\langle >_{tw} \rangle$ 50) **where**

$\langle >_{tw} \rangle \equiv \lambda t s. \text{wary } t \wedge \text{wary } s \wedge t >_t s$

abbreviation *gtwg* :: (*'s*, *'v*) *tm* \Rightarrow (*'s*, *'v*) *tm* \Rightarrow *bool* (**infix** $\langle >_{twg} \rangle$ 50) **where**

$\langle >_{twg} \rangle \equiv \lambda t s. \text{ground } t \wedge t >_{tw} s$

lemma *ground_gt_unary*:

assumes *gr_t*: *ground* *t*

shows $\neg gt_unary\ t\ s$
 $\langle proof \rangle$

theorem $gt_wf: wfP\ (\lambda s\ t. t >_{tw}\ s)$
 $\langle proof \rangle$

end

end

7 Properties of Lambda-Free KBO on the Lambda Encoding

theory *Lambda_Encoding_KBO*
imports *Lambda_Free_RPOs.Lambda_Encoding Lambda_Free_KBO_Basic*
begin

This theory explores the properties of the λ -free KBO on the proposed encoding of λ -expressions.

locale $kbo_lambda_encoding = kbo_basic\ _____\ \lambda_ :: 'v. UNIV :: 's\ set + lambda_encoding\ lam$
for $lam :: 's +$
assumes

$gt_db_db: j > i \implies db\ j >_s db\ i$ **and**
 $wt_db_db: wt_sym\ (db\ j) = wt_sym\ (db\ i)$

begin

notation gt (**infix** $\langle >_t \rangle$ 50)

notation gt_hd (**infix** $\langle >_{hd} \rangle$ 50)

abbreviation $ge :: ('s, 'v)\ tm \Rightarrow ('s, 'v)\ tm \Rightarrow bool$ (**infix** $\langle \geq_t \rangle$ 50) **where**
 $t \geq_t s \equiv t >_t s \vee t = s$

lemma $wary_raw_db_subst: wary_subst\ (raw_db_subst\ i\ x)$
 $\langle proof \rangle$

lemma $wt_subst_db: wt\ (subst_db\ i\ x\ s) = wt\ (subst\ (raw_db_subst\ j\ x)\ s)$
 $\langle proof \rangle$

lemma $subst_db_Suc_ge: subst_db\ (Suc\ i)\ x\ s \geq_t subst_db\ i\ x\ s$
 $\langle proof \rangle$

lemma $gt_subst_db: t >_t s \implies subst_db\ i\ x\ t >_t subst_db\ i\ x\ s$
 $\langle proof \rangle$

end

end

8 Knuth–Bendix Orders for Lambda-Free Higher-Order Terms

theory *Lambda_Free_KBOs*
imports *Lambda_Free_KBO_App Lambda_Free_KBO_Basic Lambda_Free_TKBO_Coefs Lambda_Encoding_KBO*
begin

locale *simple_kbo_instances*
begin

definition $arity_sym :: nat \Rightarrow enat$ **where**
 $arity_sym\ n = \infty$

definition $arity_var :: nat \Rightarrow enat$ **where**
 $arity_var\ n = \infty$

definition $ground_head_var :: nat \Rightarrow nat\ set$ **where**

$ground_head_var\ x = UNIV$

definition $gt_sym :: nat \Rightarrow nat \Rightarrow bool$ **where**
 $gt_sym\ g\ f \longleftrightarrow g > f$

definition $\varepsilon :: nat$ **where**
 $\varepsilon = 1$

definition $\delta :: nat$ **where**
 $\delta = 0$

definition $wt_sym :: nat \Rightarrow nat$ **where**
 $wt_sym\ n = 1$

definition $wt_sym_h :: nat \Rightarrow hmultiset$ **where**
 $wt_sym_h\ n = 1$

definition $coef_sym_h :: nat \Rightarrow nat \Rightarrow hmultiset$ **where**
 $coef_sym_h\ n\ i = 1$

sublocale kbo_app : $kbo_app\ gt_sym\ wt_sym\ \varepsilon\ len_lexext$
 $\langle proof \rangle$

sublocale kbo_basic : $kbo_basic\ gt_sym\ wt_sym\ \varepsilon\ \lambda f. len_lexext\ ground_head_var$
 $\langle proof \rangle$

sublocale kbo_std : $kbo_std\ ground_head_var\ gt_sym\ \varepsilon\ \delta\ \lambda f. len_lexext\ arity_sym\ arity_var\ wt_sym$
 $\langle proof \rangle$

sublocale $tkbo_coefs$: $tkbo_coefs\ ground_head_var\ gt_sym\ \varepsilon\ \delta\ \lambda f. len_lexext\ arity_sym\ arity_var$
 $wt_sym_h\ coef_sym_h$
 $\langle proof \rangle$

end

end