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

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

October 21, 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* :: $'s \Rightarrow 's \Rightarrow$ *bool* (**infix** $>_s$ *50*) +
 fixes
  wt_sym :: $'s \Rightarrow$ *nat* **and**
  ε :: *nat* **and**
  ground_heads_var :: $'v \Rightarrow 's$ *set* **and**
  extf :: $'s \Rightarrow (('s, 'v)$ *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**
  ground_heads_var_nonempty: *ground_heads_var x* $\neq \{\}$ **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 :: $'s \Rightarrow 's \Rightarrow$ *bool* (**infix** $>_s$ *50*) **and**
   arity_sym :: $'s \Rightarrow$ *enat* **and**
   arity_var :: $'v \Rightarrow$ *enat* +
 fixes
  wt_sym :: $'s \Rightarrow 'n$::{*ord,semiring_1*} **and**
  ε :: *nat* **and**
  δ :: *nat* **and**
  extf :: $'s \Rightarrow (('s, 'v)$ *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**
  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* $\geq$ *of_nat* (ε − *the_enat* (*of_nat* δ ∗ *arity_sym f*)) **and**
  unary_wt_sym_0_gt: *arity_sym f* $= 1 \Longrightarrow$ *wt_sym f* $= 0 \Longrightarrow f >_s g \lor g = f$ **and**
  unary_wt_sym_0_imp_δ_eq_ε: *arity_sym f* $= 1 \Longrightarrow$ *wt_sym f* $= 0 \Longrightarrow \delta = \varepsilon$ **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*: $\delta > 0 \Longrightarrow$ *arity_sym f* $\neq \infty$
⟨*proof*⟩

lemma *arity_var_ne_infinity_if_δ_gt_0*: $\delta > 0 \Longrightarrow$ *arity_var x* $\neq \infty$
⟨*proof*⟩

lemma *arity_ne_infinity_if_δ_gt_0*: $\delta > 0 \Longrightarrow$ *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*: $\delta = \varepsilon \Longrightarrow$ *extf f 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* ⇒ *'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: $\varepsilon > 0$ **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** $\geq_t$ *50*) **where**
 t $\geq_t$ *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*
abbrevs >*t* = $>_t$
 and ≥*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* ⇒ *enat* **and**
  arity_var :: *'v* ⇒ *enat* **and**
  wt_sym :: *'s* ⇒ *nat*
begin

4.2 Weights

primrec *wt* :: (*'s*, *'v*) *tm* ⇒ *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: $wt\ (Hd\ (Sym\ f)) = wt\_sym\ f + the\_enat\ (\delta * arity\_sym\ f)$
⟨*proof*⟩

lemma $exists\_wt\_sym$: $\exists f \in ground\_heads\ \zeta.\ wt\ (Hd\ \zeta) = wt\_sym\ f + the\_enat\ (\delta * arity\_sym\ f)$
⟨*proof*⟩

lemma $wt\_le\_wt\_sym$: $f \in ground\_heads\ \zeta \Longrightarrow wt\ (Hd\ \zeta) \le wt\_sym\ f + the\_enat\ (\delta * arity\_sym\ f)$
⟨*proof*⟩

lemma $enat\_the\_enat\_\delta\_times\_arity\_sym[simp]$: $enat\ (the\_enat\ (\delta * arity\_sym\ f)) = \delta * arity\_sym\ f$
⟨*proof*⟩

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

lemma wt_ge_ε: $wt\ s \ge \varepsilon$
⟨*proof*⟩

lemma wt_ge_δ: $wt\ s \ge \delta$
⟨*proof*⟩

lemma $wt\_gt\_\delta\_if\_superunary$: $arity\_hd\ (head\ s) > 1 \Longrightarrow wt\ s > \delta$
⟨*proof*⟩

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

lemma $wt\_App\_ge\_fun$: $wt\ (App\ s\ t) \ge wt\ s$
⟨*proof*⟩

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

lemma $wt\_\delta\_imp\_\delta\_eq\_\varepsilon$: $wt\ s = \delta \Longrightarrow \delta = \varepsilon$
⟨*proof*⟩

lemma $wt\_ge\_arity\_head\_if\_\delta\_gt\_0$:
assumes δ_gt_0: $\delta > 0$
shows $wt\ s \ge arity\_hd\ (head\ s)$
⟨*proof*⟩

lemma $wt\_ge\_num\_args\_if\_\delta\_eq\_0$:
assumes δ_eq_0: $\delta = 0$
shows $wt\ s \ge num\_args\ s$
⟨*proof*⟩

lemma $wt\_ge\_num\_args$: $wary\ s \Longrightarrow wt\ s \ge num\_args\ s$
⟨*proof*⟩

4.3 Inductive Definitions

inductive $gt :: ('s, 'v)\ tm \Rightarrow ('s, 'v)\ tm \Rightarrow bool$ (**infix** $>_t$ 50) **where**
gt_wt: $vars\_mset\ t \supseteq\#\ vars\_mset\ s \Longrightarrow wt\ t > wt\ s \Longrightarrow t >_t s$
$|\ gt\_unary$: $wt\ t = wt\ s \Longrightarrow \neg\ head\ t \le\ge_{hd} head\ s \Longrightarrow num\_args\ t = 1 \Longrightarrow$
$(\exists f \in ground\_heads\ (head\ t).\ arity\_sym\ f = 1 \wedge wt\_sym\ f = 0) \Longrightarrow arg\ t >_t s \vee arg\ t = s \Longrightarrow$
$t >_t s$
$|\ gt\_diff$: $vars\_mset\ t \supseteq\#\ vars\_mset\ s \Longrightarrow wt\ t = wt\ s \Longrightarrow head\ t >_{hd} head\ s \Longrightarrow t >_t s$
$|\ gt\_same$: $vars\_mset\ t \supseteq\#\ vars\_mset\ s \Longrightarrow wt\ t = wt\ s \Longrightarrow head\ t = head\ s \Longrightarrow$
$(\forall f \in ground\_heads\ (head\ t).\ extf\ f\ (>_t)\ (args\ t)\ (args\ s)) \Longrightarrow t >_t s$

abbreviation $ge :: ('s, 'v)\ tm \Rightarrow ('s, 'v)\ tm \Rightarrow bool$ (**infix** $\ge_t$ 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: *vars_mset t* $\supseteq\#$ *vars_mset s* $\Longrightarrow$ *wt t* $>$ *wt s* $\Longrightarrow$ *gt_wt t s*

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

inductive *gt_unary* :: $('s, 'v)$ *tm* $\Rightarrow$ $('s, 'v)$ *tm* $\Rightarrow$ *bool* **where**
gt_unaryI: *wt t* $=$ *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_same* :: $('s, 'v)$ *tm* $\Rightarrow$ $('s, 'v)$ *tm* $\Rightarrow$ *bool* **where**
gt_sameI: *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*)) $\Longrightarrow$ *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_vars_mset*: $t >_t s \Longrightarrow$ *vars_mset t* $\supseteq\#$ *vars_mset s*
⟨*proof*⟩

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

4.4 Irreflexivity

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

4.5 Transitivity

lemma *gt_imp_wt_ge*: $t >_t s \Longrightarrow$ *wt t* $\geq$ *wt s*
⟨*proof*⟩

lemma *not_extf_gt_nil_singleton_if_δ_eq_ε*:
assumes *wary_s*: *wary s* **and** *δ_eq_ε*: $\delta = \varepsilon$
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*⟩

4.6 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*⟩

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 $wary\_\varrho$: $wary\_subst\ \varrho$ **and** $wary\_s$: $wary\ s$
shows $wt\ (subst\ \varrho\ s) = wt\ s + extra\_wt\ \varrho\ s$
⟨*proof*⟩

theorem gt_subst:
assumes $wary\_\varrho$: $wary\_subst\ \varrho$
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 $t >_t s \lor s >_t t \lor t = s$
⟨*proof*⟩

4.11 Well-foundedness

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

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

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* _ λ_. ∞ λ_. ∞
  ⟨*proof*⟩

fun *wt* :: $('s, 'v)$ *tm* $\Rightarrow$ *nat* **where**
  wt (*Hd* ζ) = (*LEAST w*. $\exists f \in$ *ground_heads* ζ. *w* = *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_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: (λ*t s*. *t* $>_t$ *s* $\wedge$ *local.kbo_std.gt t s*) $\leq$ *kbo_std.gt* **and**
  gt_and_basic_gt_le_basic_gt: (λ*t s*. *local.kbo_std.gt t s* $\wedge$ *t* $>_t$ *s*) $\leq$ ($>_t$)
  ⟨*proof*⟩

lemma *basic_gt_iff_lt*: *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*⟩

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**
$\delta_h \equiv of\_nat\ \delta$

abbreviation ε_h :: *hmultiset* **where**
$\varepsilon_h \equiv of\_nat\ \varepsilon$

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** $\geq_p$ *50*)
where
q $\geq_p$ *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* $\geq_p$ *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*: ¬ *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* $\geq_p$ *p* ⟹ *r* $\geq_p$ *p* **and**
eq_gt_tpoly_trans: *r* $=_p$ *q* ⟹ *q* $>_p$ *p* ⟹ *r* $>_p$ *p* **and**
ge_eq_tpoly_trans: *r* $\geq_p$ *q* ⟹ *q* $=_p$ *p* ⟹ *r* $\geq_p$ *p* **and**
ge_ge_tpoly_trans: *r* $\geq_p$ *q* ⟹ *q* $\geq_p$ *p* ⟹ *r* $\geq_p$ *p* **and**
ge_gt_tpoly_trans: *r* $\geq_p$ *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* ⇒ *nat* ⇒ ('*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* ζ *i*) > *0*
⟨*proof*⟩

primrec *coef* :: ('*s*, '*v*) *tm* ⇒ *nat* ⇒ ('*v pvar*, *hmultiset*) *tpoly* **where**
coef (*Hd* ζ) *i* = *coef_hd* ζ *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* ⟹ *eval_ztpoly A* (*coef s i*) > *0*
⟨*proof*⟩

lemma *exists_min_ground_head*:
∃*f*. *f* ∈ *ground_heads* ζ ∧
(∀ *g* ∈ *ground_heads* ζ. *wt_sym g* + δ_h ∗ *arity_sym*$_h$ *g* ≥ *wt_sym f* + δ_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* ζ ∈ *ground_heads* ζ
⟨*proof*⟩

lemma *min_ground_head_min*:
f ∈ *ground_heads* ζ ⟹
wt_sym f + δ_h ∗ *arity_sym*$_h$ *f* ≥ *wt_sym* (*min_ground_head* ζ) + δ_h ∗ *arity_sym*$_h$ (*min_ground_head* ζ)
⟨*proof*⟩

lemma *min_ground_head_antimono*:
ground_heads ζ ⊆ *ground_heads* ξ ⟹
wt_sym (*min_ground_head* ζ) + δ_h ∗ *arity_sym*$_h$ (*min_ground_head* ζ)
≥ *wt_sym* (*min_ground_head* ξ) + δ_h ∗ *arity_sym*$_h$ (*min_ground_head* ξ)
⟨*proof*⟩

primrec *wt0* :: ('*s*, '*v*) *hd* ⇒ ('*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 ⟹ *eval_ztpoly A* (*wt0* ζ) ≥ *zhmset_of* (*wt_sym* (*min_ground_head* ζ))
⟨*proof*⟩

lemma *eval_ztpoly_nonneg*: *legal_zpassign A* ⟹ *eval_ztpoly A p* ≥ *0*
⟨*proof*⟩

lemma *in_zip_imp_size_lt_apps*: (*s*, *y*) ∈ *set* (*zip ss ys*) ⟹ *size s* < *size* (*apps* (*Hd* ζ) *ss*)
⟨*proof*⟩

function *wt* :: ('*s*, '*v*) *tm* ⇒ ('*v pvar*, *hmultiset*) *tpoly* **where**
wt (*apps* (*Hd* ζ) *ss*) =
PSum ([*wt0* ζ, *PNum* (δ_h ∗ (*arity_sym*$_h$ (*min_ground_head* ζ) − *of_nat* (*length ss*)))] @

$map\ (\lambda(s,\ i).\ PMult\ [coef\_hd\ \zeta\ i,\ wt\ s])\ (zip\ ss\ [0..<length\ ss]))$
⟨proof⟩
termination
⟨proof⟩

definition
$wt\_args :: nat \Rightarrow ('v\ pvar \Rightarrow zhmultiset) \Rightarrow ('s,\ 'v)\ hd \Rightarrow ('s,\ 'v)\ tm\ list \Rightarrow zhmultiset$
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 $coef\_hd\_cong$:
$(\forall x \in vars\_hd\ \zeta.\ \forall i.\ A\ (PCoef\ x\ i) = B\ (PCoef\ x\ i)) \Longrightarrow$
$eval\_ztpoly\ A\ (coef\_hd\ \zeta\ i) = eval\_ztpoly\ B\ (coef\_hd\ \zeta\ i)$
⟨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 $ground\_eval\_ztpoly\_wt\_eq$: $ground\ s \Longrightarrow eval\_ztpoly\ A\ (wt\ s) = eval\_ztpoly\ B\ (wt\ s)$
⟨proof⟩

lemma $exists\_wt\_sym$:
assumes $legal$: $legal\_zpassign\ A$
shows $\exists f \in ground\_heads\ \zeta.\ eval\_ztpoly\ A\ (wt\ (Hd\ \zeta)) \geq zhmset\_of\ (wt\_sym\ f + \delta_h * arity\_sym_h\ f)$
⟨proof⟩

lemma $wt\_ge\_\varepsilon_h$:
assumes $legal$: $legal\_zpassign\ A$
shows $eval\_ztpoly\ A\ (wt\ s) \geq zhmset\_of\ \varepsilon_h$
⟨proof⟩

lemma $wt\_args\_ge\_length\_times\_\varepsilon_h$:
assumes $legal$: $legal\_zpassign\ A$
shows $wt\_args\ i\ A\ \zeta\ ss \geq of\_nat\ (length\ ss) * zhmset\_of\ \varepsilon_h$
⟨proof⟩

lemma $wt\_ge\_\delta_h$: $legal\_zpassign\ A \Longrightarrow eval\_ztpoly\ A\ (wt\ s) \geq zhmset\_of\ \delta_h$
⟨proof⟩

lemma wt_gt_0: $legal\_zpassign\ A \Longrightarrow eval\_ztpoly\ A\ (wt\ s) > 0$
⟨proof⟩

lemma $wt\_gt\_\delta_h\_if\_superunary$:
assumes
$legal$: $legal\_zpassign\ A$ **and**
$superunary$: $arity\_hd_h\ (head\ s) > 1$
shows $eval\_ztpoly\ A\ (wt\ s) > zhmset\_of\ \delta_h$
⟨proof⟩

lemma $wt\_App\_plus\_\delta_h\_ge$:

$eval\_ztpoly\ A\ (wt\ (App\ s\ t)) + zhmset\_of\ \delta_h$
$\geq eval\_ztpoly\ A\ (wt\ s) + eval\_ztpoly\ A\ (coef\ s\ 0) * eval\_ztpoly\ A\ (wt\ t)$
⟨*proof*⟩

lemma $wt\_App\_fun\_\delta_h$:
assumes
$legal$: $legal\_zpassign\ A$ **and**
wt_st: $eval\_ztpoly\ A\ (wt\ (App\ s\ t)) = eval\_ztpoly\ A\ (wt\ t)$
shows $eval\_ztpoly\ A\ (wt\ s) = zhmset\_of\ \delta_h$
⟨*proof*⟩

lemma $wt\_App\_arg\_\delta_h$:
assumes
$legal$: $legal\_zpassign\ A$ **and**
wt_st: $eval\_ztpoly\ A\ (wt\ (App\ s\ t)) = eval\_ztpoly\ A\ (wt\ s)$
shows $eval\_ztpoly\ A\ (wt\ t) = zhmset\_of\ \delta_h$
⟨*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 $wt\_\delta_h\_imp\_\delta_h\_eq\_\varepsilon_h$:
assumes
$legal$: $legal\_zpassign\ A$ **and**
$wt\_s\_eq\_\delta$: $eval\_ztpoly\ A\ (wt\ s) = zhmset\_of\ \delta_h$
shows $\delta_h = \varepsilon_h$
⟨*proof*⟩

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

lemma $sum\_coefs\_ge\_num\_args\_if\_\delta_h\_eq\_0$:
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 $gt\_iff\_wt\_unary\_diff\_same$: $t >_t s \longleftrightarrow gt\_wt\ t\ s \lor gt\_unary\ t\ s \lor gt\_diff\ t\ s \lor gt\_same\ t\ s$
⟨*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_irrefl: $wary\ s \Longrightarrow \neg\ s >_t s$
⟨*proof*⟩

6.5 Transitivity

lemma $not\_extf\_gt\_nil\_singleton\_if\_\delta_h\_eq\_\varepsilon_h$:
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 $\varrho\ A\ (PWt\ x) =$
   eval_ztpoly $A\ (wt\ (\varrho\ x)) -$ *zhmset_of* $(\delta_h *$ *arity_sym*$_h$ (*min_ground_head* $(Var\ x)))$
| *subst_zpassign* $\varrho\ A\ (PCoef\ x\ i) =$ *eval_ztpoly* A $(coef\ (\varrho\ x)\ i)$

lemma *legal_subst_zpassign*:
  assumes
    legal: *legal_zpassign* A **and**
    *wary_*ϱ: *wary_subst* ϱ
  shows *legal_zpassign* (*subst_zpassign* $\varrho\ A$)
  ⟨*proof*⟩

lemma *wt_subst*:
  assumes
    legal: *legal_zpassign* A **and**
    *wary_*ϱ: *wary_subst* ϱ
  shows *wary* $s \Longrightarrow$ *eval_ztpoly* A $(wt\ (subst\ \varrho\ s)) =$ *eval_ztpoly* (*subst_zpassign* $\varrho\ A$) $(wt\ s)$
⟨*proof*⟩

theorem *gt_subst*:
  assumes *wary_*ϱ: *wary_subst* ϱ
  shows *wary* $t \Longrightarrow$ *wary* $s \Longrightarrow t >_t s \Longrightarrow subst\ \varrho\ t >_t subst\ \varrho\ s$
⟨*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* $(\lambda s\ t.\ t >_{tw} s)$
⟨*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 $\lambda f.$ *len_lexext ground_head_var*
  ⟨*proof*⟩

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

sublocale *tkbo_coefs*: *tkbo_coefs ground_head_var gt_sym* ε δ $\lambda f.$ *len_lexext arity_sym arity_var*
    wt_sym_h $coef\_sym_h$
  ⟨*proof*⟩

end

end