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

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

July 20, 2018

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

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 our CADE-26 paper for details.[1]

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*: $\varepsilon > 0$ **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*: $\varepsilon > 0$ **and**
  δ*_le_ε*: $\delta \leq \varepsilon$ **and**
  arity_hd_ne_infinity_if_δ_gt_0: $\delta > 0 \Longrightarrow$ *arity_hd* $\zeta \neq \infty$ **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* ⟹ $\delta = \varepsilon$ **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_δ_eq_ε: $\delta = \varepsilon \Longrightarrow$ *ext_snoc* (*extf f*)
begin

[1] https://www21.in.tum.de/~blanchet/lambda_free_kbo_conf.pdf

lemma *arity_sym_ne_infinity_if_δ_gt_0*: $\delta > 0 \Longrightarrow \mathit{arity\_sym}\ f \neq \infty$
⟨*proof*⟩

lemma *arity_var_ne_infinity_if_δ_gt_0*: $\delta > 0 \Longrightarrow \mathit{arity\_var}\ x \neq \infty$
⟨*proof*⟩

lemma *arity_ne_infinity_if_δ_gt_0*: $\delta > 0 \Longrightarrow \mathit{arity}\ s \neq \infty$
⟨*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_δ_eq_ε* = *ext_snoc.snoc*[*OF extf_ext_snoc_if_δ_eq_ε*]

lemma *extf_singleton_nil_if_δ_eq_ε*: $\delta = \varepsilon \Longrightarrow \mathit{extf}\ f\ \mathit{gt}\ [s]\ []$
⟨*proof*⟩

end

sublocale *kbo_basic_basis* < *kbo_std_basis* _ _ λ_. ∞ λ_. ∞ _ _ *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 $\geq t = \geq_t$
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* :: $'s \Rightarrow 's \Rightarrow$ *bool* (**infix** $>_s$ *50*) +
 fixes
  wt_sym :: $'s \Rightarrow$ *nat* **and**
  ε :: *nat* **and**
  ext :: $(('s, 'v)\ tm \Rightarrow ('s, 'v)\ tm \Rightarrow bool) \Rightarrow ('s, 'v)\ tm\ list \Rightarrow ('s, 'v)\ tm\ list \Rightarrow bool$
 assumes
  ε*_gt_0*: $\varepsilon > 0$ **and**
  wt_sym_ge_ε: $wt\_sym\ f \geq \varepsilon$ **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*]: $gt \leq gt' \Longrightarrow ext\ gt \leq ext\ gt'$
 ⟨*proof*⟩

fun *wt* :: $('s, 'v)\ tm \Rightarrow nat$ **where**
 $wt\ (Hd\ (Var\ x)) = \varepsilon$
| $wt\ (Hd\ (Sym\ f)) = wt\_sym\ f$
| $wt\ (App\ s\ t) = wt\ s + wt\ t$

inductive *gt* :: $('s, 'v)\ tm \Rightarrow ('s, 'v)\ tm \Rightarrow bool$ (**infix** $>_t$ *50*) **where**
 gt_wt: $vars\_mset\ t \supseteq\# vars\_mset\ s \Longrightarrow wt\ t > wt\ s \Longrightarrow t >_t s$
| *gt_sym_sym*: $wt\_sym\ g = wt\_sym\ f \Longrightarrow g >_s f \Longrightarrow Hd\ (Sym\ g) >_t Hd\ (Sym\ f)$
| *gt_sym_app*: $vars\ s = \{\} \Longrightarrow wt\ t = wt\ s \Longrightarrow t = Hd\ (Sym\ g) \Longrightarrow is\_App\ s \Longrightarrow t >_t s$
| *gt_app_app*: $vars\_mset\ t \supseteq\# vars\_mset\ s \Longrightarrow wt\ t = wt\ s \Longrightarrow t = App\ t1\ t2 \Longrightarrow s = App\ s1\ s2 \Longrightarrow$
  $ext\ (>_t)\ [t1, t2]\ [s1, s2] \Longrightarrow t >_t s$

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

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*
abbrevs $>t = >_t$
 and $\geq t = \geq_t$
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 :: $'s \Rightarrow enat$ **and**
  arity_var :: $'v \Rightarrow enat$ **and**
  wt_sym :: $'s \Rightarrow nat$
begin

4.2 Weights

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

wt (*Hd* ζ) = (*LEAST* *w*. $\exists f \in$ *ground_heads* ζ. *w* = *wt_sym f* + *the_enat* (δ * *arity_sym f*))
| *wt* (*App s t*) = (*wt s* − δ) + *wt t*

lemma *wt_Hd_Sym*: *wt* (*Hd* (*Sym f*)) = *wt_sym f* + *the_enat* (δ * *arity_sym f*)
⟨*proof*⟩

lemma *exists_wt_sym*: $\exists f \in$ *ground_heads* ζ. *wt* (*Hd* ζ) = *wt_sym f* + *the_enat* (δ * *arity_sym f*)
⟨*proof*⟩

lemma *wt_le_wt_sym*: *f* $\in$ *ground_heads* ζ $\Longrightarrow$ *wt* (*Hd* ζ) $\leq$ *wt_sym f* + *the_enat* (δ * *arity_sym f*)
⟨*proof*⟩

lemma *enat_the_enat_δ_times_arity_sym*[*simp*]: *enat* (*the_enat* (δ * *arity_sym f*)) = δ * *arity_sym f*
⟨*proof*⟩

lemma *wt_arg_le*: *wt* (*arg s*) $\leq$ *wt s*
⟨*proof*⟩

lemma *wt_ge_ε*: *wt s* $\geq$ ε
⟨*proof*⟩

lemma *wt_ge_δ*: *wt s* $\geq$ δ
⟨*proof*⟩

lemma *wt_gt_δ_if_superunary*: *arity_hd* (*head s*) > 1 $\Longrightarrow$ *wt s* > δ
⟨*proof*⟩

lemma *wt_App_δ*: *wt* (*App s t*) = *wt t* $\Longrightarrow$ *wt s* = δ
⟨*proof*⟩

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

lemma *wt_hd_le*: *wt* (*Hd* (*head s*)) $\leq$ *wt s*
⟨*proof*⟩

lemma *wt_δ_imp_δ_eq_ε*: *wt s* = δ $\Longrightarrow$ δ = ε
⟨*proof*⟩

lemma *wt_ge_arity_head_if_δ_gt_0*:
assumes *δ_gt_0*: δ > 0
shows *wt s* $\geq$ *arity_hd* (*head s*)
⟨*proof*⟩

lemma *wt_ge_num_args_if_δ_eq_0*:
assumes *δ_eq_0*: δ = 0
shows *wt s* $\geq$ *num_args s*
⟨*proof*⟩

lemma *wt_ge_num_args*: *wary s* $\Longrightarrow$ *wt s* $\geq$ *num_args s*
⟨*proof*⟩

4.3 Inductive Definitions

inductive *gt* :: ('*s*, '*v*) *tm* ⇒ ('*s*, '*v*) *tm* ⇒ *bool* (**infix** $>_t$ 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* $\Longrightarrow$ ¬ *head t* $\leq\geq_{hd}$ *head s* $\Longrightarrow$ *num_args t* = 1 $\Longrightarrow$
($\exists f \in$ *ground_heads* (*head t*). *arity_sym f* = 1 ∧ *wt_sym f* = 0) $\Longrightarrow$ *arg t* $>_t$ *s* ∨ *arg t* = *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 t*). *extf f* ($>_t$) (*args t*) (*args s*)) $\Longrightarrow$ *t* $>_t$ *s*

abbreviation *ge* :: ('*s*, '*v*) *tm* ⇒ ('*s*, '*v*) *tm* ⇒ *bool* (**infix** $\geq_t$ 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: *vars_mset t* ⊇# *vars_mset s* ⟹ *wt t* > *wt s* ⟹ *gt_wt t s*

inductive *gt_diff* :: ('s, 'v) *tm* ⇒ ('s, 'v) *tm* ⇒ *bool* **where**
gt_diffI: *vars_mset t* ⊇# *vars_mset s* ⟹ *wt t* = *wt s* ⟹ *head t* $>_{hd}$ *head s* ⟹ *gt_diff t s*

inductive *gt_unary* :: ('s, 'v) *tm* ⇒ ('s, 'v) *tm* ⇒ *bool* **where**
gt_unaryI: *wt t* = *wt s* ⟹ ¬ *head t* $\leq\geq_{hd}$ *head s* ⟹ *num_args t* = 1 ⟹
(∃*f* ∈ *ground_heads* (*head t*). *arity_sym f* = 1 ∧ *wt_sym f* = 0) ⟹ *arg t* $\geq_t$ *s* ⟹ *gt_unary t s*

inductive *gt_same* :: ('s, 'v) *tm* ⇒ ('s, 'v) *tm* ⇒ *bool* **where**
gt_sameI: *vars_mset t* ⊇# *vars_mset s* ⟹ *wt t* = *wt s* ⟹ *head t* = *head s* ⟹
(∀*f* ∈ *ground_heads* (*head t*). *extf f* ($>_t$) (*args t*) (*args s*)) ⟹ *gt_same t s*

lemma *gt_iff_wt_unary_diff_same*: $t >_t s \longleftrightarrow$ *gt_wt t s* ∨ *gt_unary t s* ∨ *gt_diff t s* ∨ *gt_same t s*
⟨*proof*⟩

lemma *gt_imp_vars_mset*: $t >_t s$ ⟹ *vars_mset t* ⊇# *vars_mset s*
⟨*proof*⟩

lemma *gt_imp_vars*: $t >_t s$ ⟹ *vars t* ⊇ *vars s*
⟨*proof*⟩

4.4 Irreflexivity

theorem *gt_irefl*: *wary s* ⟹ ¬ $s >_t s$
⟨*proof*⟩

4.5 Transitivity

lemma *gt_imp_wt_ge*: $t >_t s$ ⟹ *wt t* ≥ *wt s*
⟨*proof*⟩

lemma *not_extf_gt_nil_singleton_if_δ_eq_ε*:
assumes *wary_s*: *wary s* **and** *δ_eq_ε*: δ = ε
shows ¬ *extf f* ($>_t$) [] [*s*]
⟨*proof*⟩

lemma *gt_sub_arg*: *wary* (*App s t*) ⟹ *App s t* $>_t$ *t*
⟨*proof*⟩

lemma *gt_arg*: *wary s* ⟹ *is_App s* ⟹ *s* $>_t$ *arg s*
⟨*proof*⟩

theorem *gt_trans*: *wary u* ⟹ *wary t* ⟹ *wary s* ⟹ $u >_t t$ ⟹ $t >_t s$ ⟹ $u >_t s$
⟨*proof*⟩

lemma *gt_antisym*: *wary s* ⟹ *wary t* ⟹ $t >_t s$ ⟹ ¬ $s >_t t$
⟨*proof*⟩

4.6 Subterm Property

lemma *gt_sub_fun*: *App s t* $>_t$ *s*
⟨*proof*⟩

theorem *gt_proper_sub*: *wary t* ⟹ *proper_sub s t* ⟹ $t >_t s$
⟨*proof*⟩

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**
$extra\_wt\ \varrho\ s = sum\_mset\ \{\#wt\ (\varrho\ x) - wt\ (Hd\ (Var\ x)).\ x \in\# vars\_mset\ s\#\}$

lemma
extra_wt_Var[*simp*]: $extra\_wt\ \varrho\ (Hd\ (Var\ x)) = wt\ (\varrho\ x) - wt\ (Hd\ (Var\ x))$ **and**
extra_wt_Sym[*simp*]: $extra\_wt\ \varrho\ (Hd\ (Sym\ f)) = 0$ **and**
extra_wt_App[*simp*]: $extra\_wt\ \varrho\ (App\ s\ t) = extra\_wt\ \varrho\ s + extra\_wt\ \varrho\ t$
⟨*proof*⟩

lemma *extra_wt_subseteq*:
assumes *vars_s*: $vars\_mset\ t \supseteq\# vars\_mset\ s$
shows $extra\_wt\ \varrho\ t \geq extra\_wt\ \varrho\ s$
⟨*proof*⟩

lemma *wt_subst*:
assumes *wary_ϱ*: *wary_subst ϱ* **and** *wary_s*: *wary s*
shows $wt\ (subst\ \varrho\ s) = wt\ s + extra\_wt\ \varrho\ s$
⟨*proof*⟩

theorem *gt_subst*:
assumes *wary_ϱ*: *wary_subst ϱ*
shows *wary t* $\Longrightarrow$ *wary s* $\Longrightarrow$ $t >_t s \Longrightarrow subst\ \varrho\ t >_t subst\ \varrho\ s$
⟨*proof*⟩

4.10 Totality on Ground Terms

theorem *gt_total_ground*:
assumes
extf_total: $\bigwedge f.\ ext\_total\ (extf\ f)$ **and**
gr_t: *ground t* **and**
gr_s: *ground s*
shows $t >_t s \vee s >_t t \vee t = s$
⟨*proof*⟩

4.11 Well-foundedness

abbreviation *gtw* :: $('s, 'v)\ tm \Rightarrow ('s, 'v)\ tm \Rightarrow bool$ (**infix** $>_{tw}$ *50*) **where**
$(>_{tw}) \equiv \lambda t\ s.\ wary\ t \wedge wary\ s \wedge t >_t s$

abbreviation *gtwg* :: $('s, 'v)\ tm \Rightarrow ('s, 'v)\ tm \Rightarrow bool$ (**infix** $>_{twg}$ *50*) **where**
$(>_{twg}) \equiv \lambda t\ s.\ ground\ t \wedge t >_{tw} s$

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* ⇒ *'s set*
begin

sublocale *kbo_std*: *kbo_std* _ _ _ *0* _ λ_. ∞ λ_. ∞
  ⟨*proof*⟩

fun *wt* :: (*'s*, *'v*) *tm* ⇒ *nat* **where**
  wt (*Hd* ζ) = (*LEAST w*. ∃*f* ∈ *ground_heads* ζ. *w* = *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_diff*: *vars_mset t* ⊇# *vars_mset s* ⟹ *wt t* = *wt s* ⟹ *head t* >$_{hd}$ *head s* ⟹ *t* >$_t$ *s*
| *gt_same*: *vars_mset t* ⊇# *vars_mset s* ⟹ *wt t* = *wt s* ⟹ *head t* = *head s* ⟹
    (∀*f* ∈ *ground_heads* (*head s*). *extf f* (>$_t$) (*args t*) (*args s*)) ⟹ *t* >$_t$ *s*

lemma *arity_hd_eq_inf*[*simp*]: *arity_hd* ζ = ∞
  ⟨*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: (λ*t s*. *t* >$_t$ *s* ∧ *local.kbo_std.gt t s*) ≤ *kbo_std.gt* **and**
  gt_and_basic_gt_le_basic_gt: (λ*t s*. *local.kbo_std.gt t s* ∧ *t* >$_t$ *s*) ≤ (>$_t$)
  ⟨*proof*⟩

lemma *basic_gt_iff_lt*: *t* >$_t$ *s* ⟷ *kbo_std.gt t s*
⟨*proof*⟩

theorem *gt_irrefl*: ¬ *s* >$_t$ *s*
  ⟨*proof*⟩

theorem *gt_trans*: *u* >$_t$ *t* ⟹ *t* >$_t$ *s* ⟹ *u* >$_t$ *s*
  ⟨*proof*⟩

theorem *gt_proper_sub*: *proper_sub s t* ⟹ *t* >$_t$ *s*
  ⟨*proof*⟩

theorem *gt_compat_fun*: *t'* >$_t$ *t* ⟹ *App s t'* >$_t$ *App s t*
  ⟨*proof*⟩

theorem *gt_compat_arg*: *s'* >$_t$ *s* ⟹ *App s' t* >$_t$ *App s t*
  ⟨*proof*⟩

theorem *gt_subst*: *wary_subst* $\varrho \Longrightarrow t >_t s \Longrightarrow$ *subst* $\varrho\ t >_t$ *subst* $\varrho\ s$
⟨*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

hide-const (**open**) *Complex.arg*

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 δ_h :: *hmultiset* **where**
$\delta_h \equiv$ *of_nat* δ

abbreviation ε_h :: *hmultiset* **where**
$\varepsilon_h \equiv$ *of_nat* ε

abbreviation *arity_sym*$_h$ :: $'s \Rightarrow$ *hmultiset* **where**
arity_sym$_h$ *f* $\equiv$ *hmset_of_enat* (*arity_sym f*)

abbreviation *arity_var*$_h$ :: $'v \Rightarrow$ *hmultiset* **where**
arity_var$_h$ *f* $\equiv$ *hmset_of_enat* (*arity_var f*)

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

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

lemma *arity*$_h$*_conv*: *arity*$_h$ *s* $=$ *arity_hd*$_h$ (*head s*) $-$ *of_nat* (*num_args s*)
⟨*proof*⟩

lemma $arity_h\_App[simp]$: $arity_h\ (App\ s\ t) = arity_h\ s - 1$
⟨*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 $\varepsilon_h\_gt\_0 = \varepsilon\_gt\_0[folded\ of\_nat\_less\_hmset,\ unfolded\ of\_nat\_0]$
lemmas $\delta_h\_le\_\varepsilon_h = \delta\_le\_\varepsilon[folded\ of\_nat\_le\_hmset]$
lemmas $arity\_hd_h\_lt\_\omega\_if\_\delta_h\_gt\_0 = arity\_hd\_ne\_infinity\_if\_\delta\_gt\_0$
$[folded\ of\_nat\_less\_hmset,\ unfolded\ of\_nat\_0,\ folded\ hmset\_of\_enat\_lt\_iff\_ne\_infinity]$

lemma $wt\_sym\_ge_h$: $wt\_sym\ f \geq \varepsilon_h - \delta_h * arity\_sym_h\ f$
⟨*proof*⟩

lemmas $unary\_wt\_sym\_0\_gt_h = unary\_wt\_sym\_0\_gt[folded\ hmset\_of\_enat\_inject,\ unfolded\ hmset\_of\_enat\_1]$
lemmas $unary\_wt\_sym\_0\_imp\_\delta_h\_eq\_\varepsilon_h = unary\_wt\_sym\_0\_imp\_\delta\_eq\_\varepsilon$
$[folded\ of\_nat\_inject\_hmset,\ unfolded\ of\_nat\_0]$
lemmas $extf\_ext\_snoc\_if\_\delta_h\_eq\_\varepsilon_h = extf\_ext\_snoc\_if\_\delta\_eq\_\varepsilon[folded\ of\_nat\_inject\_hmset]$
lemmas $extf\_snoc\_if\_\delta_h\_eq\_\varepsilon_h = ext\_snoc.snoc[OF\ extf\_ext\_snoc\_if\_\delta_h\_eq\_\varepsilon_h]$
lemmas $arity\_sym_h\_lt\_\omega\_if\_\delta_h\_gt\_0 = arity\_sym\_ne\_infinity\_if\_\delta\_gt\_0$
$[folded\ of\_nat\_less\_hmset\ hmset\_of\_enat\_lt\_iff\_ne\_infinity,\ unfolded\ of\_nat\_0]$
lemmas $arity\_var_h\_lt\_\omega\_if\_\delta_h\_gt\_0 = arity\_var\_ne\_infinity\_if\_\delta\_gt\_0$
$[folded\ of\_nat\_less\_hmset\ hmset\_of\_enat\_lt\_iff\_ne\_infinity,\ unfolded\ of\_nat\_0]$
lemmas $arity_h\_lt\_\omega\_if\_\delta_h\_gt\_0 = arity\_ne\_infinity\_if\_\delta\_gt\_0$
$[folded\ of\_nat\_less\_hmset\ hmset\_of\_enat\_lt\_iff\_ne\_infinity,\ unfolded\ of\_nat\_0]$
lemmas $warywary\_subst\_subst_h\_conv = wary\_subst\_def[folded\ hmset\_of\_enat\_le]$
lemmas $extf\_singleton\_nil\_if\_\delta_h\_eq\_\varepsilon_h = extf\_singleton\_nil\_if\_\delta\_eq\_\varepsilon[folded\ of\_nat\_inject\_hmset]$

lemma $arity\_sym_h\_if\_\delta_h\_gt\_0\_E$:
assumes δ_gt_0: $\delta_h > 0$
obtains n **where** $arity\_sym_h\ f = of\_nat\ n$
⟨*proof*⟩

lemma $arity\_var_h\_if\_\delta_h\_gt\_0\_E$:
assumes δ_gt_0: $\delta_h > 0$
obtains n **where** $arity\_var_h\ f = of\_nat\ n$
⟨*proof*⟩

6.2 Weights and Subterm Coefficients

abbreviation $zhmset\_of\_tpoly :: ('a,\ hmultiset)\ tpoly \Rightarrow ('a,\ zhmultiset)\ tpoly$ **where**
$zhmset\_of\_tpoly \equiv map\_tpoly\ (\lambda x.\ x)\ zhmset\_of$

abbreviation $eval\_ztpoly :: ('a \Rightarrow zhmultiset) \Rightarrow ('a,\ hmultiset)\ tpoly \Rightarrow zhmultiset$ **where**
$eval\_ztpoly\ A\ p \equiv eval\_tpoly\ A\ (zhmset\_of\_tpoly\ p)$

lemma $eval\_tpoly\_eq\_eval\_ztpoly[simp]$:
$zhmset\_of\ (eval\_tpoly\ A\ p) = eval\_ztpoly\ (\lambda v.\ zhmset\_of\ (A\ v))\ p$
⟨*proof*⟩

definition $min\_ground\_head :: ('s,\ 'v)\ hd \Rightarrow 's$ **where**
$min\_ground\_head\ \zeta =$
$(SOME\ f.\ f \in ground\_heads\ \zeta \wedge$
$(\forall g \in ground\_heads\ \zeta.\ wt\_sym\ g + \delta_h * arity\_sym_h\ g \geq wt\_sym\ f + \delta_h * arity\_sym_h\ f))$

datatype $'va\ pvar =$
$PWt\ 'va$

| *PCoef 'va nat*

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

abbreviation *min_zpassign* :: *'v pvar* ⇒ *zhmultiset* **where**
min_zpassign v ≡ *zhmset_of (min_passign v)*

lemma *min_zpassign_simps*[*simp*]:
min_zpassign (PWt x) = zhmset_of (wt_sym (min_ground_head (Var x)))
min_zpassign (PCoef x i) = 1
⟨*proof*⟩

definition *legal_passign* :: (*'v pvar* ⇒ *hmultiset*) ⇒ *bool* **where**
legal_passign A ⟷ (∀ *x*. *A x* ≥ *min_passign x*)

definition *legal_zpassign* :: (*'v pvar* ⇒ *zhmultiset*) ⇒ *bool* **where**
legal_zpassign A ⟷ (∀ *x*. *A x* ≥ *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* ⟹ *A x* ≥ *0*
⟨*proof*⟩

definition
eq_tpoly :: (*'v pvar*, *hmultiset*) *tpoly* ⇒ (*'v pvar*, *hmultiset*) *tpoly* ⇒ *bool* (**infix** =$_p$ *50*)
where
q =$_p$ *p* ⟷ (∀ *A*. *legal_zpassign A* ⟶ *eval_ztpoly A q* = *eval_ztpoly A p*)

definition
ge_tpoly :: (*'v pvar*, *hmultiset*) *tpoly* ⇒ (*'v pvar*, *hmultiset*) *tpoly* ⇒ *bool* (**infix** ≥$_p$ *50*)
where
q ≥$_p$ *p* ⟷ (∀ *A*. *legal_zpassign A* ⟶ *eval_ztpoly A q* ≥ *eval_ztpoly A p*)

definition
gt_tpoly :: (*'v pvar*, *hmultiset*) *tpoly* ⇒ (*'v pvar*, *hmultiset*) *tpoly* ⇒ *bool* (**infix** >$_p$ *50*)
where
q >$_p$ *p* ⟷ (∀ *A*. *legal_zpassign A* ⟶ *eval_ztpoly A q* > *eval_ztpoly A p*)

lemma *gt_tpoly_imp_ge*[*intro*]: *q* >$_p$ *p* ⟹ *q* ≥$_p$ *p*
⟨*proof*⟩

lemma *eq_tpoly_refl*[*simp*]: *p* =$_p$ *p*
⟨*proof*⟩

lemma *ge_tpoly_refl*[*simp*]: *p* ≥$_p$ *p*
⟨*proof*⟩

lemma *gt_tpoly_irrefl*: ¬ *p* >$_p$ *p*
⟨*proof*⟩

lemma
eq_eq_tpoly_trans: *r* =$_p$ *q* ⟹ *q* =$_p$ *p* ⟹ *r* =$_p$ *p* **and**
eq_ge_tpoly_trans: *r* =$_p$ *q* ⟹ *q* ≥$_p$ *p* ⟹ *r* ≥$_p$ *p* **and**
eq_gt_tpoly_trans: *r* =$_p$ *q* ⟹ *q* >$_p$ *p* ⟹ *r* >$_p$ *p* **and**
ge_eq_tpoly_trans: *r* ≥$_p$ *q* ⟹ *q* =$_p$ *p* ⟹ *r* ≥$_p$ *p* **and**
ge_ge_tpoly_trans: *r* ≥$_p$ *q* ⟹ *q* ≥$_p$ *p* ⟹ *r* ≥$_p$ *p* **and**
ge_gt_tpoly_trans: *r* ≥$_p$ *q* ⟹ *q* >$_p$ *p* ⟹ *r* >$_p$ *p* **and**

gt_eq_tpoly_trans: $r >_p q \Longrightarrow q =_p p \Longrightarrow r >_p p$ **and**
gt_ge_tpoly_trans: $r >_p q \Longrightarrow q \geq_p p \Longrightarrow r >_p p$ **and**
gt_gt_tpoly_trans: $r >_p q \Longrightarrow q >_p p \Longrightarrow r >_p p$
⟨*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* $\Longrightarrow$ *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*:
$f \in ground\_heads\ \zeta \Longrightarrow$
$wt\_sym\ f + \delta_h * arity\_sym_h\ f \geq wt\_sym\ (min\_ground\_head\ \zeta) + \delta_h * arity\_sym_h\ (min\_ground\_head\ \zeta)$
⟨*proof*⟩

lemma *min_ground_head_antimono*:
$ground\_heads\ \zeta \subseteq ground\_heads\ \xi \Longrightarrow$
$wt\_sym\ (min\_ground\_head\ \zeta) + \delta_h * arity\_sym_h\ (min\_ground\_head\ \zeta)$
$\geq wt\_sym\ (min\_ground\_head\ \xi) + \delta_h * arity\_sym_h\ (min\_ground\_head\ \xi)$
⟨*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 $\Longrightarrow$ *eval_ztpoly A* $(wt0\ \zeta) \geq zhmset\_of\ (wt\_sym\ (min\_ground\_head\ \zeta))$
⟨*proof*⟩

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

lemma *in_zip_imp_size_lt_apps*: $(s, y) \in set\ (zip\ ss\ ys) \Longrightarrow size\ s < size\ (apps\ (Hd\ \zeta)\ ss)$
⟨*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 (λ(s, i). PMult [coef_hd ζ i, wt s]) (zip ss [0..<length ss]))
  ⟨proof⟩
termination
  ⟨proof⟩

definition
  wt_args :: nat ⇒ ('v pvar ⇒ zhmultiset) ⇒ ('s, 'v) hd ⇒ ('s, 'v) tm list ⇒ zhmultiset
where
  wt_args i A ζ ss = sum_list
    (map (eval_ztpoly A ∘ (λ(s, i). PMult [coef_hd ζ i, wt s])) (zip ss [i..<i + length ss]))

lemma wt_Hd[simp]: wt (Hd ζ) = PSum [wt0 ζ, PNum (δ_h * arity_sym_h (min_ground_head ζ))]
  ⟨proof⟩

lemma coef_hd_cong:
  (∀ x ∈ vars_hd ζ. ∀ i. A (PCoef x i) = B (PCoef x i)) ⟹
   eval_ztpoly A (coef_hd ζ i) = eval_ztpoly B (coef_hd ζ i)
  ⟨proof⟩

lemma wt0_cong:
  assumes pwt_eq: ∀ x ∈ vars_hd ζ. A (PWt x) = B (PWt x)
  shows eval_ztpoly A (wt0 ζ) = eval_ztpoly B (wt0 ζ)
  ⟨proof⟩

lemma wt_cong:
  assumes
    ∀ x ∈ vars s. A (PWt x) = B (PWt x) and
    ∀ x ∈ vars s. ∀ 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 ⟹ eval_ztpoly A (wt s) = eval_ztpoly B (wt s)
  ⟨proof⟩

lemma exists_wt_sym:
  assumes legal: legal_zpassign A
  shows ∃ f ∈ ground_heads ζ. eval_ztpoly A (wt (Hd ζ)) ≥ zhmset_of (wt_sym f + δ_h * arity_sym_h f)
  ⟨proof⟩

lemma wt_ge_ε_h:
  assumes legal: legal_zpassign A
  shows eval_ztpoly A (wt s) ≥ zhmset_of ε_h
⟨proof⟩

lemma wt_args_ge_length_times_ε_h:
  assumes legal: legal_zpassign A
  shows wt_args i A ζ ss ≥ of_nat (length ss) * zhmset_of ε_h
  ⟨proof⟩

lemma wt_ge_δ_h: legal_zpassign A ⟹ eval_ztpoly A (wt s) ≥ zhmset_of δ_h
  ⟨proof⟩

lemma wt_gt_0: legal_zpassign A ⟹ eval_ztpoly A (wt s) > 0
  ⟨proof⟩

lemma wt_gt_δ_h_if_superunary:
  assumes
    legal: legal_zpassign A and
    superunary: arity_hd_h (head s) > 1
  shows eval_ztpoly A (wt s) > zhmset_of δ_h
⟨proof⟩

lemma wt_App_plus_δ_h_ge:
```

eval_ztpoly A (wt (App s t)) + zhmset_of δ_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) = zhmset_of* δ_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) = zhmset_of* δ_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) = zhmset_of* δ_h
shows $\delta_h = \varepsilon_h$
⟨*proof*⟩

lemma *wt_ge_vars*: *wt t* $\geq_p$ *wt s* $\Longrightarrow$ *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* $\Rightarrow$ ('*s*, '*v*) *tm* $\Rightarrow$ *bool* (**infix** $>_t$ *50*) **where**
gt_wt: *wt t* $>_p$ *wt s* $\Longrightarrow$ *t* $>_t$ *s*
| *gt_unary*: *wt t* $\geq_p$ *wt s* $\Longrightarrow$ $\neg$ *head t* $\leq\geq_{hd}$ *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*: *wt t* $\geq_p$ *wt s* $\Longrightarrow$ *head t* $>_{hd}$ *head s* $\Longrightarrow$ *t* $>_t$ *s*
| *gt_same*: *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))* $\Longrightarrow$ *t* $>_t$ *s*

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

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

inductive *gt_unary* :: ('*s*, '*v*) *tm* $\Rightarrow$ ('*s*, '*v*) *tm* $\Rightarrow$ *bool* **where**
gt_unaryI: *wt t* $\geq_p$ *wt s* $\Longrightarrow$ $\neg$ *head t* $\leq\geq_{hd}$ *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* $\geq_t$ *s* $\Longrightarrow$ *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*: $t >_t s \longleftrightarrow gt\_wt\ t\ s \vee gt\_unary\ t\ s \vee gt\_diff\ t\ s \vee gt\_same\ t\ s$
⟨*proof*⟩

lemma *gt_imp_wt*: $t >_t s \Longrightarrow wt\ t \geq_p wt\ s$
⟨*proof*⟩

lemma *gt_imp_vars*: $t >_t s \Longrightarrow vars\ t \supseteq vars\ s$
⟨*proof*⟩

6.4 Irreflexivity

theorem *gt_irefl*: $wary\ s \Longrightarrow \neg\ s >_t s$
⟨*proof*⟩

6.5 Transitivity

lemma *not_extf_gt_nil_singleton_if_*δ_h*_eq_*ε_h:
assumes *wary_s*: $wary\ s$ **and** δ_eq_ε: $\delta_h = \varepsilon_h$
shows $\neg\ extf\ f\ (>_t)\ []\ [s]$
⟨*proof*⟩

lemma *gt_sub_arg*: $wary\ (App\ s\ t) \Longrightarrow App\ s\ t >_t t$
⟨*proof*⟩

lemma *gt_arg*: $wary\ s \Longrightarrow is\_App\ s \Longrightarrow s >_t arg\ s$
⟨*proof*⟩

theorem *gt_trans*: $wary\ u \Longrightarrow wary\ t \Longrightarrow wary\ s \Longrightarrow u >_t t \Longrightarrow t >_t s \Longrightarrow u >_t s$
⟨*proof*⟩

lemma *gt_antisym*: $wary\ s \Longrightarrow wary\ t \Longrightarrow t >_t s \Longrightarrow \neg\ s >_t t$
⟨*proof*⟩

6.6 Subterm Property

lemma *gt_sub_fun*: $App\ s\ t >_t s$
⟨*proof*⟩

theorem *gt_proper_sub*: $wary\ t \Longrightarrow proper\_sub\ s\ t \Longrightarrow t >_t s$
⟨*proof*⟩

6.7 Compatibility with Functions

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

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

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* ⟹ *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* ⟹ *wary s* ⟹ $t >_t s$ ⟹ *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: $\bigwedge f.$ *ext_total* (*extf f*) **and**
gr_t: *ground t* **and**
gr_s: *ground s*
shows $t >_t s \vee s >_t t \vee t = s$
⟨*proof*⟩

6.11 Well-foundedness

abbreviation *gtw* :: $('s, 'v)\ tm \Rightarrow ('s, 'v)\ tm \Rightarrow bool$ (**infix** $>_{tw}$ *50*) **where**
$(>_{tw}) \equiv \lambda t\ s.\ wary\ t \wedge wary\ s \wedge t >_t s$

abbreviation *gtwg* :: $('s, 'v)\ tm \Rightarrow ('s, 'v)\ tm \Rightarrow bool$ (**infix** $>_{twg}$ *50*) **where**
$(>_{twg}) \equiv \lambda t\ s.\ ground\ t \wedge t >_{tw} s$

lemma *ground_gt_unary*:
  assumes *gr_t*: *ground t*
  shows ¬ *gt_unary t s*
⟨*proof*⟩

theorem *gt_wf*: *wfP* (λ*s t*. *t* >$_{tw}$ *s*)
⟨*proof*⟩

end

end

7 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*
begin

locale *simple_kbo_instances*
begin

definition *arity_sym* :: *nat* ⇒ *enat* **where**
  arity_sym n = ∞

definition *arity_var* :: *nat* ⇒ *enat* **where**
  arity_var n = ∞

definition *ground_head_var* :: *nat* ⇒ *nat set* **where**
  ground_head_var x = *UNIV*

definition *gt_sym* :: *nat* ⇒ *nat* ⇒ *bool* **where**
  gt_sym g f ⟷ *g* > *f*

definition ε :: *nat* **where**
  ε = *1*

definition δ :: *nat* **where**
  δ = *0*

definition *wt_sym* :: *nat* ⇒ *nat* **where**
  wt_sym n = *1*

definition *wt_sym$_h$* :: *nat* ⇒ *hmultiset* **where**
  wt_sym$_h$ n = *1*

definition *coef_sym$_h$* :: *nat* ⇒ *nat* ⇒ *hmultiset* **where**
  coef_sym$_h$ n i = *1*

sublocale *kbo_app*: *kbo_app gt_sym wt_sym* ε *len_lexext*
  ⟨*proof*⟩

sublocale *kbo_basic*: *kbo_basic gt_sym wt_sym* ε λ*f*. *len_lexext ground_head_var*
  ⟨*proof*⟩

sublocale *kbo_std*: *kbo_std ground_head_var gt_sym* ε δ λ*f*. *len_lexext arity_sym arity_var wt_sym*
  ⟨*proof*⟩

sublocale *tkbo_coefs*: *tkbo_coefs ground_head_var gt_sym* ε δ λ*f*. *len_lexext arity_sym arity_var*
    wt_sym$_h$ coef_sym$_h$
  ⟨*proof*⟩

end

end