The Akra–Bazzi theorem and the Master theorem

Manuel Eberl

April 17, 2016

Abstract

This article contains a formalisation of the Akra–Bazzi method [1] based on a proof by Leighton [2]. It is a generalisation of the well-known Master Theorem for analysing the complexity of Divide & Conquer algorithms. We also include a generalised version of the Master theorem based on the Akra–Bazzi theorem, which is easier to apply than the Akra–Bazzi theorem itself.

Some proof methods that facilitate applying the Master theorem are also included. For a more detailed explanation of the formalisation and the proof methods, see the accompanying paper (publication forthcoming).

Contents

1 Auxiliary lemmas

theory *Akra-Bazzi-Library*
imports
  Complex-Main
  ../Landau-Symbols/Landau-Symbols
begin

declare *DERIV-powr*[*THEN DERIV-chain2*, *derivative-intros*]

lemma *setsum-pos'*:
  assumes *finite I*
  assumes $\exists x{\in}I.\ f\ x > (0 :: \text{-} :: \mathit{linordered\text{-}ab\text{-}group\text{-}add})$
  assumes $\bigwedge x.\ x \in I \Longrightarrow f\ x \geq 0$
  shows *setsum f I* > 0
proof−
  from *assms*(*2*) **guess** *x* **by** (*elim bexE*) **note** *x* = *this*
  from *x* **have** *I* = *insert x I* **by** *blast*
  also from *assms*(*1*) **have** *setsum f* ... = *f x* + *setsum f* (*I* − {*x*}) **by** (*rule setsum.insert-remove*)
  also from *x assms* **have** ... > *0* **by** (*intro add-pos-nonneg setsum-nonneg*) *simp-all*
  finally show *?thesis* **.**
qed

lemma *min-mult-left*:
  assumes (*x*::*real*) > *0*
  shows *x* ∗ *min y z* = *min* (*x*∗*y*) (*x*∗*z*)
  using *assms* **by** (*auto simp add*: *min-def algebra-simps*)

lemma *max-mult-left*:
  assumes (*x*::*real*) > *0*
  shows *x* ∗ *max y z* = *max* (*x*∗*y*) (*x*∗*z*)
  using *assms* **by** (*auto simp add*: *max-def algebra-simps*)

lemma *DERIV-nonneg-imp-mono*:
  assumes $\bigwedge t.\ t{\in}\{x..y\} \Longrightarrow$ (*f has-field-derivative f' t*) (*at t*)
  assumes $\bigwedge t.\ t{\in}\{x..y\} \Longrightarrow f'\ t \geq 0$
  assumes (*x*::*real*) ≤ *y*
  shows (*f x* :: *real*) ≤ *f y*
proof (*cases x y rule*: *linorder-cases*)

assume *xy*: $x < y$
hence $\exists z.\ x < z \wedge z < y \wedge f\, y - f\, x = (y - x) * f'\, z$
by (*rule MVT2*) (*insert assms*(*1*), *simp*)
then guess z **by** (*elim exE conjE*) **note** $z = this$
from *z*(*1*,*2*) *assms*(*2*) *xy* **have** $0 \le (y - x) * f'\, z$ **by** (*intro mult-nonneg-nonneg*) *simp-all*
also note *z*(*3*)[*symmetric*]
finally show $f\, x \le f\, y$ **by** *simp*
qed (*insert assms*(*3*), *simp-all*)

lemma *eventually-conjE*: *eventually* $(\lambda x.\ P\ x \wedge Q\ x)$ $F \Longrightarrow$ (*eventually* $P\ F \Longrightarrow$ *eventually* $Q\ F \Longrightarrow R) \Longrightarrow R$
apply (*frule eventually-rev-mp*[*of - - P*], *simp*)
apply (*drule eventually-rev-mp*[*of - - Q*], *simp*)
apply *assumption*
done

lemma *real-natfloor-nat*: $x \in \mathbb{N} \Longrightarrow$ *real* (*nat* $\lfloor x \rfloor$) $= x$ **by** (*elim Nats-cases*) *simp*

lemma *eventually-natfloor*:
assumes *eventually* P (*at-top* :: *nat filter*)
shows *eventually* $(\lambda x.\ P$ (*nat* $\lfloor x \rfloor$)) (*at-top* :: *real filter*)
proof−
from *assms* **obtain** N **where** N: $\bigwedge n.\ n \ge N \Longrightarrow P\ n$ **using** *eventually-at-top-linorder* **by** *blast*
have $\forall n \ge$ *real* $N.\ P$ (*nat* $\lfloor n \rfloor$) **by** (*intro allI impI N le-nat-floor*) *simp-all*
thus *?thesis* **using** *eventually-at-top-linorder* **by** *blast*
qed

lemma *tendsto-0-smallo-1*: $f \in o(\lambda x.\ 1$ :: *real*) $\Longrightarrow$ $(f \longrightarrow 0)$ *at-top*
by (*drule smalloD-tendsto*) *simp*

lemma *smallo-1-tendsto-0*: $(f \longrightarrow 0)$ *at-top* $\Longrightarrow f \in o(\lambda x.\ 1$ :: *real*)
by (*rule smalloI-tendsto*) *simp-all*

lemma *filterlim-at-top-smallomega-1*:
$f \in \omega(\lambda x.\ 1$ :: *real*) $\Longrightarrow$ *eventually* $(\lambda x.\ f\ x > 0)$ *at-top* $\Longrightarrow$ *filterlim f at-top at-top*
apply (*drule smallomegaD-filterlim-at-top*, *simp*)
apply (*subst filterlim-cong*[*OF refl refl*, *of* - $\lambda x.$ *abs* $(f\ x)$])
apply (*auto elim*!: *eventually-mono*)
done

lemma *smallo-imp-abs-less*:
assumes $f \in o(g)$ *eventually* $(\lambda x.\ g\ x > (0$::'*a*::*linordered-field*)) *at-top*
shows *eventually* $(\lambda x.\ |f\ x| < g\ x)$ *at-top*
proof −
have $1/2 > (0$::'*a*) **by** *simp*
from *landau-o.smallD*[*OF assms*(*1*) *this*] *assms*(*2*) **show** *?thesis*

by *eventually-elim auto*
qed

lemma *smallo-imp-less*:
assumes $f \in o(g)$ *eventually* $(\lambda x.\ g\ x > 0)$ *at-top*
shows *eventually* $(\lambda x.\ f\ x < g\ x)$ *at-top*
using *smallo-imp-abs-less*[*OF assms*] **by** *eventually-elim simp*

lemma *smallo-imp-le*:
assumes $f \in o(g)$ *eventually* $(\lambda x.\ g\ x \geq 0)$ *at-top*
shows *eventually* $(\lambda x.\ f\ x \leq g\ x)$ *at-top*
using *landau-o.smallD*[*OF assms*(*1*) *zero-less-one*] *assms*(*2*) **by** *eventually-elim simp*

lemma *filterlim-at-right*:
filterlim f (*at-right* a) $F \longleftrightarrow$ *eventually* $(\lambda x.\ f\ x > a)$ $F \wedge$ *filterlim* f (*nhds* a) F
by (*subst filterlim-at*) (*auto elim*!: *eventually-mono*)

lemma *one-plus-x-powr-approx-ex*:
assumes x: *abs* $(x{::}real) \leq 1/2$
obtains t **where** *abs* $t < 1/2$ $(1 + x)$ *powr* $p =$
$1 + p * x + p * (p - 1) * (1 + t)$ *powr* $(p - 2)\ /\ 2 * x\ \hat{}\ 2$
proof (*cases* $x = 0$)
assume x': $x \neq 0$
let $?f = \lambda x.\ (1 + x)$ *powr* p
let $?f' = \lambda x.\ p * (1 + x)$ *powr* $(p - 1)$
let $?f'' = \lambda x.\ p * (p - 1) * (1 + x)$ *powr* $(p - 2)$
let $?fs = op!\ [?f,\ ?f',\ ?f'']$

have A: $\forall\, m\ t.\ m < 2 \wedge t \geq -0.5 \wedge t \leq 0.5 \longrightarrow$ ($?fs\ m$ *has-real-derivative* $?fs$ (*Suc* m) t) (*at* t)
proof (*clarify*)
fix m :: *nat* **and** t :: *real* **assume** m: $m < 2$ **and** t: $t \geq -0.5$ $t \leq 0.5$
thus ($?fs\ m$ *has-real-derivative* $?fs$ (*Suc* m) t) (*at* t)
using m **by** (*cases* m) (*force intro*: *derivative-eq-intros algebra-simps*)+
qed
have $\exists\, t.$ (*if* $x < 0$ *then* $x < t \wedge t < 0$ *else* $0 < t \wedge t < x$) $\wedge$
$(1 + x)$ *powr* $p = (\sum m{<}2.\ ?fs\ m\ 0\ /\ (fact\ m) * (x - 0)\hat{}m) +$
$?fs\ 2\ t\ /\ (fact\ 2) * (x - 0)^2$
using *assms* x' **by** (*intro taylor*[*OF - - A*]) *simp-all*
then guess t **by** (*elim exE conjE*)
note $t = this$
with *assms* **have** *abs* $t < 1/2$ **by** (*auto split*: *if-split-asm*)
moreover from $t(2)$ **have** $(1 + x)$ *powr* $p = 1 + p * x + p * (p - 1) * (1 + t)$ *powr* $(p - 2)\ /\ 2 * x\ \hat{}\ 2$
by (*simp add*: *numeral-2-eq-2 of-nat-Suc*)
ultimately show *?thesis* **by** (*rule that*)

next
  assume $x = 0$
  with *that*[*of 0*] **show** *?thesis* **by** *simp*
qed

lemma *one-plus-x-powr-taylor2*:
  obtains k **where** $\bigwedge x.$ *abs* $(x$::*real*$) \leq 1/2 \Longrightarrow$ *abs* $((1 + x)$ *powr* $p - 1 - p{*}x)$ $\leq k{*}x\hat{\ }2$
proof−
  def $k \equiv |p{*}(p - 1)| * max\ ((1/2)\ powr\ (p - 2))\ ((3/2)\ powr\ (p - 2))\ /\ 2$
  show *?thesis*
  proof (*rule that*[*of k*])
    fix x :: *real* **assume** *abs* $x \leq 1/2$
    from *one-plus-x-powr-approx-ex*[*OF this*, *of p*] **guess** t **.** **note** $t = this$
    from t **have** *abs* $((1 + x)$ *powr* $p - 1 - p{*}x) = |p{*}(p - 1)| * (1 + t)$ *powr* $(p - 2)/2 * x^2$
      by (*simp add*: *abs-mult*)
    also from $t(1)$ **have** $(1 + t)$ *powr* $(p - 2) \leq max\ ((1/2)\ powr\ (p - 2))\ ((3/2)\ powr\ (p - 2))$
      by (*intro powr-upper-bound*) *simp-all*
    finally show *abs* $((1 + x)$ *powr* $p - 1 - p{*}x) \leq k{*}x\hat{\ }2$
      by (*simp add*: *mult-left-mono mult-right-mono k-def*)
  qed
qed

lemma *one-plus-x-powr-taylor2-bigo*:
  assumes *lim*: $(f \longrightarrow 0)$ *at-top*
  shows $(\lambda x.\ (1 + f\,x)$ *powr* $(p$::*real*$) - 1 - p * f\,x) \in O(\lambda x.\ f\,x\ \hat{\ }\ 2)$
proof −
  from *one-plus-x-powr-taylor2*[*of p*] **guess** k **.**
  moreover from *tendstoD*[*OF lim*, *of 1/2*]
    have *eventually* $(\lambda x.$ *abs* $(f\,x) < 1/2)$ *at-top* **by** (*simp add*: *dist-real-def*)
  ultimately have *eventually* $(\lambda x.$ *abs* $((1 + f\,x)$ *powr* $p - 1 - p * f\,x) \leq k *$ *abs* $(f\,x\ \hat{\ }\ 2))$ *at-top*
    by (*auto elim*!: *eventually-mono*)
  thus *?thesis* **by** (*rule bigoI*)
qed

lemma *one-plus-x-powr-taylor1-bigo*:
  assumes *lim*: $(f \longrightarrow 0)$ *at-top*
  shows $(\lambda x.\ (1 + f\,x)$ *powr* $(p$::*real*$) - 1) \in O(\lambda x.\ f\,x)$
proof −
  from *assms* **have** $(\lambda x.\ (1 + f\,x)$ *powr* $p - 1 - p * f\,x) \in O(\lambda x.\ (f\,x)^2)$
    by (*rule one-plus-x-powr-taylor2-bigo*)
  also from *assms* **have** $f \in O(\lambda$-$.\ 1)$ **by** (*intro bigoI-tendsto*) *simp-all*
  from *landau-o.big.mult*[*of f f*, *OF - this*] **have** $(\lambda x.\ (f\,x)\hat{\ }2) \in O(\lambda x.\ f\,x)$
    by (*simp add*: *power2-eq-square*)
  finally have A: $(\lambda x.\ (1 + f\,x)$ *powr* $p - 1 - p * f\,x) \in O(f)$ **.**
  have B: $(\lambda x.\ p * f\,x) \in O(f)$ **by** *simp*

from *sum-in-bigo*(*1*)[*OF A B*] **show** *?thesis* **by** *simp*
qed

lemma *x-times-x-minus-1-nonneg*: $x \le 0 \lor x \ge 1 \Longrightarrow (x{::}\text{-}{::}linordered\text{-}idom) * (x - 1) \ge 0$
proof (*elim disjE*)
  assume *x*: $x \le 0$
  also have $0 \le x\hat{\ }2$ **by** *simp*
  finally show $x * (x - 1) \ge 0$ **by** (*simp add*: *power2-eq-square algebra-simps*)
qed *simp*

lemma *x-times-x-minus-1-nonpos*: $x \ge 0 \Longrightarrow x \le 1 \Longrightarrow (x{::}\text{-}{::}linordered\text{-}idom) * (x - 1) \le 0$
  by (*intro mult-nonneg-nonpos*) *simp-all*

end

2 Asymptotic bounds

theory *Akra-Bazzi-Asymptotics*
imports
  Complex-Main
  Akra-Bazzi-Library
  ../Landau-Symbols/Landau-Symbols
begin

locale *akra-bazzi-asymptotics-bep* =
  fixes *b e p hb* :: *real*
  assumes *bep*: $b > 0$ $b < 1$ $e > 0$ $hb > 0$
begin

context
begin

Functions that are negligible w.r.t. $ln\ (b * x)\ powr\ (e\ /\ 2 + 1)$.

private abbreviation (*input*) *negl* :: (*real* ⇒ *real*) ⇒ *bool* **where**
  $negl\ f \equiv f \in o(\lambda x.\ ln\ (b{*}x)\ powr\ (-(e/2 + 1)))$

private lemma *neglD*: $negl\ f \Longrightarrow c > 0 \Longrightarrow eventually\ (\lambda x.\ |f\ x| \le c\ /\ ln\ (b{*}x)\ powr\ (e/2{+}1))\ at\text{-}top$
  by (*drule* (*1*) *landau-o.smallD*, *subst* (*asm*) *powr-minus*) (*simp add*: *field-simps*)

private lemma *negl-mult*: $negl\ f \Longrightarrow negl\ g \Longrightarrow negl\ (\lambda x.\ f\ x * g\ x)$
  by (*erule landau-o.small-1-mult*, *rule landau-o.small-imp-big*, *erule landau-o.small-trans*)
    (*insert bep*, *simp*)

private lemma *ev4*:
  assumes *g*: *negl g*
  shows $eventually\ (\lambda x.\ ln\ (b{*}x)\ powr\ (-e/2) - ln\ x\ powr\ (-e/2) \ge g\ x)\ at\text{-}top$

proof (*rule smallo-imp-le*)
def *h1* ≡ (λ*x*. (*1* + *ln b*/*ln x*) *powr* (−*e*/*2*) − *1* + *e*/*2* ∗ (*ln b*/*ln x*))
def *h2* ≡ λ*x*. *ln x powr* (− *e* / *2*) ∗ ((*1* + *ln b* / *ln x*) *powr* (− *e* / *2*) − *1*)
from *bep* **have** ((λ*x*. *ln b* / *ln x*) ⟶ *0*) *at-top*
by (*simp add*: *tendsto-0-smallo-1*)
note *one-plus-x-powr-taylor2-bigo*[*OF this*, *of* −*e*/*2*]
also have (λ*x*. (*1* + *ln b* / *ln x*) *powr* (− *e* / *2*) − *1* − − *e* / *2* ∗ (*ln b* / *ln x*)) = *h1*
by (*simp add*: *h1-def*)
finally have *h1* ∈ *o*(λ*x*. *1* / *ln x*)
by (*rule landau-o.big-small-trans*) (*insert bep*, *simp add*: *power2-eq-square*)
with *bep* **have** (λ*x*. *h1 x* − *e*/*2* ∗ (*ln b* / *ln x*)) ∈ Θ(λ*x*. *1* / *ln x*) **by** *simp*
also have (λ*x*. *h1 x* − *e*/*2* ∗ (*ln b*/*ln x*)) = (λ*x*. (*1* + *ln b*/ *ln x*) *powr* (−*e*/*2*) − *1*)
by (*rule ext*) (*simp add*: *h1-def*)
finally have *h2* ∈ Θ(λ*x*. *ln x powr* (−*e*/*2*) ∗ (*1* / *ln x*)) **unfolding** *h2-def*
by (*intro landau-theta.mult*) *simp-all*
also have (λ*x*. *ln x powr* (−*e*/*2*) ∗ (*1* / *ln x*)) ∈ Θ(λ*x*. *ln x powr* (−(*e*/*2*+*1*))) **by** *simp*
also from *g bep* **have** (λ*x*. *ln x powr* (−(*e*/*2*+*1*))) ∈ ω(*g*) **by** (*simp add*: *smallomega-iff-smallo*)
finally have *g* ∈ *o*(*h2*) **by** (*simp add*: *smallomega-iff-smallo*)
also have *eventually* (λ*x*. *h2 x* = *ln* (*b*∗*x*) *powr* (−*e*/*2*) − *ln x powr* (−*e*/*2*)) *at-top*
using *eventually-gt-at-top*[*of 1*::*real*] *eventually-gt-at-top*[*of 1*/*b*]
by *eventually-elim*
(*insert bep*, *simp add*: *field-simps powr-mult* [*symmetric*] *ln-mult* [*symmetric*] *h2-def*)
hence *h2* ∈ Θ(λ*x*. *ln* (*b*∗*x*) *powr* (−*e*/*2*) − *ln x powr* (−*e*/*2*)) **by** (*rule bigthetaI-cong*)
finally show *g* ∈ *o*(λ*x*. *ln* (*b* ∗ *x*) *powr* (− *e* / *2*) − *ln x powr* (− *e* / *2*)) **.**
next
show *eventually* (λ*x*. *ln* (*b*∗*x*) *powr* (−*e*/*2*) − *ln x powr* (−*e*/*2*) ≥ *0*) *at-top*
using *eventually-gt-at-top*[*of 1*/*b*] *eventually-gt-at-top*[*of 1*::*real*]
by *eventually-elim* (*insert bep*, *auto intro*!: *powr-mono2′ simp*: *field-simps*)
qed

private lemma *ev1*:
negl (λ*x*. (*1* + *c* ∗ *inverse b* ∗ *ln x powr* (−(*1*+*e*))) *powr p* − *1*)
proof−
from *bep* **have** ((λ*x*. *c* ∗ *inverse b* ∗ *ln x powr* (−(*1*+*e*))) ⟶ *0*) *at-top*
by (*simp add*: *tendsto-0-smallo-1*)
have (λ*x*. (*1* + *c* ∗ *inverse b* ∗ *ln x powr* (−(*1*+*e*))) *powr p* − *1*)
∈ *O*(λ*x*. *c* ∗ *inverse b* ∗ *ln x powr* − (*1* + *e*))
using *bep* **by** (*intro one-plus-x-powr-taylor1-bigo*) (*simp add*: *tendsto-0-smallo-1*)
also from *bep* **have** *negl* (λ*x*. *c* ∗ *inverse b* ∗ *ln x powr* − (*1* + *e*)) **by** *simp*
finally show *?thesis* **.**
qed

private lemma *ev2-aux*:

defines $f \equiv \lambda x.\ (1 + 1/ln\ (b{*}x) * ln\ (1 + hb\ /\ b * ln\ x\ powr\ (-1-e)))\ powr\ (-e/2)$
obtains h **where** *eventually* $(\lambda x.\ f\ x \geq 1 + h\ x)$ *at-top* $h \in o(\lambda x.\ 1\ /\ ln\ x)$
proof (*rule that*[*of* $\lambda x.\ f\ x - 1$])
def $g \equiv \lambda x.\ 1/ln\ (b{*}x) * ln\ (1 + hb\ /\ b * ln\ x\ powr\ (-1-e))$
have *lim*: $((\lambda x.\ ln\ (1 + hb\ /\ b * ln\ x\ powr\ (-\ 1\ -\ e))) \longrightarrow 0)$ *at-top*
by (*rule tendsto-eq-rhs*[*OF tendsto-ln*[*OF tendsto-add*[*OF tendsto-const*, *of* -0]]])
(*insert bep*, *simp-all add*: *tendsto-0-smallo-1*)
hence *lim'*: $(g \longrightarrow 0)$ *at-top* **unfolding** *g-def*
by (*intro tendsto-mult-zero*) (*insert bep*, *simp add*: *tendsto-0-smallo-1*)
from *one-plus-x-powr-taylor2-bigo*[*OF this*, *of* $-e/2$]
have $(\lambda x.\ (1 + g\ x)\ powr\ (-e/2) - 1 - -\ e/2 * g\ x) \in O(\lambda x.\ (g\ x)^2)$ **.**
also from *lim'* **have** $(\lambda x.\ g\ x\ \hat{}\ 2) \in o(\lambda x.\ g\ x * 1)$ **unfolding** *power2-eq-square*
by (*intro landau-o.big-small-mult smalloI-tendsto*) *simp-all*
also have $o(\lambda x.\ g\ x * 1) = o(g)$ **by** *simp*
also have $(\lambda x.\ (1 + g\ x)\ powr\ (-e/2) - 1 - -\ e/2 * g\ x) = (\lambda x.\ f\ x - 1 + e/2 * g\ x)$
by (*simp add*: *f-def g-def*)
finally have *A*: $(\lambda x.\ f\ x - 1 + e\ /\ 2 * g\ x) \in O(g)$ **by** (*rule landau-o.small-imp-big*)
hence $(\lambda x.\ f\ x - 1 + e/2 * g\ x - e/2 * g\ x) \in O(g)$
by (*rule sum-in-bigo*) (*insert bep*, *simp*)
also have $(\lambda x.\ f\ x - 1 + e/2 * g\ x - e/2 * g\ x) = (\lambda x.\ f\ x - 1)$ **by** *simp*
finally have $(\lambda x.\ f\ x - 1) \in O(g)$ **.**
also from *bep lim* **have** $g \in o(\lambda x.\ 1\ /\ ln\ x)$ **unfolding** *g-def*
by (*auto intro*!: *smallo-1-tendsto-0*)
finally show $(\lambda x.\ f\ x - 1) \in o(\lambda x.\ 1\ /\ ln\ x)$ **.**
qed *simp-all*

private lemma *ev2*:
defines $f \equiv \lambda x.\ ln\ (b * x + hb * x\ /\ ln\ x\ powr\ (1 + e))\ powr\ (-e/2)$
obtains h **where**
negl h
eventually $(\lambda x.\ f\ x \geq ln\ (b * x)\ powr\ (-e/2) + h\ x)$ *at-top*
eventually $(\lambda x.\ |ln\ (b * x)\ powr\ (-e/2) + h\ x| < 1)$ *at-top*
proof −
def $f' \equiv \lambda x.\ (1 + 1\ /\ ln\ (b{*}x) * ln\ (1 + hb\ /\ b * ln\ x\ powr\ (-1-e)))\ powr\ (-e/2)$
from *ev2-aux* **obtain** g **where** g: *eventually* $(\lambda x.\ 1 + g\ x \leq f'\ x)$ *at-top* $g \in o(\lambda x.\ 1\ /\ ln\ x)$
unfolding *f'-def* **.**
def $h \equiv \lambda x.\ ln\ (b{*}x)\ powr\ (-e/2) * g\ x$
show *?thesis*
proof (*rule that*[*of h*])
from *bep g* **show** *negl h* **unfolding** *h-def*
by (*auto simp*: *powr-divide2*[*symmetric*] *elim*: *landau-o.small-big-trans*)
next
from *g*(*2*) **have** $g \in o(\lambda x.\ 1)$ **by** (*rule landau-o.small-big-trans*) *simp*
with *bep* **have** *eventually* $(\lambda x.\ |ln\ (b{*}x)\ powr\ (-e/2) * (1 + g\ x)| < 1)$ *at-top*

by (*intro smallo-imp-abs-less*) *simp-all*
thus *eventually* (*λx.* |*ln* (*b∗x*) *powr* (−*e*/*2*) + *h x*| < *1*) *at-top*
by (*simp add*: *algebra-simps h-def*)
next
from *eventually-gt-at-top*[*of 1*/*b*] **and** *g*(*1*)
show *eventually* (*λx. f x* ≥ *ln* (*b∗x*) *powr* (−*e*/*2*) + *h x*) *at-top*
proof *eventually-elim*
case (*elim x*)
from *bep* **have** *b* ∗ *x* + *hb* ∗ *x* / *ln x powr* (*1* + *e*) = *b∗x* ∗ (*1* + *hb* / *b* ∗ *ln x powr* (−*1* − *e*))
by (*simp add*: *field-simps powr-divide2* [*symmetric*] *powr-add powr-minus*)
also from *elim*(*1*) *bep*
have *ln* . . . = *ln* (*b∗x*) ∗ (*1* + *1*/*ln* (*b∗x*) ∗ *ln* (*1* + *hb* / *b* ∗ *ln x powr* (−*1*−*e*)))
by (*subst ln-mult*) (*simp-all add*: *add-pos-nonneg field-simps*)
also from *elim*(*1*) *bep* **have** . . . *powr* (−*e*/*2*) = *ln* (*b∗x*) *powr* (−*e*/*2*) ∗ *f* ′ *x*
by (*subst powr-mult*) (*simp-all add*: *field-simps f* ′*-def*)
also from *elim* **have** . . . ≥ *ln* (*b∗x*) *powr* (−*e*/*2*) ∗ (*1* + *g x*)
by (*intro mult-left-mono*) *simp-all*
finally show *f x* ≥ *ln* (*b∗x*) *powr* (−*e*/*2*) + *h x*
by (*simp add*: *f-def h-def algebra-simps*)
qed
qed
qed

private lemma *ev21*:
obtains *g* **where**
negl g
eventually (*λx. 1* + *ln* (*b* ∗ *x* + *hb* ∗ *x* / *ln x powr* (*1* + *e*)) *powr* (−*e*/*2*) ≥ *1* + *ln* (*b* ∗ *x*) *powr* (−*e*/*2*) + *g x*) *at-top*
eventually (*λx. 1* + *ln* (*b* ∗ *x*) *powr* (−*e*/*2*) + *g x* > *0*) *at-top*
proof−
from *ev2* **guess** *g* **. note** *g* = *this*
from *g*(*3*) **have** *eventually* (*λx. 1* + *ln* (*b* ∗ *x*) *powr* (−*e*/*2*) + *g x* > *0*) *at-top*
by *eventually-elim simp*
with *g*(*1*,*2*) **show** *?thesis* **by** (*intro that*[*of g*]) *simp-all*
qed

private lemma *ev22*:
obtains *g* **where**
negl g
eventually (*λx. 1* − *ln* (*b* ∗ *x* + *hb* ∗ *x* / *ln x powr* (*1* + *e*)) *powr* (−*e*/*2*) ≤ *1* − *ln* (*b* ∗ *x*) *powr* (−*e*/*2*) − *g x*) *at-top*
eventually (*λx. 1* − *ln* (*b* ∗ *x*) *powr* (−*e*/*2*) − *g x* > *0*) *at-top*
proof−
from *ev2* **guess** *g* **. note** *g* = *this*
from *g*(*2*) **have** *eventually* (*λx. 1* − *ln* (*b* ∗ *x* + *hb* ∗ *x* / *ln x powr* (*1* + *e*)) *powr* (−*e*/*2*) ≤

$1 - ln\ (b * x)\ powr\ (-e/2) - g\ x)$ *at-top*
  by *eventually-elim simp*
 moreover from $g(3)$ **have** *eventually* $(\lambda x.\ 1 - ln\ (b * x)\ powr\ (-e/2) - g\ x > 0)$ *at-top*
  by *eventually-elim simp*
 ultimately show *?thesis* **using** $g(1)$ **by** (*intro that*[*of g*]) *simp-all*
qed

lemma *asymptotics1*:
 shows *eventually* $(\lambda x.$
   $(1 + c * inverse\ b * ln\ x\ powr\ -(1+e))\ powr\ p *$
   $(1 + ln\ (b * x + hb * x\ /\ ln\ x\ powr\ (1 + e))\ powr\ (-\ e\ /\ 2)) \geq$
   $1 + (ln\ x\ powr\ (-e/2)))$ *at-top*
proof−
 let $?f = \lambda x.\ (1 + c * inverse\ b * ln\ x\ powr\ -(1+e))\ powr\ p$
 let $?g = \lambda x.\ 1 + ln\ (b * x + hb * x\ /\ ln\ x\ powr\ (1 + e))\ powr\ (-\ e\ /\ 2)$
 def $f \equiv \lambda x.\ 1 - ?f\ x$
 from *ev1*[*of c*] **have** *negl f* **unfolding** *f-def*
  by (*subst landau-o.small.uminus-in-iff* [*symmetric*]) *simp*
 from *landau-o.smallD*[*OF this zero-less-one*]
  have *f*: *eventually* $(\lambda x.\ f\ x \leq ln\ (b*x)\ powr\ -(e/2+1))$ *at-top*
  by *eventually-elim* (*simp add*: *f-def*)

 from *ev21* **guess** *g* **. note** *g = this*
 def $h \equiv \lambda x.\ -g\ x + f\ x + f\ x * ln\ (b*x)\ powr\ (-e/2) + f\ x * g\ x$

 have *A*: *eventually* $(\lambda x.\ ?f\ x * ?g\ x \geq 1 + ln\ (b*x)\ powr\ (-e/2) - h\ x)$ *at-top*
  using $g(2,3)$ *f*
 proof *eventually-elim*
  case (*elim x*)
  let $?t = ln\ (b*x)\ powr\ (-e/2)$
  have $1 + ?t - h\ x = (1 - f\ x) * (1 + ln\ (b*x)\ powr\ (-e/2) + g\ x)$
   by (*simp add*: *algebra-simps h-def*)
  also from *elim* **have** $?f\ x * ?g\ x \geq (1 - f\ x) * (1 + ln\ (b*x)\ powr\ (-e/2) + g\ x)$
   by (*intro mult-mono*[*OF - elim*(*1*)]) (*simp-all add*: *algebra-simps f-def*)
  finally show $?f\ x * ?g\ x \geq 1 + ln\ (b*x)\ powr\ (-e/2) - h\ x$ **.**
 qed
 from *bep* ‹*negl f*› $g(1)$ **have** *negl h* **unfolding** *h-def*
  by (*fastforce intro*!: *sum-in-smallo landau-o.small.mult simp*: *powr-divide2*[*symmetric*]

      intro: *landau-o.small-trans*)+
 from *ev4*[*OF this*] *A* **show** *?thesis* **by** *eventually-elim simp*
qed

lemma *asymptotics2*:
 shows *eventually* $(\lambda x.$
   $(1 + c * inverse\ b * ln\ x\ powr\ -(1+e))\ powr\ p *$

```
        (1 − ln (b ∗ x + hb ∗ x / ln x powr (1 + e)) powr (− e / 2)) ≤
        1 − (ln x powr (−e/2))) at-top
proof−
  let ?f = λx. (1 + c ∗ inverse b ∗ ln x powr −(1+e)) powr p
  let ?g = λx. 1 − ln (b ∗ x + hb ∗ x / ln x powr (1 + e)) powr (− e / 2)

  def f ≡ λx. 1 − ?f x
  from ev1[of c] have negl f unfolding f-def
    by (subst landau-o.small.uminus-in-iff [symmetric]) simp
  from landau-o.smallD[OF this zero-less-one]
    have f: eventually (λx. f x ≤ ln (b∗x) powr −(e/2+1)) at-top
    by eventually-elim (simp add: f-def)

  from ev22 guess g . note g = this
  def h ≡ λx. −g x − f x + f x ∗ ln (b∗x) powr (−e/2) + f x ∗ g x
  have ((λx. ln (b ∗ x + hb ∗ x / ln x powr (1 + e)) powr − (e / 2)) ⟶ 0)
at-top
    apply (insert bep, intro tendsto-neg-powr, simp)
    apply (rule filterlim-compose[OF ln-at-top])
    apply (rule filterlim-at-top-smallomega-1, simp)
    using eventually-gt-at-top[of max 1 (1/b)]
    apply (auto elim!: eventually-mono intro!: add-pos-nonneg simp: field-simps)
    done
  hence ev-g: eventually (λx. |1 − ?g x| < 1) at-top
    by (intro smallo-imp-abs-less smalloI-tendsto) simp-all

  have A: eventually (λx. ?f x ∗ ?g x ≤ 1 − ln (b∗x) powr (−e/2) + h x) at-top
    using g(2,3) ev-g f
  proof eventually-elim
    case (elim x)
    let ?t = ln (b∗x) powr (−e/2)
    from elim have ?f x ∗ ?g x ≤ (1 − f x) ∗ (1 − ln (b∗x) powr (−e/2) − g x)
      by (intro mult-mono) (simp-all add: f-def)
    also have ... = 1 − ?t + h x by (simp add: algebra-simps h-def)
    finally show ?f x ∗ ?g x ≤ 1 − ln (b∗x) powr (−e/2) + h x .
  qed
  from bep ‹negl f› g(1) have negl h unfolding h-def
   by (fastforce intro!: sum-in-smallo landau-o.small.mult simp: powr-divide2[symmetric]

                intro: landau-o.small-trans)+
  from ev4[OF this] A show ?thesis by eventually-elim simp
qed

lemma asymptotics3: eventually (λx. (1 + (ln x powr (−e/2))) / 2 ≤ 1) at-top
  (is eventually (λx. ?f x ≤ 1) -)
proof (rule eventually-mp[OF always-eventually], clarify)
  from bep have (?f ⟶ 1/2) at-top
    by (force intro: tendsto-eq-intros tendsto-neg-powr ln-at-top)
  hence ⋀e. e>0 ⟹ eventually (λx. |?f x − 0.5| < e) at-top
```

by (*subst* (*asm*) *tendsto-iff*) (*simp add*: *dist-real-def*)
from *this*[*of 0.5*] **show** *eventually* (λ*x*. |*?f x* − *0.5*| < *0.5*) *at-top* **by** *simp*
fix *x* **assume** |*?f x* − *0.5*| < *0.5*
thus *?f x* ≤ *1* **by** *simp*
qed

lemma *asymptotics4*: *eventually* (λ*x*. (*1* − (*ln x powr* (−*e*/*2*))) ∗ *2* ≥ *1*) *at-top*
(**is** *eventually* (λ*x*. *?f x* ≥ *1*) -)
proof (*rule eventually-mp*[*OF always-eventually*], *clarify*)
from *bep* **have** (*?f* ⟶ *2*) *at-top*
by (*force intro*: *tendsto-eq-intros tendsto-neg-powr ln-at-top*)
hence ⋀*e*. *e*>*0* ⟹ *eventually* (λ*x*. |*?f x* − *2*| < *e*) *at-top*
by (*subst* (*asm*) *tendsto-iff*) (*simp add*: *dist-real-def*)
from *this*[*of 1*] **show** *eventually* (λ*x*. |*?f x* − *2*| < *1*) *at-top* **by** *simp*
fix *x* **assume** |*?f x* − *2*| < *1*
thus *?f x* ≥ *1* **by** *simp*
qed

lemma *asymptotics5*: *eventually* (λ*x*. *ln* (*b*∗*x* − *hb*∗*x*∗*ln x powr* −(*1*+*e*)) *powr* (−*e*/*2*) < *1*) *at-top*
proof−
from *bep* **have** ((λ*x*. *b* − *hb* ∗ *ln x powr* −(*1*+*e*)) ⟶ *b* − *0*) *at-top*
by (*intro tendsto-intros tendsto-mult-right-zero tendsto-neg-powr ln-at-top*) *simp-all*
hence *LIM x at-top*. (*b* − *hb* ∗ *ln x powr* −(*1*+*e*)) ∗ *x* :> *at-top*
by (*rule filterlim-tendsto-pos-mult-at-top*[*OF - - filterlim-ident*], *insert bep*) *simp-all*
also have (λ*x*. (*b* − *hb* ∗ *ln x powr* −(*1*+*e*)) ∗ *x*) = (λ*x*. *b*∗*x* − *hb*∗*x*∗*ln x powr* −(*1*+*e*))
by (*intro ext*) (*simp add*: *algebra-simps*)
finally have *filterlim* ... *at-top at-top* **.**
with *bep* **have** ((λ*x*. *ln* (*b*∗*x* − *hb*∗*x*∗*ln x powr* −(*1*+*e*)) *powr* −(*e*/*2*)) ⟶ *0*) *at-top*
by (*intro tendsto-neg-powr filterlim-compose*[*OF ln-at-top*]) *simp-all*
hence *eventually* (λ*x*. |*ln* (*b*∗*x* − *hb*∗*x*∗*ln x powr* −(*1*+*e*)) *powr* (−*e*/*2*)| < *1*) *at-top*
by (*subst* (*asm*) *tendsto-iff*) (*simp add*: *dist-real-def*)
thus *?thesis* **by** *simp*
qed

lemma *asymptotics6*: *eventually* (λ*x*. *hb* / *ln x powr* (*1* + *e*) < *b*/*2*) *at-top*
and *asymptotics7*: *eventually* (λ*x*. *hb* / *ln x powr* (*1* + *e*) < (*1* − *b*) / *2*) *at-top*
and *asymptotics8*: *eventually* (λ*x*. *x*∗(*1* − *b* − *hb* / *ln x powr* (*1* + *e*)) > *1*) *at-top*
proof−
from *bep* **have** *A*: (λ*x*. *hb* / *ln x powr* (*1* + *e*)) ∈ *o*(λ-. *1*) **by** *simp*
from *bep* **have** *B*: *b*/*3* > *0* **and** *C*: (*1* − *b*)/*3* > *0* **by** *simp-all*
from *landau-o.smallD*[*OF A B*] **show** *eventually* (λ*x*. *hb* / *ln x powr* (*1*+*e*) < *b*/*2*) *at-top*

by *eventually-elim* (*insert bep*, *simp*)
from *landau-o.smallD*[*OF A C*] **show** *eventually* (λx. *hb* / *ln x powr* (*1* + *e*) < (*1* − *b*)/*2*) *at-top*
by *eventually-elim* (*insert bep*, *simp*)

from *bep* **have** (λx. *hb* / *ln x powr* (*1* + *e*)) ∈ *o*(λ-. *1*) (*1* − *b*) / *2* > *0* **by** *simp-all*
from *landau-o.smallD*[*OF this*] *eventually-gt-at-top*[*of 1*::*real*]
have *A*: *eventually* (λx. *1* − *b* − *hb* / *ln x powr* (*1* + *e*) > *0*) *at-top*
by *eventually-elim* (*insert bep*, *simp add*: *field-simps*)
from *bep* **have** (λx. *x* ∗ (*1* − *b* − *hb* / *ln x powr* (*1*+*e*))) ∈ ω(λ-. *1*) (*0*::*real*) < *2* **by** *simp-all*
from *landau-omega.smallD*[*OF this*] *A* *eventually-gt-at-top*[*of 0*::*real*]
show *eventually* (λx. *x*∗(*1* − *b* − *hb* / *ln x powr* (*1* + *e*)) > *1*) *at-top*
by *eventually-elim* (*simp-all add*: *abs-mult*)
qed

end
end

definition *akra-bazzi-asymptotic1 b hb e p x* ⟷
(*1* − *hb* ∗ *inverse b* ∗ *ln x powr* −(*1*+*e*)) *powr p* ∗ (*1* + *ln* (*b*∗*x* + *hb*∗*x*/*ln x powr* (*1*+*e*)) *powr* (−*e*/*2*))
≥ *1* + (*ln x powr* (−*e*/*2*) :: *real*)
definition *akra-bazzi-asymptotic1′ b hb e p x* ⟷
(*1* + *hb* ∗ *inverse b* ∗ *ln x powr* −(*1*+*e*)) *powr p* ∗ (*1* + *ln* (*b*∗*x* + *hb*∗*x*/*ln x powr* (*1*+*e*)) *powr* (−*e*/*2*))
≥ *1* + (*ln x powr* (−*e*/*2*) :: *real*)
definition *akra-bazzi-asymptotic2 b hb e p x* ⟷
(*1* + *hb* ∗ *inverse b* ∗ *ln x powr* −(*1*+*e*)) *powr p* ∗ (*1* − *ln* (*b*∗*x* + *hb*∗*x*/*ln x powr* (*1*+*e*)) *powr* (−*e*/*2*))
≤ *1* − *ln x powr* (−*e*/*2* :: *real*)
definition *akra-bazzi-asymptotic2′ b hb e p x* ⟷
(*1* − *hb* ∗ *inverse b* ∗ *ln x powr* −(*1*+*e*)) *powr p* ∗ (*1* − *ln* (*b*∗*x* + *hb*∗*x*/*ln x powr* (*1*+*e*)) *powr* (−*e*/*2*))
≤ *1* − *ln x powr* (−*e*/*2* :: *real*)
definition *akra-bazzi-asymptotic3 e x* ⟷ (*1* + (*ln x powr* (−*e*/*2*))) / *2* ≤ (*1*::*real*)
definition *akra-bazzi-asymptotic4 e x* ⟷ (*1* − (*ln x powr* (−*e*/*2*))) ∗ *2* ≥ (*1*::*real*)
definition *akra-bazzi-asymptotic5 b hb e x* ⟷
ln (*b*∗*x* − *hb*∗*x*∗*ln x powr* −(*1*+*e*)) *powr* (−*e*/*2*::*real*) < *1*

definition *akra-bazzi-asymptotic6 b hb e x* ⟷ *hb* / *ln x powr* (*1* + *e* :: *real*) < *b*/*2*
definition *akra-bazzi-asymptotic7 b hb e x* ⟷ *hb* / *ln x powr* (*1* + *e* :: *real*) < (*1* − *b*) / *2*
definition *akra-bazzi-asymptotic8 b hb e x* ⟷ *x*∗(*1* − *b* − *hb* / *ln x powr* (*1* +

e :: *real*)) > *1*

definition *akra-bazzi-asymptotics b hb e p x* $\longleftrightarrow$
akra-bazzi-asymptotic1 b hb e p x $\wedge$ *akra-bazzi-asymptotic1′ b hb e p x* $\wedge$
akra-bazzi-asymptotic2 b hb e p x $\wedge$ *akra-bazzi-asymptotic2′ b hb e p x* $\wedge$
akra-bazzi-asymptotic3 e x $\wedge$ *akra-bazzi-asymptotic4 e x* $\wedge$ *akra-bazzi-asymptotic5 b hb e x* $\wedge$
akra-bazzi-asymptotic6 b hb e x $\wedge$ *akra-bazzi-asymptotic7 b hb e x* $\wedge$
akra-bazzi-asymptotic8 b hb e x

lemmas *akra-bazzi-asymptotic-defs* =
akra-bazzi-asymptotic1-def akra-bazzi-asymptotic1′-def
akra-bazzi-asymptotic2-def akra-bazzi-asymptotic2′-def akra-bazzi-asymptotic3-def
akra-bazzi-asymptotic4-def akra-bazzi-asymptotic5-def akra-bazzi-asymptotic6-def
akra-bazzi-asymptotic7-def akra-bazzi-asymptotic8-def akra-bazzi-asymptotics-def

lemma *akra-bazzi-asymptotics*:
assumes $\bigwedge b.\ b \in$ *set bs* $\Longrightarrow b \in \{0{<}..{<}1\}$
assumes *hb* > *0* *e* > *0*
shows *eventually* ($\lambda x.\ \forall b{\in}$*set bs*. *akra-bazzi-asymptotics b hb e p x*) *at-top*
proof (*intro eventually-ball-finite ballI*)
fix *b* **assume** *b* $\in$ *set bs*
with *assms* **interpret** *akra-bazzi-asymptotics-bep b e p hb* **by** *unfold-locales auto*

show *eventually* ($\lambda x.$ *akra-bazzi-asymptotics b hb e p x*) *at-top*
unfolding *akra-bazzi-asymptotic-defs*
using *asymptotics1*[*of* −*c* **for** *c*] *asymptotics2*[*of* −*c* **for** *c*]
by (*intro eventually-conj asymptotics1 asymptotics2 asymptotics3*
asymptotics4 asymptotics5 asymptotics6 asymptotics7 asymptotics8)
simp-all
qed *simp*

end

3 The continuous Akra-Bazzi theorem

theory *Akra-Bazzi-Real*
imports
Complex-Main
../Landau-Symbols/Landau-Symbols
Akra-Bazzi-Asymptotics
begin

We want to be generic over the integral definition used; we fix some arbitrary notions of integrability and integral and assume just the properties we need. The user can then instantiate the theorems with any desired integral definition.

locale *akra-bazzi-integral* =

```
  fixes integrable :: (real ⇒ real) ⇒ real ⇒ real ⇒ bool
    and integral   :: (real ⇒ real) ⇒ real ⇒ real ⇒ real
  assumes integrable-const: c ≥ 0 ⟹ integrable (λ-. c) a b
      and integral-const:   c ≥ 0 ⟹ a ≤ b ⟹ integral (λ-. c) a b = (b − a) ∗ c
      and integrable-subinterval:
            integrable f a b ⟹ a ≤ a′ ⟹ b′ ≤ b ⟹ integrable f a′ b′
      and integral-le:
            integrable f a b ⟹ integrable g a b ⟹ (⋀x. x ∈ {a..b} ⟹ f x ≤ g x)
⟹
                integral f a b ≤ integral g a b
      and integral-combine:
            a ≤ c ⟹ c ≤ b ⟹ integrable f a b ⟹
                integral f a c + integral f c b = integral f a b
begin
lemma integral-nonneg:
  a ≤ b ⟹ integrable f a b ⟹ (⋀x. x ∈ {a..b} ⟹ f x ≥ 0) ⟹ integral f a b
≥ 0
  using integral-le[OF integrable-const[of 0], of f a b]  by (simp add: integral-const)
end


declare setsum.cong[fundef-cong]

lemma strict-mono-imp-ex1-real:
  fixes f :: real ⇒ real
  assumes lim-neg-inf: LIM x at-bot. f x :> at-top
  assumes lim-inf: (f ⟶ z) at-top
  assumes mono: ⋀a b. a < b ⟹ f b < f a
  assumes cont: ⋀x. isCont f x
  assumes y-greater-z: z < y
  shows   ∃!x. f x = y
proof (rule ex-ex1I)
  fix a b assume f a = y f b = y
  thus a = b by (cases rule: linorder-cases[of a b]) (auto dest: mono)
next
 from lim-neg-inf have eventually (λx. y ≤ f x) at-bot by (subst (asm) filterlim-at-top)
simp
 then obtain l where l: ⋀x. x ≤ l ⟹ y ≤ f x by (subst (asm) eventually-at-bot-linorder)
auto

  from order-tendstoD(2)[OF lim-inf y-greater-z]
   obtain u where u: ⋀x. x ≥ u ⟹ f x < y by (subst (asm) eventually-at-top-linorder)
auto
  def a ≡ min l u and b ≡ max l u
  have a: f a ≥ y unfolding a-def by (intro l) simp
  moreover have b: f b < y unfolding b-def by (intro u) simp
  moreover have a-le-b: a ≤ b by (simp add: a-def b-def)
  ultimately have ∃x≥a. x ≤ b ∧ f x = y using cont by (intro IVT2) auto
  thus ∃x. f x = y by blast
```

qed

The parameter p in the Akra-Bazzi theorem always exists and is unique.

definition *akra-bazzi-exponent* :: *real list* ⇒ *real list* ⇒ *real* **where**
akra-bazzi-exponent as bs ≡ (*THE p*. ($\sum$ *i*<*length as*. *as*!*i* ∗ *bs*!*i powr p*) = *1*)

locale *akra-bazzi-params* =
fixes *k* :: *nat* **and** *as bs* :: *real list*
assumes *length-as*: *length as* = *k*
and *length-bs*: *length bs* = *k*
and *k-not-0*: $k \neq 0$
and *a-ge-0*: $a \in set\ as \Longrightarrow a \geq 0$
and *b-bounds*: $b \in set\ bs \Longrightarrow b \in \{0<..<1\}$
begin

abbreviation *p* :: *real* **where** *p* ≡ *akra-bazzi-exponent as bs*

lemma *p-def*: *p* = (*THE p*. ($\sum$ *i*<*k*. *as*!*i* ∗ *bs*!*i powr p*) = *1*)
by (*simp add*: *akra-bazzi-exponent-def length-as*)

lemma *b-pos*: $b \in set\ bs \Longrightarrow b > 0$ **and** *b-less-1*: $b \in set\ bs \Longrightarrow b < 1$
using *b-bounds* **by** *simp-all*

lemma *as-nonempty* [*simp*]: *as* ≠ [] **and** *bs-nonempty* [*simp*]: *bs* ≠ []
using *length-as length-bs k-not-0* **by** *auto*

lemma *a-in-as*[*intro*, *simp*]: $i < k \Longrightarrow as\ !\ i \in set\ as$
by (*rule nth-mem*) (*simp add*: *length-as*)

lemma *b-in-bs*[*intro*, *simp*]: $i < k \Longrightarrow bs\ !\ i \in set\ bs$
by (*rule nth-mem*) (*simp add*: *length-bs*)

end

locale *akra-bazzi-params-nonzero* =
fixes *k* :: *nat* **and** *as bs* :: *real list*
assumes *length-as*: *length as* = *k*
and *length-bs*: *length bs* = *k*
and *a-ge-0*: $a \in set\ as \Longrightarrow a \geq 0$
and *ex-a-pos*: $\exists\, a \in set\ as.\ a > 0$
and *b-bounds*: $b \in set\ bs \Longrightarrow b \in \{0<..<1\}$
begin

sublocale *akra-bazzi-params k as bs*
by *unfold-locales* (*insert length-as length-bs a-ge-0 ex-a-pos b-bounds*, *auto*)

lemma *akra-bazzi-p-strict-mono*:
assumes $x < y$

shows $(\sum i{<}k.\ as!i * bs!i\ powr\ y) < (\sum i{<}k.\ as!i * bs!i\ powr\ x)$
proof (*intro setsum-strict-mono-ex1 ballI*)
from *ex-a-pos* **obtain** a **where** $a \in set\ as\ a > 0$ **by** *blast*
then obtain i **where** $i < k\ as!i > 0$ **by** (*force simp*: *in-set-conv-nth length-as*)
with *b-bounds* ‹$x < y$› **have** $as!i * bs!i\ powr\ y < as!i * bs!i\ powr\ x$
by (*intro mult-strict-left-mono powr-less-mono'*) *auto*
with ‹$i < k$› **show** $\exists i{\in}\{..{<}k\}.\ as!i * bs!i\ powr\ y < as!i * bs!i\ powr\ x$ **by** *blast*
next
fix i **assume** $i \in \{..{<}k\}$
with *a-ge-0 b-bounds*[*of bs*!*i*] ‹$x < y$› **show** $as!i * bs!i\ powr\ y \le as!i * bs!i\ powr$
x
by (*intro mult-left-mono powr-mono'*) *simp-all*
qed *simp-all*

lemma *akra-bazzi-p-mono*:
assumes $x \le y$
shows $(\sum i{<}k.\ as!i * bs!i\ powr\ y) \le (\sum i{<}k.\ as!i * bs!i\ powr\ x)$
apply (*cases* $x < y$)
using *akra-bazzi-p-strict-mono*[*of x y*] *assms* **apply** *simp-all*
done

lemma *akra-bazzi-p-unique*:
$\exists!p.\ (\sum i{<}k.\ as!i * bs!i\ powr\ p) = 1$
proof (*rule strict-mono-imp-ex1-real*)
from *as-nonempty* **have** [*simp*]: $k > 0$ **by** (*auto simp*: *length-as*[*symmetric*])
have [*simp*]: $\bigwedge i.\ i < k \Longrightarrow as!i \ge 0$ **by** (*rule a-ge-0*) *simp*
from *ex-a-pos* **obtain** a **where** $a \in set\ as\ a > 0$ **by** *blast*
then obtain i **where** i: $i < k\ as!i > 0$ **by** (*force simp*: *in-set-conv-nth length-as*)

hence *LIM p at-bot*. $as!i * bs!i\ powr\ p$:> *at-top* **using** *b-bounds i*
by (*intro filterlim-tendsto-pos-mult-at-top*[*OF tendsto-const*] *powr-at-bot-neg*)
simp-all
moreover have $\forall p.\ as!i{*}bs!i\ powr\ p \le (\sum i{\in}\{..{<}k\}.\ as\ !\ i * bs\ !\ i\ powr\ p)$
proof
fix p :: *real*
from *a-ge-0 b-bounds* **have** $(\sum i{\in}\{..{<}k\}{-}\{i\}.\ as\ !\ i * bs\ !\ i\ powr\ p) \ge 0$
by (*intro setsum-nonneg mult-nonneg-nonneg*) *simp-all*
also have $as!i * bs!i\ powr\ p + ... = (\sum i{\in}insert\ i\ \{..{<}k\}.\ as\ !\ i * bs\ !\ i\ powr$
$p)$
by (*simp add*: *setsum.insert-remove*)
also from i **have** $insert\ i\ \{..{<}k\} = \{..{<}k\}$ **by** *blast*
finally show $as!i{*}bs!i\ powr\ p \le (\sum i{\in}\{..{<}k\}.\ as\ !\ i * bs\ !\ i\ powr\ p)$ **by** *simp*
qed
ultimately show *LIM p at-bot*. $\sum i{<}k.\ as\ !\ i * bs\ !\ i\ powr\ p$:> *at-top*
by (*rule filterlim-at-top-mono*[*OF - always-eventually*])
next
from *b-bounds* **show** $((\lambda x.\ \sum i{<}k.\ as\ !\ i * bs\ !\ i\ powr\ x) \longrightarrow (\sum i{<}k.\ 0))$
at-top

by (*intro tendsto-setsum tendsto-mult-right-zero powr-at-top-neg*) *simp-all*
next
fix x
from *b-bounds* **have** A: $\bigwedge i.\ i < k \Longrightarrow bs\ !\ i > 0$ **by** *simp*
show $isCont\ (\lambda x.\ \sum i{<}k.\ as\ !\ i * bs\ !\ i\ powr\ x)\ x$
using *b-bounds*[*OF nth-mem*] **by** (*intro continuous-intros*) (*auto dest*: A)
qed (*simp-all add*: *akra-bazzi-p-strict-mono*)

lemma *p-props*: $(\sum i{<}k.\ as!i * bs!i\ powr\ p) = 1$
and *p-unique*: $(\sum i{<}k.\ as!i * bs!i\ powr\ p') = 1 \Longrightarrow p = p'$
proof−
from *theI'*[*OF akra-bazzi-p-unique*] *the1-equality*[*OF akra-bazzi-p-unique*]
show $(\sum i{<}k.\ as!i * bs!i\ powr\ p) = 1$ $(\sum i{<}k.\ as!i * bs!i\ powr\ p') = 1 \Longrightarrow p = p'$
unfolding *p-def* **by** − *blast*+
qed

lemma *p-greaterI*: $1 < (\sum i{<}k.\ as!i * bs!i\ powr\ p') \Longrightarrow p' < p$
by (*rule disjE*[*OF le-less-linear*, *of p p'*], *drule akra-bazzi-p-mono*, *subst* (*asm*) *p-props*, *simp-all*)

lemma *p-lessI*: $1 > (\sum i{<}k.\ as!i * bs!i\ powr\ p') \Longrightarrow p' > p$
by (*rule disjE*[*OF le-less-linear*, *of p' p*], *drule akra-bazzi-p-mono*, *subst* (*asm*) *p-props*, *simp-all*)

lemma *p-geI*: $1 \leq (\sum i{<}k.\ as!i * bs!i\ powr\ p') \Longrightarrow p' \leq p$
by (*rule disjE*[*OF le-less-linear*, *of p' p*], *simp*, *drule akra-bazzi-p-strict-mono*, *subst* (*asm*) *p-props*, *simp-all*)

lemma *p-leI*: $1 \geq (\sum i{<}k.\ as!i * bs!i\ powr\ p') \Longrightarrow p' \geq p$
by (*rule disjE*[*OF le-less-linear*, *of p p'*], *simp*, *drule akra-bazzi-p-strict-mono*, *subst* (*asm*) *p-props*, *simp-all*)

lemma *p-boundsI*: $(\sum i{<}k.\ as!i * bs!i\ powr\ x) \leq 1 \wedge (\sum i{<}k.\ as!i * bs!i\ powr\ y) \geq 1 \Longrightarrow p \in \{y..x\}$
by (*elim conjE*, *drule p-leI*, *drule p-geI*, *simp*)

lemma *p-boundsI'*: $(\sum i{<}k.\ as!i * bs!i\ powr\ x) < 1 \wedge (\sum i{<}k.\ as!i * bs!i\ powr\ y) > 1 \Longrightarrow p \in \{y{<}..{<}x\}$
by (*elim conjE*, *drule p-lessI*, *drule p-greaterI*, *simp*)

lemma *p-nonneg*: $listsum\ as \geq 1 \Longrightarrow p \geq 0$
proof (*rule p-geI*)
assume $listsum\ as \geq 1$
also have ... $= (\sum i{<}k.\ as!i)$ **by** (*simp add*: *listsum-setsum-nth length-as atLeast0LessThan*)
also {
fix i **assume** $i < k$
with *b-bounds* **have** $bs!i > 0$ **by** *simp*
hence $as!i * bs!i\ powr\ 0 = as!i$ **by** *simp*

}
hence $(\sum i<k.\ as!i) = (\sum i<k.\ as!i * bs!i\ powr\ 0)$ **by** (*intro setsum.cong*) *simp-all*
finally show $1 \leq (\sum i<k.\ as\ !\ i * bs\ !\ i\ powr\ 0)$ **.**
qed

end

locale *akra-bazzi-real-recursion* =
fixes *as bs* :: *real list* **and** *hs* :: (*real* ⇒ *real*) *list* **and** *k* :: *nat* **and** x_0 x_1 *hb e p* :: *real*
assumes *length-as*: *length as* = *k*
and *length-bs*: *length bs* = *k*
and *length-hs*: *length hs* = *k*
and *k-not-0*: $k \neq 0$
and *a-ge-0*: $a \in set\ as \Longrightarrow a \geq 0$
and *b-bounds*: $b \in set\ bs \Longrightarrow b \in \{0<..<1\}$

and *x0-ge-1*: $x_0 \geq 1$
and *x0-le-x1*: $x_0 \leq x_1$
and *x1-ge*: $b \in set\ bs \Longrightarrow x_1 \geq 2 * x_0 * inverse\ b$

and *e-pos*: $e > 0$
and *h-bounds*: $x \geq x_1 \Longrightarrow h \in set\ hs \Longrightarrow |h\ x| \leq hb * x\ /\ ln\ x\ powr\ (1 + e)$

and *asymptotics*: $x \geq x_0 \Longrightarrow b \in set\ bs \Longrightarrow$ *akra-bazzi-asymptotics b hb e p x*
begin

sublocale *akra-bazzi-params k as bs*
using *length-as length-bs k-not-0 a-ge-0 b-bounds* **by** *unfold-locales*

lemma *h-in-hs*[*intro*, *simp*]: $i < k \Longrightarrow hs\ !\ i \in set\ hs$
by (*rule nth-mem*) (*simp add*: *length-hs*)

lemma *x1-gt-1*: $x_1 > 1$
proof−
from *bs-nonempty* **obtain** *b* **where** $b \in set\ bs$ **by** (*cases bs*) *auto*
from *b-pos*[*OF this*] *b-less-1*[*OF this*] *x0-ge-1* **have** $1 < 2 * x_0 * inverse\ b$
by (*simp add*: *field-simps*)
also from *x1-ge* **and** ‹$b \in set\ bs$› **have** ... $\leq x_1$ **by** *simp*
finally show *?thesis* **.**
qed

lemma *x1-ge-1*: $x_1 \geq 1$ **using** *x1-gt-1* **by** *simp*

lemma *x1-pos*: $x_1 > 0$ **using** *x1-ge-1* **by** *simp*

lemma *bx-le-x*: $x \geq 0 \Longrightarrow b \in set\ bs \Longrightarrow\ b * x \leq x$
  using *b-pos*[*of b*] *b-less-1*[*of b*] **by** (*intro mult-left-le-one-le*) (*simp-all*)

lemma *x0-pos*: $x_0 > 0$ **using** *x0-ge-1* **by** *simp*

lemma
  assumes $x \geq x_0$ $b \in set\ bs$
  shows *x0-hb-bound0*: $hb\ /\ ln\ x\ powr\ (1 + e) < b/2$
  and *x0-hb-bound1*: $hb\ /\ ln\ x\ powr\ (1 + e) < (1 - b)\ /\ 2$
  and *x0-hb-bound2*: $x*(1 - b - hb\ /\ ln\ x\ powr\ (1 + e)) > 1$
using *asymptotics*[*OF assms*] **unfolding** *akra-bazzi-asymptotic-defs* **by** *blast+*

lemma *step-diff*:
  assumes $i < k$ $x \geq x_1$
  shows $bs\ !\ i * x + (hs\ !\ i)\ x + 1 < x$
proof−
  have $bs\ !\ i * x + (hs\ !\ i)\ x + 1 \leq bs\ !\ i * x + |(hs\ !\ i)\ x| + 1$ **by** *simp*
  also from *assms* **have** $|(hs\ !\ i)\ x| \leq hb * x\ /\ ln\ x\ powr\ (1 + e)$ **by** (*simp add*: *h-bounds*)
  also from *assms x0-le-x1* **have** $x*(1 - bs\ !\ i - hb\ /\ ln\ x\ powr\ (1 + e)) > 1$
    by (*simp add*: *x0-hb-bound2*)
  hence $bs\ !\ i * x + hb * x\ /\ ln\ x\ powr\ (1 + e) + 1 < x$ **by** (*simp add*: *algebra-simps*)
  finally show *?thesis* **by** *simp*
qed

lemma *step-le-x*: $i < k \Longrightarrow x \geq x_1 \Longrightarrow bs\ !\ i * x + (hs\ !\ i)\ x \leq x$
  by (*drule* (*1*) *step-diff*) *simp*

lemma *x0-hb-bound0'*: $\bigwedge x\ b.\ x \geq x_0 \Longrightarrow b \in set\ bs \Longrightarrow hb\ /\ ln\ x\ powr\ (1 + e) < b$
  by (*drule* (*1*) *x0-hb-bound0*, *erule less-le-trans*) (*simp add*: *b-pos*)

lemma *step-pos*:
  assumes $i < k$ $x \geq x_1$
  shows $bs\ !\ i * x + (hs\ !\ i)\ x > 0$
proof−
  from *assms x0-le-x1* **have** $hb\ /\ ln\ x\ powr\ (1 + e) < bs\ !\ i$ **by** (*simp add*: *x0-hb-bound0'*)
  with *assms x0-pos x0-le-x1* **have** $x * 0 < x * (bs\ !\ i - hb\ /\ ln\ x\ powr\ (1 + e))$ **by** *simp*
  also have ... $= bs\ !\ i * x - hb * x\ /\ ln\ x\ powr\ (1 + e)$
    by (*simp add*: *algebra-simps*)
  also from *assms* **have** $-hb * x\ /\ ln\ x\ powr\ (1 + e) \leq -|(hs\ !\ i)\ x|$ **by** (*simp add*: *h-bounds*)
  hence $bs\ !\ i * x - hb * x\ /\ ln\ x\ powr\ (1 + e) \leq bs\ !\ i * x + -|(hs\ !\ i)\ x|$ **by** *simp*

also have $-|(hs\ !\ i)\ x| \leq (hs\ !\ i)\ x$ **by** *simp*
finally show $bs\ !\ i * x + (hs\ !\ i)\ x > 0$ **by** *simp*
qed

lemma *step-nonneg*: $i < k \Longrightarrow x \geq x_1 \Longrightarrow bs\ !\ i * x + (hs\ !\ i)\ x \geq 0$
by (*drule* (*1*) *step-pos*) *simp*

lemma *step-nonneg'*: $i < k \Longrightarrow x \geq x_1 \Longrightarrow bs\ !\ i + (hs\ !\ i)\ x\ /\ x \geq 0$
by (*frule* (*1*) *step-nonneg*, *insert x0-pos x0-le-x1*) (*simp-all add*: *field-simps*)

lemma *hb-nonneg*: $hb \geq 0$
proof−
from *k-not-0* **and** *length-hs* **have** $hs \neq []$ **by** *auto*
then obtain h **where** h: $h \in set\ hs$ **by** (*cases hs*) *auto*
have $0 \leq |h\ x_1|$ **by** *simp*
also from h **have** $|h\ x_1| \leq hb * x_1\ /\ ln\ x_1\ powr\ (1{+}e)$ **by** (*intro h-bounds*) *simp-all*
finally have $0 \leq hb * x_1\ /\ ln\ x_1\ powr\ (1 + e)$ **.**
hence $0 \leq ... * (ln\ x_1\ powr\ (1 + e)\ /\ x_1)$
by (*rule mult-nonneg-nonneg*) (*intro divide-nonneg-nonneg*, *insert x1-pos*, *simp-all*)
also have $... = hb$ **using** *x1-gt-1* **by** (*simp add*: *field-simps*)
finally show *?thesis* **.**
qed

lemma *x0-hb-bound3*:
assumes $x \geq x_1$ $i < k$
shows $x - (bs\ !\ i * x + (hs\ !\ i)\ x) \leq x$
proof−
have $-(hs\ !\ i)\ x \leq |(hs\ !\ i)\ x|$ **by** *simp*
also from *assms* **have** $... \leq hb * x\ /\ ln\ x\ powr\ (1 + e)$ **by** (*simp add*: *h-bounds*)
also have $... = x * (hb\ /\ ln\ x\ powr\ (1 + e))$ **by** *simp*
also from *assms x0-pos x0-le-x1* **have** $... < x * bs\ !\ i$
by (*intro mult-strict-left-mono x0-hb-bound0'*) *simp-all*
finally show *?thesis* **by** (*simp add*: *algebra-simps*)
qed

lemma *x0-hb-bound4*:
assumes $x \geq x_1$ $i < k$
shows $(bs\ !\ i + (hs\ !\ i)\ x\ /\ x) > bs\ !\ i\ /\ 2$
proof−
from *assms x0-le-x1* **have** $hb\ /\ ln\ x\ powr\ (1 + e) < bs\ !\ i\ /\ 2$ **by** (*intro x0-hb-bound0*) *simp-all*
with *assms x0-pos x0-le-x1* **have** $(-bs\ !\ i\ /\ 2) * x < (-hb\ /\ ln\ x\ powr\ (1 + e)) * x$
by (*intro mult-strict-right-mono*) *simp-all*
also from *assms x0-pos* **have** $... \leq -|(hs\ !\ i)\ x|$ **using** *h-bounds* **by** *simp*
also have $... \leq (hs\ !\ i)\ x$ **by** *simp*
finally show *?thesis* **using** *assms x1-pos* **by** (*simp add*: *field-simps*)
qed

lemma *x0-hb-bound4'*: $x \geq x_1 \Longrightarrow i < k \Longrightarrow (bs \ ! \ i + (hs \ ! \ i) \ x \ / \ x) \geq bs \ ! \ i \ / \ 2$
by (*drule* (*1*) *x0-hb-bound4*) *simp*

lemma *x0-hb-bound5*:
assumes $x \geq x_1$ $i < k$
shows $(bs \ ! \ i + (hs \ ! \ i) \ x \ / \ x) \leq bs \ ! \ i * 3/2$
proof−
have $(hs \ ! \ i) \ x \leq |(hs \ ! \ i) \ x|$ **by** *simp*
also from *assms* **have** ... $\leq hb * x \ / \ ln \ x \ powr \ (1 + e)$ **by** (*simp add*: *h-bounds*)
also have ... $= x * (hb \ / \ ln \ x \ powr \ (1 + e))$ **by** *simp*
also from *assms x0-pos x0-le-x1* **have** ... $< x * (bs \ ! \ i \ / \ 2)$
by (*intro mult-strict-left-mono x0-hb-bound0*) *simp-all*
finally show *?thesis* **using** *assms x1-pos* **by** (*simp add*: *field-simps*)
qed

lemma *x0-hb-bound6*:
assumes $x \geq x_1$ $i < k$
shows $x * ((1 - bs \ ! \ i) \ / \ 2) \leq x - (bs \ ! \ i * x + (hs \ ! \ i) \ x)$
proof−
from *assms x0-le-x1* **have** $hb \ / \ ln \ x \ powr \ (1 + e) < (1 - bs \ ! \ i) \ / \ 2$ **using** *x0-hb-bound1* **by** *simp*
with *assms x1-pos* **have** $x * ((1 - bs \ ! \ i) \ / \ 2) \leq x * (1 - (bs \ ! \ i + hb \ / \ ln \ x \ powr \ (1 + e)))$
by (*intro mult-left-mono*) (*simp-all add*: *field-simps*)
also have ... $= x - bs \ ! \ i * x + -hb * x \ / \ ln \ x \ powr \ (1 + e)$ **by** (*simp add*: *algebra-simps*)
also from *h-bounds assms* **have** $-hb * x \ / \ ln \ x \ powr \ (1 + e) \leq -|(hs \ ! \ i) \ x|$
by (*simp add*: *length-hs*)
also have ... $\leq -(hs \ ! \ i) \ x$ **by** *simp*
finally show *?thesis* **by** (*simp add*: *algebra-simps*)
qed

lemma *x0-hb-bound7*:
assumes $x \geq x_1$ $i < k$
shows $bs!i*x + (hs!i) \ x > x_0$
proof−
from *assms x0-le-x1* **have** x': $x \geq x_0$ **by** *simp*
from *x1-ge assms* **have** $2 * x_0 * inverse \ (bs!i) \leq x_1$ **by** *simp*
with *assms b-pos* **have** $x_0 \leq x_1 * (bs!i \ / \ 2)$ **by** (*simp add*: *field-simps*)
also from *assms* x' **have** $bs!i/2 < bs!i + (hs!i) \ x \ / \ x$ **by** (*intro x0-hb-bound4*)
also from *assms step-nonneg'* x' **have** $x_1 * ... \leq x * ...$ **by** (*intro mult-right-mono*) (*simp-all*)
also from *assms x1-pos* **have** $x * (bs!i + (hs!i) \ x \ / \ x) = bs!i*x + (hs!i) \ x$
by (*simp add*: *field-simps*)
finally show *?thesis* **using** *x1-pos* **by** *simp*
qed

lemma *x0-hb-bound7'*: $x \geq x_1 \Longrightarrow i < k \Longrightarrow bs!i*x + (hs!i) \ x > 1$

by (*rule le-less-trans*[*OF - x0-hb-bound7*]) (*insert x0-le-x1 x0-ge-1*, *simp-all*)

lemma *x0-hb-bound8*:
assumes $x \geq x_1$ $i < k$
shows $bs!i*x - hb * x \,/\, ln\ x\ powr\ (1+e) > x_0$
proof−
from *assms* **have** $2 * x_0 * inverse\ (bs!i) \leq x_1$ **by** (*intro x1-ge*) *simp-all*
with *b-pos assms* **have** $x_0 \leq x_1 * (bs!i/2)$ **by** (*simp add*: *field-simps*)
also from *assms b-pos* **have** ... $\leq x * (bs!i/2)$ **by** *simp*
also from *assms x0-le-x1* **have** $hb \,/\, ln\ x\ powr\ (1+e) < bs!i/2$ **by** (*intro x0-hb-bound0*) *simp-all*
with *assms* **have** $bs!i/2 < bs!i - hb \,/\, ln\ x\ powr\ (1+e)$ **by** (*simp add*: *field-simps*)
also have $x *$ ... $= bs!i*x - hb * x \,/\, ln\ x\ powr\ (1+e)$ **by** (*simp add*: *algebra-simps*)
finally show *?thesis* **using** *assms x1-pos* **by** (*simp add*: *field-simps*)
qed

lemma *x0-hb-bound8'*:
assumes $x \geq x_1$ $i < k$
shows $bs!i*x + hb * x \,/\, ln\ x\ powr\ (1+e) > x_0$
proof−
from *assms* **have** $x_0 < bs!i*x - hb * x \,/\, ln\ x\ powr\ (1+e)$ **by** (*rule x0-hb-bound8*)
also from *assms hb-nonneg x1-pos* **have** $hb * x \,/\, ln\ x\ powr\ (1+e) \geq 0$
by (*intro mult-nonneg-nonneg divide-nonneg-nonneg*) *simp-all*
hence $bs!i*x - hb * x \,/\, ln\ x\ powr\ (1+e) \leq bs!i*x + hb * x \,/\, ln\ x\ powr\ (1+e)$ **by** *simp*
finally show *?thesis* **.**
qed

lemma
assumes $x \geq x_0$
shows *asymptotics1*: $i < k \Longrightarrow 1 + ln\ x\ powr\ (-\ e \,/\, 2) \leq$
$(1 - hb * inverse\ (bs!i) * ln\ x\ powr\ -(1+e))\ powr\ p\ *$
$(1 + ln\ (bs!i*x + hb*x/ln\ x\ powr\ (1+e))\ powr\ (-e/2))$
and *asymptotics2*: $i < k \Longrightarrow 1 - ln\ x\ powr\ (-\ e \,/\, 2) \geq$
$(1 + hb * inverse\ (bs!i) * ln\ x\ powr\ -(1+e))\ powr\ p\ *$
$(1 - ln\ (bs!i*x + hb*x/ln\ x\ powr\ (1+e))\ powr\ (-e/2))$
and *asymptotics1'*: $i < k \Longrightarrow 1 + ln\ x\ powr\ (-\ e \,/\, 2) \leq$
$(1 + hb * inverse\ (bs!i) * ln\ x\ powr\ -(1+e))\ powr\ p\ *$
$(1 + ln\ (bs!i*x + hb*x/ln\ x\ powr\ (1+e))\ powr\ (-e/2))$
and *asymptotics2'*: $i < k \Longrightarrow 1 - ln\ x\ powr\ (-\ e \,/\, 2) \geq$
$(1 - hb * inverse\ (bs!i) * ln\ x\ powr\ -(1+e))\ powr\ p\ *$
$(1 - ln\ (bs!i*x + hb*x/ln\ x\ powr\ (1+e))\ powr\ (-e/2))$
and *asymptotics3*: $(1 + ln\ x\ powr\ (-\ e \,/\, 2)) \,/\, 2 \leq 1$
and *asymptotics4*: $(1 - ln\ x\ powr\ (-\ e \,/\, 2)) * 2 \geq 1$
and *asymptotics5*: $i < k \Longrightarrow ln\ (bs!i*x - hb*x*ln\ x\ powr\ -(1+e))\ powr\ (-e/2) < 1$
apply −
using *assms asymptotics*[*of x bs*!*i*] **unfolding** *akra-bazzi-asymptotic-defs*
apply *simp-all*[*4*]

using *assms asymptotics*[*of x bs*!*0*] **unfolding** *akra-bazzi-asymptotic-defs*
apply *simp-all*[*2*]
using *assms asymptotics*[*of x bs*!*i*] **unfolding** *akra-bazzi-asymptotic-defs*
apply *simp-all*
done

lemma *x0-hb-bound9*:
assumes $x \geq x_1$ $i < k$
shows *ln* (*bs*!*i*∗*x* + (*hs*!*i*) *x*) *powr* −(*e*/*2*) < *1*
proof−
from *b-pos assms* **have** *0* < *bs*!*i*/*2* **by** *simp*
also from *assms x0-le-x1* **have** ... < *bs*!*i* + (*hs*!*i*) *x* / *x* **by** (*intro x0-hb-bound4*) *simp-all*
also from *assms x1-pos* **have** *x* ∗ ... = *bs*!*i*∗*x* + (*hs*!*i*) *x* **by** (*simp add*: *field-simps*)
finally have *pos*: *bs*!*i*∗*x* + (*hs*!*i*) *x* > *0* **using** *assms x1-pos* **by** *simp*
from *x0-hb-bound8*[*OF assms*] *x0-ge-1* **have** *pos′*: *bs*!*i*∗*x* − *hb* ∗ *x* / *ln x powr* (*1*+*e*) > *1* **by** *simp*

from *assms* **have** −(*hb* ∗ *x* / *ln x powr* (*1*+*e*)) ≤ −|(*hs*!*i*) *x*|
by (*intro le-imp-neg-le h-bounds*) *simp-all*
also have ... ≤ (*hs*!*i*) *x* **by** *simp*
finally have *ln* (*bs*!*i*∗*x* − *hb* ∗ *x* / *ln x powr* (*1*+*e*)) ≤ *ln* (*bs*!*i*∗*x* + (*hs*!*i*) *x*)
using *assms b-pos x0-pos pos′* **by** (*intro ln-mono mult-pos-pos pos*) *simp-all*
hence *ln* (*bs*!*i*∗*x* + (*hs*!*i*) *x*) *powr* −(*e*/*2*) ≤ *ln* (*bs*!*i*∗*x* − *hb* ∗ *x* / *ln x powr* (*1*+*e*)) *powr* −(*e*/*2*)
using *assms e-pos asymptotics5*[*of x*] *pos′* **by** (*intro powr-mono2′ ln-gt-zero*) *simp-all*
also have ... < *1* **using** *asymptotics5*[*of x i*] *assms x0-le-x1*
by (*subst* (*asm*) *powr-minus*) (*simp-all add*: *field-simps*)
finally show *?thesis* **.**
qed

definition *akra-bazzi-measure* :: *real* ⇒ *nat* **where**
akra-bazzi-measure x = *nat* ⌈*x*⌉

lemma *akra-bazzi-measure-decreases*:
assumes $x \geq x_1$ $i < k$
shows *akra-bazzi-measure* (*bs*!*i*∗*x* + (*hs*!*i*) *x*) < *akra-bazzi-measure x*
proof−
from *step-diff assms* **have** (*bs*!*i* ∗ *x* + (*hs*!*i*) *x*) + *1* < *x* **by** (*simp add*: *algebra-simps*)
hence ⌈(*bs*!*i* ∗ *x* + (*hs*!*i*) *x*) + *1*⌉ ≤ ⌈*x*⌉ **by** (*intro ceiling-mono*) *simp*
hence ⌈(*bs*!*i* ∗ *x* + (*hs*!*i*) *x*)⌉ < ⌈*x*⌉ **by** *simp*
with *assms x1-pos* **have** *nat* ⌈(*bs*!*i* ∗ *x* + (*hs*!*i*) *x*)⌉ < *nat* ⌈*x*⌉ **by** (*subst nat-mono-iff*) *simp-all*
thus *?thesis* **unfolding** *akra-bazzi-measure-def* **.**

qed

lemma *akra-bazzi-induct*[*consumes 1*, *case-names base rec*]:
  assumes $x \geq x_0$
  assumes *base*: $\bigwedge x.\ x \geq x_0 \Longrightarrow x \leq x_1 \Longrightarrow P\ x$
  assumes *rec*: $\bigwedge x.\ x > x_1 \Longrightarrow (\bigwedge i.\ i < k \Longrightarrow P\ (bs!i{*}x + (hs!i)\ x)) \Longrightarrow P\ x$
  shows $P\ x$
proof (*insert* ‹$x \geq x_0$›, *induction akra-bazzi-measure x arbitrary*: *x rule*: *less-induct*)
  case *less*
  show *?case*
  proof (*cases* $x \leq x_1$)
    case *True*
    with *base* **and** ‹$x \geq x_0$› **show** *?thesis* **.**
  next
    case *False*
    hence *x*: $x > x_1$ **by** *simp*
    thus *?thesis*
    proof (*rule rec*)
      fix *i* **assume** *i*: $i < k$
      from *x0-hb-bound7*[*OF - i*, *of x*] *x* **have** $bs!i{*}x + (hs!i)\ x \geq x_0$ **by** *simp*
      with *i x* **show** $P\ (bs\ !\ i * x + (hs\ !\ i)\ x)$
        by (*intro less akra-bazzi-measure-decreases*) *simp-all*
    qed
  qed
qed

end

locale *akra-bazzi-real* = *akra-bazzi-real-recursion* +
  fixes *integrable integral*
  assumes *integral*: *akra-bazzi-integral integrable integral*
  fixes f :: *real* ⇒ *real*
  and g :: *real* ⇒ *real*
  and C :: *real*
  assumes *p-props*: $(\sum i{<}k.\ as!i * bs!i\ powr\ p) = 1$
  and *f-base*: $x \geq x_0 \Longrightarrow x \leq x_1 \Longrightarrow f\ x \geq 0$
  and *f-rec*: $x > x_1 \Longrightarrow f\ x = g\ x + (\sum i{<}k.\ as!i * f\ (bs!i * x + (hs!i)$
$x))$
  and *g-nonneg*: $x \geq x_0 \Longrightarrow g\ x \geq 0$
  and *C-bound*: $b \in set\ bs \Longrightarrow x \geq x_1 \Longrightarrow C{*}x \leq b{*}x - hb{*}x/ln\ x\ powr$
$(1{+}e)$
  and *g-integrable*: $x \geq x_0 \Longrightarrow integrable\ (\lambda u.\ g\ u\ /\ u\ powr\ (p + 1))\ x_0\ x$
begin

interpretation *akra-bazzi-integral integrable integral* **by** (*rule integral*)

lemma *akra-bazzi-integrable*:

$a \geq x_0 \Longrightarrow a \leq b \Longrightarrow$ *integrable* $(\lambda x.\ g\ x\ /\ x$ *powr* $(p + 1))\ a\ b$
by (*rule integrable-subinterval*[*OF g-integrable*, *of b*]) *simp-all*

definition *g-approx* :: *nat* ⇒ *real* ⇒ *real* **where**
g-approx i x = *x powr p* ∗ *integral* (λ*u*. *g u* / *u powr* (*p* + *1*)) (*bs*!*i* ∗ *x* + (*hs*!*i*) *x*) *x*

lemma *f-nonneg*: $x \geq x_0 \Longrightarrow f\ x \geq 0$
proof (*induction x rule*: *akra-bazzi-induct*)
case (*base x*)
with *f-base*[*of x*] **show** *?case* **by** *simp*
next
case (*rec x*)
with *x0-le-x1* **have** $g\ x \geq 0$ **by** (*intro g-nonneg*) *simp-all*
moreover **{**
fix *i* **assume** *i*: $i < k$
with *rec.IH* **have** *f* (*bs*!*i*∗*x* + (*hs*!*i*) *x*) ≥ *0* **by** *simp*
with *i* **have** *as*!*i* ∗ *f* (*bs*!*i*∗*x* + (*hs*!*i*) *x*) ≥ *0*
by (*intro mult-nonneg-nonneg*[*OF a-ge-0*]) *simp-all*
}
hence ($\sum$ *i*<*k*. *as*!*i* ∗ *f* (*bs*!*i*∗*x* + (*hs*!*i*) *x*)) ≥ *0* **by** (*intro setsum-nonneg*) *blast*
ultimately show $f\ x \geq 0$ **using** *rec.hyps* **by** (*subst f-rec*) *simp-all*
qed

definition *f-approx* :: *real* ⇒ *real* **where**
f-approx x = *x powr p* ∗ (*1* + *integral* (λ*u*. *g u* / *u powr* (*p* + *1*)) x_0 *x*)

lemma *f-approx-aux*:
assumes $x \geq x_0$
shows *1* + *integral* (λ*u*. *g u* / *u powr* (*p* + *1*)) x_0 *x* ≥ *1*
proof−
from *assms* **have** *integral* (λ*u*. *g u* / *u powr* (*p* + *1*)) x_0 *x* ≥ *0*
by (*intro integral-nonneg ballI g-nonneg divide-nonneg-nonneg g-integrable*) *simp-all*
thus *?thesis* **by** *simp*
qed

lemma *f-approx-pos*: $x \geq x_0 \Longrightarrow$ *f-approx x* > *0*
unfolding *f-approx-def* **by** (*intro mult-pos-pos*, *insert x0-pos*, *simp*, *drule f-approx-aux*, *simp*)

lemma *f-approx-nonneg*: $x \geq x_0 \Longrightarrow$ *f-approx x* ≥ *0*
using *f-approx-pos*[*of x*] **by** *simp*

lemma *f-approx-bounded-below*:
obtains *c* **where** $\bigwedge x.\ x \geq x_0 \Longrightarrow x \leq x_1 \Longrightarrow$ *f-approx x* ≥ *c* *c* > *0*
proof−

{
fix x **assume** x: $x \geq x_0$ $x \leq x_1$
with *x0-pos* **have** x *powr* $p \geq$ *min* $(x_0$ *powr* $p)$ $(x_1$ *powr* $p)$
by (*intro powr-lower-bound*) *simp-all*
with x **have** *f-approx* $x \geq$ *min* $(x_0$ *powr* $p)$ $(x_1$ *powr* $p) * 1$
unfolding *f-approx-def* **by** (*intro mult-mono f-approx-aux*) *simp-all*
}
from *this x0-pos x1-pos* **show** *?thesis* **by** (*intro that*[*of min* $(x_0$ *powr* $p)$ $(x_1$ *powr* $p)$]) *auto*
qed

lemma *asymptotics-aux*:
assumes $x \geq x_1$ $i < k$
assumes $s \equiv$ (*if* $p \geq 0$ *then* 1 *else* -1)
shows $(bs!i*x - s*hb*x*ln\ x\ powr\ -(1+e))$ *powr* $p \leq (bs!i*x + (hs!i)\ x)$ *powr* p (**is** *?thesis1*)
and $(bs!i*x + (hs!i)\ x)$ *powr* $p \leq (bs!i*x + s*hb*x*ln\ x\ powr\ -(1+e))$ *powr* p (**is** *?thesis2*)
proof−
from *assms x1-gt-1* **have** *ln-x-pos*: *ln* $x > 0$ **by** *simp*
from *assms x1-pos* **have** *x-pos*: $x > 0$ **by** *simp*
from *assms x0-le-x1* **have** $hb\ /\ ln\ x\ powr\ (1+e) < bs!i/2$ **by** (*intro x0-hb-bound0*) *simp-all*
moreover from *this hb-nonneg ln-x-pos* **have** $(bs!i - hb * ln\ x\ powr\ -(1+e)) > 0$
by (*subst powr-minus*) (*simp-all add*: *field-simps*)
ultimately have $0 < x * (bs!i - hb * ln\ x\ powr\ -(1+e))$ **using** *x-pos*
by (*subst* (*asm*) *powr-minus*, *intro mult-pos-pos*)
hence *A*: $0 < bs!i*x - hb * x * ln\ x\ powr\ -(1+e)$ **by** (*simp add*: *algebra-simps*)

from *assms* **have** $-(hb*x*ln\ x\ powr\ -(1+e)) \leq -|(hs!i)\ x|$
using *h-bounds*[*of x hs*!*i*] **by** (*subst neg-le-iff-le*, *subst powr-minus*) (*simp add*: *field-simps*)
also have ... $\leq (hs!i)\ x$ **by** *simp*
finally have *B*: $bs!i*x - hb*x*ln\ x\ powr\ -(1+e) \leq bs!i*x + (hs!i)\ x$ **by** *simp*

have $(hs!i)\ x \leq |(hs!i)\ x|$ **by** *simp*
also from *assms* **have** ... $\leq (hb*x*ln\ x\ powr\ -(1+e))$
using *h-bounds*[*of x hs*!*i*] **by** (*subst powr-minus*) (*simp-all add*: *field-simps*)
finally have *C*: $bs!i*x + hb*x*ln\ x\ powr\ -(1+e) \geq bs!i*x + (hs!i)\ x$ **by** *simp*

from *A B C* **show** *?thesis1*
by (*cases* $p \geq 0$) (*auto intro*: *powr-mono2 powr-mono2'* *simp*: *assms*(3))
from *A B C* **show** *?thesis2*
by (*cases* $p \geq 0$) (*auto intro*: *powr-mono2 powr-mono2'* *simp*: *assms*(3))
qed

lemma *asymptotics1'*:

assumes $x \geq x_1$ $i < k$
shows $(bs!i*x)$ *powr* $p * (1 + ln\ x$ *powr* $(-e/2)) \leq$
$(bs!i*x + (hs!i)\ x)$ *powr* $p * (1 + ln\ (bs!i*x + (hs!i)\ x)$ *powr* $(-e/2))$
proof−
from *assms x0-le-x1* **have** x: $x \geq x_0$ **by** *simp*
from *b-pos*[*of bs*!*i*] *assms* **have** *b-pos*: $bs!i > 0$ $bs!i \neq 0$ **by** *simp-all*
from *b-less-1*[*of bs*!*i*] *assms* **have** *b-less-1*: $bs!i < 1$ **by** *simp*
from *x1-gt-1 assms* **have** *ln-x-pos*: $ln\ x > 0$ **by** *simp*
have *mono*: $\bigwedge a\ b.\ a \leq b \Longrightarrow (bs!i*x)$ *powr* $p * a \leq (bs!i*x)$ *powr* $p * b$
by (*rule mult-left-mono*) *simp-all*

def $s \equiv$ *if* $p \geq 0$ *then* 1 *else* -1 :: *real*
have $1 + ln\ x$ *powr* $(-e/2) \leq$
$(1 - s*hb*inverse(bs!i)*ln\ x$ *powr* $-(1+e))$ *powr* $p *$
$(1 + ln\ (bs!i*x + hb * x / ln\ x$ *powr* $(1+e))$ *powr* $(-e/2))$ (**is** - $\leq$ *?A* $*$ *?B*)
using *assms x* **unfolding** *s-def* **using** *asymptotics1*[*OF x assms*(*2*)] *asymptotics1'*[*OF x assms*(*2*)]
by *simp*
also have $(bs!i*x)$ *powr* $p * ... = (bs!i*x)$ *powr* $p *$ *?A* $*$ *?B* **by** *simp*
also from *x0-hb-bound0'*[*OF x*, *of bs*!*i*] *hb-nonneg x ln-x-pos assms*
have $s*hb * ln\ x$ *powr* $-(1 + e) < bs\ !\ i$
by (*subst powr-minus*) (*simp-all add*: *field-simps s-def*)
hence $(bs!i*x)$ *powr* $p *$ *?A* $= (bs!i*x*(1 - s*hb*inverse\ (bs!i)*ln\ x$ *powr* $-(1+e)))$ *powr* p
using *b-pos assms x x0-pos b-less-1 ln-x-pos*
by (*subst powr-mult*[*symmetric*]) (*simp-all add*: *s-def field-simps*)
also have $bs!i*x*(1 - s*hb*inverse\ (bs!i)*ln\ x$ *powr* $-(1+e)) = bs!i*x - s*hb*x*ln\ x$ *powr* $-(1+e)$
using *b-pos assms* **by** (*simp add*: *algebra-simps*)
also have *?B* $= 1 + ln\ (bs!i*x + hb*x*ln\ x$ *powr* $-(1+e))$ *powr* $(-e/2)$
by (*subst powr-minus*) (*simp add*: *field-simps*)

also {
from *x assms* **have** $(bs!i*x - s*hb*x*ln\ x$ *powr* $-(1+e))$ *powr* $p \leq (bs!i*x + (hs!i)\ x)$ *powr* p
using *asymptotics-aux*(*1*)[*OF assms*(*1*,*2*) *s-def*] **by** *blast*
moreover {
have $(hs!i)\ x \leq |(hs!i)\ x|$ **by** *simp*
also from *assms* **have** $|(hs!i)\ x| \leq hb * x / ln\ x$ *powr* $(1+e)$ **by** (*intro h-bounds*) *simp-all*
finally have $(hs\ !\ i)\ x \leq hb * x * ln\ x$ *powr* $-(1 + e)$
by (*subst powr-minus*) (*simp-all add*: *field-simps*)
moreover from *x hb-nonneg x0-pos* **have** $hb * x * ln\ x$ *powr* $-(1+e) \geq 0$
by (*intro mult-nonneg-nonneg*) *simp-all*
ultimately have $1 + ln\ (bs!i*x + hb * x * ln\ x$ *powr* $-(1+e))$ *powr* $(-e/2) \leq$
$1 + ln\ (bs!i*x + (hs!i)\ x)$ *powr* $(-e/2)$ **using** *assms x e-pos b-pos x0-pos*

by (*intro add-left-mono powr-mono2′ ln-mono ln-gt-zero step-pos x0-hb-bound7′ add-pos-nonneg mult-pos-pos*) *simp-all*
}
ultimately have ($bs!i{*}x - s{*}hb{*}x{*}ln\ x\ powr\ -(1{+}e)$) *powr p* $*$ ($1 + ln\ (bs!i{*}x + hb * x * ln\ x\ powr\ -(1{+}e))\ powr\ (-e/2)$)
$\leq$ ($bs!i{*}x + (hs!i)\ x$) *powr p* $*$ ($1 + ln\ (bs!i{*}x + (hs!i)\ x)\ powr\ (-e/2)$)
by (*rule mult-mono*) *simp-all*
}
finally show *?thesis* **by** (*simp-all add*: *mono*)
qed

lemma *asymptotics2′*:
assumes $x \geq x_1$ $i < k$
shows ($bs!i{*}x + (hs!i)\ x$) *powr p* $*$ ($1 - ln\ (bs!i{*}x + (hs!i)\ x)\ powr\ (-e/2)$) $\leq$
($bs!i{*}x$) *powr p* $*$ ($1 - ln\ x\ powr\ (-e/2)$)
proof−
def $s \equiv$ *if* $p \geq 0$ *then 1 else* -1 :: *real*
from *assms x0-le-x1* **have** *x*: $x \geq x_0$ **by** *simp*
from *assms x1-gt-1* **have** *ln-x-pos*: $ln\ x > 0$ **by** *simp*
from *b-pos*[*of bs*!*i*] *assms* **have** *b-pos*: $bs!i > 0$ $bs!i \neq 0$ **by** *simp-all*
from *b-pos hb-nonneg* **have** *pos*: $1 + s * hb * (inverse\ (bs!i) * ln\ x\ powr\ -(1{+}e)) > 0$
using *x0-hb-bound0′*[*OF x*, *of bs*!*i*] *b-pos assms ln-x-pos*
by (*subst powr-minus*) (*simp add*: *field-simps s-def*)
have *mono*: $\bigwedge a\ b.\ a \leq b \Longrightarrow (bs!i{*}x)\ powr\ p * a \leq (bs!i{*}x)\ powr\ p * b$
by (*rule mult-left-mono*) *simp-all*

let $?A = (1 + s{*}hb{*}inverse(bs!i){*}ln\ x\ powr\ -(1{+}e))\ powr\ p$
let $?B = 1 - ln\ (bs!i{*}x + (hs!i)\ x)\ powr\ (-e/2)$
let $?B' = 1 - ln\ (bs!i{*}x + hb * x\ /\ ln\ x\ powr\ (1{+}e))\ powr\ (-e/2)$

from *assms x* **have** ($bs!i{*}x + (hs!i)\ x$) *powr p* $\leq$ ($bs!i{*}x + s{*}hb{*}x{*}ln\ x\ powr\ -(1{+}e)$) *powr p*
by (*intro asymptotics-aux*(*2*)) (*simp-all add*: *s-def*)
moreover from *x0-hb-bound9*[*OF assms*(*1*,*2*)] **have** $?B \geq 0$ **by** (*simp add*: *field-simps*)
ultimately have ($bs!i{*}x + (hs!i)\ x$) *powr p* $* ?B \leq$
($bs!i{*}x + s{*}hb{*}x{*}ln\ x\ powr\ -(1{+}e)$) *powr p* $* ?B$ **by** (*rule mult-right-mono*)
also from *assms e-pos pos* **have** $?B \leq ?B'$
proof−
from *x0-hb-bound8′*[*OF assms*(*1*,*2*)] *x0-hb-bound8*[*OF assms*(*1*,*2*)] *x0-ge-1*
have $bs\ !\ i * x + s{*}hb * x\ /\ ln\ x\ powr\ (1 + e) > 1$ **by** (*simp add*: *s-def*)
moreover from *this* **have** ... > 0 **by** *simp*
moreover from *x0-hb-bound7*[*OF assms*(*1*,*2*)] *x0-ge-1* **have** $bs\ !\ i * x + (hs\ !\ i)\ x > 1$ **by** *simp*
moreover {

have $(hs!i)\ x \le |(hs!i)\ x|$ **by** *simp*
also from *assms x0-le-x1* **have** ... $\le hb*x/ln\ x\ powr\ (1+e)$ **by** (*intro h-bounds*) *simp-all*
finally have $bs!i*x + (hs!i)\ x \le bs!i*x + hb*x/ln\ x\ powr\ (1+e)$ **by** *simp*
}
ultimately show $?B \le ?B'$ **using** *assms e-pos x step-pos*
by (*intro diff-left-mono powr-mono2' ln-mono ln-gt-zero*) *simp-all*
qed
hence $(bs!i*x + s*hb*x*ln\ x\ powr\ -(1+e))\ powr\ p * ?B \le$
$(bs!i*x + s*hb*x*ln\ x\ powr\ -(1+e))\ powr\ p * ?B'$ **by** (*intro mult-left-mono*) *simp-all*
also have $bs!i*x + s*hb*x*ln\ x\ powr\ -(1+e) = bs!i*x*(1 + s*hb*inverse\ (bs!i)*ln\ x\ powr\ -(1+e))$
using *b-pos* **by** (*simp-all add*: *field-simps*)
also have ... $powr\ p = (bs!i*x)\ powr\ p * ?A$
using *b-pos x x0-pos pos* **by** (*intro powr-mult*) *simp-all*
also have $(bs!i*x)\ powr\ p * ?A * ?B' = (bs!i*x)\ powr\ p * (?A * ?B')$ **by** *simp*
also have $?A * ?B' \le 1 - ln\ x\ powr\ (-e/2)$ **using** *assms x*
using *asymptotics2*[*OF x assms*(*2*)] *asymptotics2'*[*OF x assms*(*2*)] **by** (*simp add*: *s-def*)
finally show *?thesis* **by** (*simp-all add*: *mono*)
qed

lemma *Cx-le-step*:
assumes $i < k\ x \ge x_1$
shows $C*x \le bs!i*x + (hs!i)\ x$
proof−
from *assms* **have** $C*x \le bs!i*x - hb*x/ln\ x\ powr\ (1+e)$ **by** (*intro C-bound*) *simp-all*
also from *assms* **have** $-(hb*x/ln\ x\ powr\ (1+e)) \le -|(hs!i)\ x|$
by (*subst neg-le-iff-le*, *intro h-bounds*) *simp-all*
hence $bs!i*x - hb*x/ln\ x\ powr\ (1+e) \le bs!i*x + -|(hs!i)\ x|$ **by** *simp*
also have $-|(hs!i)\ x| \le (hs!i)\ x$ **by** *simp*
finally show *?thesis* **by** *simp*
qed

end

locale *akra-bazzi-nat-to-real* = *akra-bazzi-real-recursion* +
fixes $f :: nat \Rightarrow real$
and $g :: real \Rightarrow real$
assumes *f-base*: $real\ x \ge x_0 \Longrightarrow real\ x \le x_1 \Longrightarrow f\ x \ge 0$
and *f-rec*: $real\ x > x_1 \Longrightarrow$
$f\ x = g\ (real\ x) + (\sum i<k.\ as!i * f\ (nat\ \lfloor bs!i * x + (hs!i)\ (real\ x)\rfloor))$
and *x0-int*: $real\ (nat\ \lfloor x_0\rfloor) = x_0$
begin

function f' :: *real* $\Rightarrow$ *real* **where**
  $x \leq x_1 \Longrightarrow f'\ x = f\ (nat\ \lfloor x \rfloor)$
| $x > x_1 \Longrightarrow f'\ x = g\ x + (\sum i{<}k.\ as!i * f'\ (bs!i * x + (hs!i)\ x))$
by (*force*, *simp-all*)
termination by (*relation Wellfounded.measure akra-bazzi-measure*)
  (*simp-all add*: *akra-bazzi-measure-decreases*)

lemma f'-*base*: $x \geq x_0 \Longrightarrow x \leq x_1 \Longrightarrow f'\ x \geq 0$
  apply (*subst* f'*.simps*(*1*), *assumption*)
  apply (*rule f-base*)
  apply (*rule order.trans*[*of - real* (*nat* $\lfloor x_0 \rfloor$)], *simp add*: *x0-int*)
  apply (*subst of-nat-le-iff*, *intro nat-mono floor-mono*, *assumption*)
  using *x0-pos* **apply** *linarith*
  done

lemmas f'-*rec* = f'*.simps*(*2*)

end

locale *akra-bazzi-real-lower* = *akra-bazzi-real* +
  fixes *fb2 gb2 c2* :: *real*
  assumes *f-base2*: $x \geq x_0 \Longrightarrow x \leq x_1 \Longrightarrow f\ x \geq fb2$
  and *fb2-pos*: $fb2 > 0$
  and *g-growth2*: $\forall x{\geq}x_1.\ \forall u{\in}\{C{*}x..x\}.\ c2 * g\ x \geq g\ u$
  and *c2-pos*: $c2 > 0$
  and *g-bounded*: $x \geq x_0 \Longrightarrow x \leq x_1 \Longrightarrow g\ x \leq gb2$
begin

interpretation *akra-bazzi-integral integrable integral* **by** (*rule integral*)

lemma *gb2-nonneg*: $gb2 \geq 0$ **using** *g-bounded*[*of* x_0] *x0-le-x1 x0-pos g-nonneg*[*of* x_0] **by** *simp*

lemma *g-growth2'*:
  assumes $x \geq x_1$ $i < k$ $u \in \{bs!i{*}x{+}(hs!i)\ x..x\}$
  shows $c2 * g\ x \geq g\ u$
proof−
  from *assms* **have** $C{*}x \leq bs!i{*}x{+}(hs!i)\ x$ **by** (*intro Cx-le-step*)
  with *assms* **have** $u \in \{C{*}x..x\}$ **by** *auto*
  with *assms g-growth2* **show** *?thesis* **by** *simp*
qed

lemma *g-bounds2*:
  obtains *c4* **where** $\bigwedge x\ i.\ x \geq x_1 \Longrightarrow i < k \Longrightarrow$ *g-approx* $i\ x \leq c4 * g\ x$ $c4 > 0$
proof−

  def $c4 \equiv Max\ \{c2\ /\ min\ 1\ (min\ ((b/2)\ powr\ (p{+}1))\ ((b{*}3/2)\ powr\ (p{+}1)))$
$|b.\ b \in set\ bs\}$

{
  from *bs-nonempty* **obtain** *b* **where** *b*: $b \in set\ bs$ **by** (*cases bs*) *auto*
  let *?m* = *min 1* (*min* ((*b/2*) *powr* (*p+1*)) ((*b***3/2*) *powr* (*p+1*)))
  from *b b-pos* **have** *?m* > *0* **unfolding** *min-def* **by** (*auto simp*: *not-le*)
  with *b b-pos c2-pos* **have** *c2* / *?m* > *0* **by** (*simp-all add*: *field-simps*)
  with *b* **have** *c4* > *0* **unfolding** *c4-def* **by** (*subst Max-gr-iff*) (*simp*, *simp*, *blast*)
}

{
  fix *x i* **assume** *i*: $i < k$ **and** *x*: $x \geq x_1$
  let *?m* = *min 1* (*min* ((*bs*!*i/2*) *powr* (*p+1*)) ((*bs*!*i***3/2*) *powr* (*p+1*)))
  have *min 1* ((*bs*!*i* + (*hs* ! *i*) *x* / *x*) *powr* (*p+1*)) ≥ *min 1* (*min* ((*bs*!*i/2*) *powr* (*p+1*)) ((*bs*!*i***3/2*) *powr* (*p+1*)))
    apply (*insert x i x0-le-x1 x1-pos step-pos b-pos*[*OF b-in-bs*[*OF i*]],
        rule min.mono, *simp*, *cases p* + *1* ≥ *0*)
    apply (*rule order.trans*[*OF min.cobounded1 powr-mono2*[*OF - - x0-hb-bound4'*]], *simp-all add*: *field-simps*) []
    apply (*rule order.trans*[*OF min.cobounded2 powr-mono2'*[*OF - - x0-hb-bound5*]], *simp-all add*: *field-simps*) []
    done
  with *i b-pos*[*of bs*!*i*] **have** *c2* / *min 1* ((*bs*!*i* + (*hs* ! *i*) *x* / *x*) *powr* (*p+1*)) ≤ *c2* / *?m* **using** *c2-pos*
    unfolding *min-def* **by** (*intro divide-left-mono*) (*auto intro*!: *mult-pos-pos dest*!: *powr-negD*)

  also from *i x* **have** ... ≤ *c4* **unfolding** *c4-def* **by** (*intro Max.coboundedI*) *auto*
  finally have *c2* / *min 1* ((*bs*!*i* + (*hs* ! *i*) *x* / *x*) *powr* (*p+1*)) ≤ *c4* **.**
} **note** *c4* = *this*

{
  fix *x* :: *real* **and** *i* :: *nat*
  assume *x*: $x \geq x_1$ **and** *i*: $i < k$
  from *x x1-pos* **have** *x-pos*: *x* > *0* **by** *simp*
  let *?x'* = *bs* ! *i* * *x* + (*hs* ! *i*) *x*
  let *?x''* = *bs* ! *i* + (*hs* ! *i*) *x* / *x*
  from *x x1-ge-1 i g-growth2' x0-le-x1 c2-pos*
    have *c2*: *c2* > *0* ∀ *u*∈{*?x'*..*x*}. *g u* ≤ *c2* * *g x* **by** *auto*

  from *x0-le-x1 x i* **have** *x'-le-x*: *?x'* ≤ *x* **by** (*intro step-le-x*) *simp-all*
  let *?m* = *min* (*?x' powr* (*p* + *1*)) (*x powr* (*p* + *1*))
  def *m'* ≡ *min 1* (*?x'' powr* (*p* + *1*))
  have [*simp*]: *bs* ! *i* > *0* **by** (*intro b-pos nth-mem*) (*simp add*: *i length-bs*)
  from *x0-le-x1 x i* **have** [*simp*]: *?x'* > *0* **by** (*intro step-pos*) *simp-all*

  {
    fix *u* **assume** *u*: *u* ≥ *?x'* *u* ≤ *x*

have *?m* $\leq$ *u powr* (*p* + *1*) **using** *x u* **by** (*intro powr-lower-bound mult-pos-pos*) *simp-all*
moreover from *c2* **and** *u* **have** *g u* $\leq$ *c2* $*$ *g x* **by** *simp*
ultimately have *g u* $*$ *?m* $\leq$ *c2* $*$ *g x* $*$ *u powr* (*p* + *1*) **using** *c2 x x1-pos x0-le-x1*
by (*intro mult-mono mult-nonneg-nonneg g-nonneg*) *auto*
}
hence *integral* (λ*u*. *g u* / *u powr* (*p*+*1*)) *?x′ x* $\leq$ *integral* (λ*u*. *c2* $*$ *g x* / *?m*) *?x′ x*
using *x-pos step-pos*[*OF i x*] *x0-hb-bound7*[*OF x i*] *c2 x x0-le-x1*
by (*intro integral-le x′-le-x akra-bazzi-integrable ballI integrable-const*)
(*auto simp*: *field-simps intro*!: *mult-nonneg-nonneg g-nonneg*)

also from *x0-pos x x0-le-x1 x′-le-x c2* **have** ... = (*x* − *?x′*) $*$ (*c2* $*$ *g x* / *?m*)
by (*subst integral-const*) (*simp-all add*: *g-nonneg*)
also from *c2 x-pos x x0-le-x1* **have** *c2* $*$ *g x* $\geq$ *0*
by (*intro mult-nonneg-nonneg g-nonneg*) *simp-all*
with *x i x0-le-x1* **have** (*x* − *?x′*) $*$ (*c2* $*$ *g x* / *?m*) $\leq$ *x* $*$ (*c2* $*$ *g x* / *?m*)
by (*intro x0-hb-bound3 mult-right-mono*) (*simp-all add*: *field-simps*)

also have *x powr* (*p* + *1*) = *x powr* (*p* + *1*) $*$ *1* **by** *simp*
also have (*bs* ! *i* $*$ *x* + (*hs* ! *i*) *x*) *powr* (*p* + *1*) =
(*bs* ! *i* + (*hs* ! *i*) *x* / *x*) *powr* (*p* + *1*) $*$ *x powr* (*p* + *1*)
using *x x1-pos step-pos*[*OF i x*] *x-pos i x0-le-x1*
by (*subst powr-mult*[*symmetric*]) (*simp add*: *field-simps*, *simp*, *simp add*: *algebra-simps*)
also have ... = *x powr* (*p* + *1*) $*$ (*bs* ! *i* + (*hs* ! *i*) *x* / *x*) *powr* (*p* + *1*) **by** *simp*
also have *min* ... (*x powr* (*p* + *1*) $*$ *1*) = *x powr* (*p* + *1*) $*$ *m′* **unfolding** *m′-def* **using** *x-pos*
by (*subst min.commute*, *intro min-mult-left*[*symmetric*]) *simp*

also from *x-pos* **have** *x* $*$ (*c2* $*$ *g x* / (*x powr* (*p* + *1*) $*$ *m′*)) = (*c2*/*m′*) $*$ (*g x* / *x powr p*)
by (*simp add*: *field-simps powr-add*)
also from *x i g-nonneg x0-le-x1 x1-pos* **have** ... $\leq$ *c4* $*$ (*g x* / *x powr p*) **unfolding** *m′-def*
by (*intro mult-right-mono c4*) (*simp-all add*: *field-simps*)
finally have *g-approx i x* $\leq$ *c4* $*$ *g x*
unfolding *g-approx-def* **using** *x-pos* **by** (*simp add*: *field-simps*)
}
thus *?thesis* **using** *that* ‹*c4* > *0*› **by** *blast*
qed

lemma *f-approx-bounded-above*:
obtains *c* **where** $\bigwedge$*x*. $x \geq x_0 \Longrightarrow x \leq x_1 \Longrightarrow$ *f-approx x* $\leq$ *c c* > *0*
proof−
let *?m1* = *max* (x_0 *powr p*) (x_1 *powr p*)
let *?m2* = *max* (x_0 *powr* (−(*p*+*1*))) (x_1 *powr* (−(*p*+*1*)))

let *?m3* = *gb2* ∗ *?m2*
let *?m4* = *1* + $(x_1 - x_0)$ ∗ *?m3*
let *?int* = λ*x*. *integral* (λ*u*. *g u* / *u powr* (*p* + *1*)) x_0 *x*
{
fix *x* **assume** *x*: $x \geq x_0$ $x \leq x_1$
with *x0-pos* **have** *x powr p* ≤ *?m1 ?m1* ≥ *0* **by** (*intro powr-upper-bound*) (*simp-all add*: *max-def*)
moreover {
fix *u* **assume** *u*: $u \in \{x_0..x\}$
have *g u* / *u powr* (*p* + *1*) = *g u* ∗ *u powr* (−(*p*+*1*))
by (*subst powr-minus*) (*simp add*: *field-simps*)
also from *u x x0-pos* **have** *u powr* (−(*p*+*1*)) ≤ *?m2*
by (*intro powr-upper-bound*) *simp-all*
hence *g u* ∗ *u powr* (−(*p*+*1*)) ≤ *g u* ∗ *?m2*
using *u g-nonneg x0-pos* **by** (*intro mult-left-mono*) *simp-all*
also from *x u x0-pos* **have** *g u* ≤ *gb2* **by** (*intro g-bounded*) *simp-all*
hence *g u* ∗ *?m2* ≤ *gb2* ∗ *?m2* **by** (*intro mult-right-mono*) (*simp-all add*: *max-def*)
finally have *g u* / *u powr* (*p* + *1*) ≤ *?m3* **.**
} **note** *A* = *this*
{
from *A x gb2-nonneg* **have** *?int x* ≤ *integral* (λ-. *?m3*) x_0 *x*
by (*intro integral-le akra-bazzi-integrable integrable-const mult-nonneg-nonneg*) (*simp-all add*: *le-max-iff-disj*)
also from *x gb2-nonneg* **have** ... ≤ $(x - x_0)$ ∗ *?m3*
by (*subst integral-const*) (*simp-all add*: *le-max-iff-disj*)
also from *x gb2-nonneg* **have** ... ≤ $(x_1 - x_0)$ ∗ *?m3*
by (*intro mult-right-mono mult-nonneg-nonneg*) (*simp-all add*: *max-def*)
finally have *1* + *?int x* ≤ *?m4* **by** *simp*
}
moreover from *x g-nonneg x0-pos* **have** *?int x* ≥ *0*
by (*intro integral-nonneg akra-bazzi-integrable*) (*simp-all add*: *powr-def field-simps*)
hence *1* + *?int x* ≥ *0* **by** *simp*
ultimately have *f-approx x* ≤ *?m1* ∗ *?m4*
unfolding *f-approx-def* **by** (*intro mult-mono*)
hence *f-approx x* ≤ *max 1* (*?m1* ∗ *?m4*) **by** *simp*
}
from *that*[*OF this*] **show** *?thesis* **by** *auto*
qed

lemma *f-bounded-below*:
assumes *c'*: *c'* > *0*
obtains *c* **where** $\bigwedge x.\ x \geq x_0 \Longrightarrow x \leq x_1 \Longrightarrow$ *2* ∗ (*c* ∗ *f-approx x*) ≤ *f x c* ≤ *c'* *c* > *0*
proof−
obtain *c* **where** *c*: $\bigwedge x.\ x_0 \leq x \Longrightarrow x \leq x_1 \Longrightarrow$ *f-approx x* ≤ *c c* > *0*
by (*rule f-approx-bounded-above*) *blast*
{
fix *x* **assume** *x*: $x_0 \leq x$ $x \leq x_1$

with *c* **have** *inverse c* $*$ *f-approx x* $\leq$ *1* **by** (*simp add*: *field-simps*)
moreover from *x f-base2 x0-pos* **have** *f x* $\geq$ *fb2* **by** *auto*
ultimately have *inverse c* $*$ *f-approx x* $*$ *fb2* $\leq$ *1* $*$ *f x* **using** *fb2-pos*
by (*intro mult-mono*) *simp-all*
hence *inverse c* $*$ *fb2* $*$ *f-approx x* $\leq$ *f x* **by** (*simp add*: *field-simps*)
moreover have *min c'* (*inverse c* $*$ *fb2*) $*$ *f-approx x* $\leq$ *inverse c* $*$ *fb2* $*$ *f-approx x*
using *f-approx-nonneg x c*
by (*intro mult-right-mono f-approx-nonneg*) (*simp-all add*: *field-simps*)
ultimately have *2* $*$ (*min c'* (*inverse c* $*$ *fb2*) / *2* $*$ *f-approx x*) $\leq$ *f x* **by** *simp*
}
moreover from *c'* **have** *min c'* (*inverse c* $*$ *fb2*) / *2* $\leq$ *c'* **by** *simp*
moreover have *min c'* (*inverse c* $*$ *fb2*) / *2* $>$ *0*
using *c fb2-pos c'* **by** *simp*
ultimately show *?thesis* **by** (*rule that*)
qed

lemma *akra-bazzi-lower*:
obtains *c5* **where** $\bigwedge x.\ x \geq x_0 \Longrightarrow$ *f x* $\geq$ *c5* $*$ *f-approx x c5* $>$ *0*
proof$-$
obtain *c4* **where** *c4*: $\bigwedge x\ i.\ x \geq x_1 \Longrightarrow i < k \Longrightarrow$ *g-approx i x* $\leq$ *c4* $*$ *g x c4* $>$ *0*
by (*rule g-bounds2*) *blast*
hence *inverse c4* / *2* $>$ *0* **by** *simp*
then obtain *c5* **where** *c5*: $\bigwedge x.\ x \geq x_0 \Longrightarrow x \leq x_1 \Longrightarrow$ *2* $*$ (*c5* $*$ *f-approx x*) $\leq$ *f x*
c5 $\leq$ *inverse c4* / *2 c5* $>$ *0*
by (*rule f-bounded-below*) *blast*

{
fix *x* :: *real* **assume** *x*: $x \geq x_0$
from *c5 x* **have** *c5* $*$ *1* $*$ *f-approx x* $\leq$ *c5* $*$ (*1* + *ln x powr* ($-$ *e* / *2*)) $*$ *f-approx x*
by (*intro mult-right-mono mult-left-mono f-approx-nonneg*) *simp-all*
also from *x* **have** *c5* $*$ (*1* + *ln x powr* ($-e/2$)) $*$ *f-approx x* $\leq$ *f x*
proof (*induction x rule*: *akra-bazzi-induct*)
case (*base x*)
have *1* + *ln x powr* ($-e/2$) $\leq$ *2* **using** *asymptotics3 base* **by** *simp*
hence (*1* + *ln x powr* ($-e/2$)) $*$ (*c5* $*$ *f-approx x*) $\leq$ *2* $*$ (*c5* $*$ *f-approx x*)
using *c5 f-approx-nonneg base x0-ge-1* **by** (*intro mult-right-mono mult-nonneg-nonneg*) *simp-all*
also from *base* **have** *2* $*$ (*c5* $*$ *f-approx x*) $\leq$ *f x* **by** (*intro c5*) *simp-all*
finally show *?case* **by** (*simp add*: *algebra-simps*)
next
case (*rec x*)
let *?a* = $\lambda i.$ *as*!*i* **and** *?b* = $\lambda i.$ *bs*!*i* **and** *?h* = $\lambda i.$ *hs*!*i*
let *?int* = *integral* ($\lambda u.$ *g u* / *u powr* (*p*+*1*)) x_0 *x*
let *?int1* = $\lambda i.$ *integral* ($\lambda u.$ *g u* / *u powr* (*p*+*1*)) x_0 (*?b i*$*$*x*+*?h i x*)

let *?int2* = λ*i*. *integral* (λ*u*. *g u* / *u powr* (*p*+*1*)) (*?b i*∗*x*+*?h i x*) *x*
let *?l* = *ln x powr* (−*e*/*2*) **and** *?l'* = λ*i*. *ln* (*?b i*∗*x* + *?h i x*) *powr* (−*e*/*2*)

from *rec* **and** *x0-le-x1 x0-ge-1* **have** *x*: *x* $\geq$ x_0 **and** *x-gt-1*: *x* > *1* **by** *simp-all*
with *x0-pos* **have** *x-pos*: *x* > *0* **and** *x-nonneg*: *x* $\geq$ *0* **by** *simp-all*
from *c5 c4* **have** *c5* ∗ *c4* $\leq$ *1*/*2* **by** (*simp add*: *field-simps*)
moreover from *asymptotics3 x* **have** (*1* + *?l*) $\leq$ *2* **by** (*simp add*: *field-simps*)
ultimately have (*c5*∗*c4*)∗(*1* + *?l*) $\leq$ (*1*/*2*) ∗ *2* **by** (*rule mult-mono*) *simp-all*
hence *0* $\leq$ *1* − *c5*∗*c4*∗(*1* + *?l*) **by** *simp*
with *g-nonneg*[*OF x*] **have** *0* $\leq$ *g x* ∗ ... **by** (*intro mult-nonneg-nonneg*) *simp-all*
hence *c5* ∗ (*1* + *?l*) ∗ *f-approx x* $\leq$ *c5* ∗ (*1* + *?l*) ∗ *f-approx x* + *g x* − *c5*∗*c4*∗(*1* + *?l*) ∗ *g x*
by (*simp add*: *algebra-simps*)
also from *x-gt-1* **have** ... = *c5* ∗ *x powr p* ∗ (*1* + *?l*) ∗ (*1* + *?int* − *c4*∗*g x*/*x powr p*) + *g x*
by (*simp add*: *field-simps f-approx-def powr-minus*)
also have *c5* ∗ *x powr p* ∗ (*1* + *?l*) ∗ (*1* + *?int* − *c4*∗*g x*/*x powr p*) =
($\sum$ *i*<*k*. (*?a i* ∗ *?b i powr p*) ∗ (*c5* ∗ *x powr p* ∗ (*1* + *?l*) ∗ (*1* + *?int* − *c4*∗*g x*/*x powr p*)))
by (*subst setsum-left-distrib*[*symmetric*]) (*simp add*: *p-props*)
also have ... $\leq$ ($\sum$ *i*<*k*. *?a i* ∗ *f* (*?b i*∗*x* + *?h i x*))
proof (*intro setsum-mono*, *clarify*)
fix *i* **assume** *i*: *i* < *k*
let *?f* = *c5* ∗ *?a i* ∗ (*?b i* ∗ *x*) *powr p*
from *rec.hyps i* **have** x_0 < *bs* ! *i* ∗ *x* + (*hs* ! *i*) *x* **by** (*intro x0-hb-bound7*) *simp-all*
hence *1* + *?int1 i* $\geq$ *1* **by** (*intro f-approx-aux x0-hb-bound7*) *simp-all*
hence *int-nonneg*: *1* + *?int1 i* $\geq$ *0* **by** *simp*

have (*?a i* ∗ *?b i powr p*) ∗ (*c5* ∗ *x powr p* ∗ (*1* + *?l*) ∗ (*1* + *?int* − *c4*∗*g x*/*x powr p*)) =
?f ∗ (*1* + *?l*) ∗ (*1* + *?int* − *c4*∗*g x*/*x powr p*) (**is** *?expr* = *?A* ∗ *?B*)
using *x-pos b-pos*[*of bs*!*i*] *i* **by** (*subst powr-mult*) *simp-all*
also from *rec.hyps i* **have** *g-approx i x* $\leq$ *c4* ∗ *g x* **by** (*intro c4*) *simp-all*
hence *c4*∗*g x*/*x powr p* $\geq$ *?int2 i* **unfolding** *g-approx-def* **using** *x-pos*
by (*simp add*: *field-simps*)
hence *?A* ∗ *?B* $\leq$ *?A* ∗ (*1* + (*?int* − *?int2 i*)) **using** *i c5 a-ge-0*
by (*intro mult-left-mono mult-nonneg-nonneg*) *simp-all*
also from *rec.hyps i* **have** x_0 < *bs* ! *i* ∗ *x* + (*hs* ! *i*) *x* **by** (*intro x0-hb-bound7*) *simp-all*
hence *?int* − *?int2 i* = *?int1 i*
apply (*subst diff-eq-eq*, *subst eq-commute*)
apply (*intro integral-combine akra-bazzi-integrable*)
apply (*insert rec.hyps step-le-x*[*OF i*, *of x*], *simp-all*)
done
also have *?A* ∗ (*1* + *?int1 i*) = (*c5*∗*?a i*∗(*1* + *?int1 i*)) ∗ ((*?b i*∗*x*) *powr p* ∗ (*1* + *?l*))
by (*simp add*: *algebra-simps*)
also have ... $\leq$ (*c5*∗*?a i*∗(*1* + *?int1 i*)) ∗ ((*?b i*∗*x* + *?h i x*) *powr p* ∗ (*1* +

```
?l′ i))
        using rec.hyps i c5 a-ge-0 int-nonneg
        by (intro mult-left-mono asymptotics1′ mult-nonneg-nonneg) simp-all
      also have ... = ?a i∗(c5∗(1 + ?l′ i)∗f-approx (?b i∗x + ?h i x))
        by (simp add: algebra-simps f-approx-def)
      also from i have ... ≤ ?a i ∗ f (?b i∗x + ?h i x)
        by (intro mult-left-mono a-ge-0 rec.IH) simp-all
      finally show ?expr ≤ ?a i ∗ f (?b i∗x + ?h i x) .
    qed
    also have ... + g x = f x using f-rec[of x] rec.hyps x0-le-x1 by simp
    finally show ?case by simp
  qed
  finally have c5 ∗ f-approx x ≤ f x by simp
  }
  from this and c5(3) show ?thesis by (rule that)
qed

lemma akra-bazzi-bigomega:
  f ∈ Ω(λx. x powr p ∗ (1 + integral (λu. g u / u powr (p + 1)) x₀ x))
apply (fold f-approx-def, rule akra-bazzi-lower, erule landau-omega.bigI)
apply (subst eventually-at-top-linorder, rule exI[of - x₀])
apply (simp add: f-nonneg f-approx-nonneg)
done

end


locale akra-bazzi-real-upper = akra-bazzi-real +
  fixes fb1 c1 :: real
  assumes f-base1:   x ≥ x₀ ⟹ x ≤ x₁ ⟹ f x ≤ fb1
  and      g-growth1: ∀x≥x₁. ∀u∈{C∗x..x}. c1 ∗ g x ≤ g u
  and      c1-pos:    c1 > 0
begin

interpretation akra-bazzi-integral integrable integral by (rule integral)

lemma g-growth1′:
  assumes x ≥ x₁ i < k u ∈ {bs!i∗x+(hs!i) x..x}
  shows   c1 ∗ g x ≤ g u
proof−
  from assms have C∗x ≤ bs!i∗x+(hs!i) x by (intro Cx-le-step)
  with assms have u ∈ {C∗x..x} by auto
  with assms g-growth1 show ?thesis by simp
qed

lemma g-bounds1:
  obtains c3 where
    ⋀x i. x ≥ x₁ ⟹ i < k ⟹ c3 ∗ g x ≤ g-approx i x c3 > 0
proof−
```

```
def c3 ≡ Min {c1*((1−b)/2) / max 1 (max ((b/2) powr (p+1)) ((b*3/2) powr
(p+1))) |b. b ∈ set bs}

{
  fix b assume b: b ∈ set bs
  let ?x = max 1 (max ((b/2) powr (p+1)) ((b*3/2) powr (p+1)))
  have ?x ≥ 1 by simp
  hence ?x > 0 by (rule less-le-trans[OF zero-less-one])
  with b b-less-1 c1-pos have c1*((1−b)/2) / ?x > 0
    by (intro divide-pos-pos mult-pos-pos) (simp-all add: algebra-simps)
}
hence c3 > 0 unfolding c3-def by (subst Min-gr-iff) auto

{
  fix x i assume i: i < k and x: x ≥ x₁
  with b-less-1 have b-less-1': bs ! i < 1 by simp
  let ?m = max 1 (max ((bs!i/2) powr (p+1)) ((bs!i*3/2) powr (p+1)))
   from i x have c3 ≤ c1*((1−bs!i)/2) / ?m unfolding c3-def by (intro
Min.coboundedI) auto
  also have max 1 ((bs!i + (hs ! i) x / x) powr (p+1)) ≤ max 1 (max ((bs!i/2)
powr (p+1)) ((bs!i*3/2) powr (p+1)))
    apply (insert x i x0-le-x1 x1-pos step-pos[OF i x] b-pos[OF b-in-bs[OF i]],
          rule max.mono, simp, cases p + 1 ≥ 0)
   apply (rule order.trans[OF powr-mono2[OF - - x0-hb-bound5] max.cobounded2],
simp-all add: field-simps) []
   apply (rule order.trans[OF powr-mono2'[OF - - x0-hb-bound4'] max.cobounded1],
simp-all add: field-simps) []
    done
  with b-less-1' c1-pos have c1*((1−bs!i)/2) / ?m ≤
       c1*((1−bs!i)/2) / max 1 ((bs!i + (hs ! i) x / x) powr (p+1))
   by (intro divide-left-mono mult-nonneg-nonneg) (simp-all add: algebra-simps)
  finally have c3 ≤ c1*((1−bs!i)/2) / max 1 ((bs!i + (hs ! i) x / x) powr
(p+1)) .
} note c3 = this

{
  fix x :: real and i :: nat
  assume x: x ≥ x₁ and i: i < k
  from x x1-pos have x-pos: x > 0 by simp
  let ?x' = bs ! i * x + (hs ! i) x
  let ?x'' = bs ! i + (hs ! i) x / x
  from x x1-ge-1 x0-le-x1 i c1-pos g-growth1'
    have c1: c1 > 0 ∀u∈{?x'..x}. g u ≥ c1 * g x by auto
  def b' ≡ (1 − bs!i)/2

  from x x0-le-x1 i have x'-le-x: ?x' ≤ x by (intro step-le-x) simp-all
  let ?m = max (?x' powr (p + 1)) (x powr (p + 1))
  def m' ≡ max 1 (?x'' powr (p + 1))
  have [simp]: bs ! i > 0 by (intro b-pos nth-mem) (simp add: i length-bs)
```

from *x x0-le-x1 i* **have** *x′-pos*: *?x′* > *0* **by** (*intro step-pos*) *simp-all*
have *m-pos*: *?m* > *0* **unfolding** *max-def* **using** *x-pos step-pos*[*OF i x*] **by** *auto*
with *x x0-le-x1 c1* **have** *c1-g-m-nonneg*: *c1* ∗ *g x* / *?m* ≥ *0*
by (*intro mult-nonneg-nonneg divide-nonneg-pos g-nonneg*) *simp-all*

from *x i g-nonneg x0-le-x1* **have** *c3* ∗ (*g x* / *x powr p*) ≤ (*c1*∗*b′*/*m′*) ∗ (*g x* / *x powr p*)
unfolding *m′-def b′-def* **by** (*intro mult-right-mono c3*) (*simp-all add*: *field-simps*)
also from *x-pos* **have** ... = (*x* ∗ *b′*) ∗ (*c1* ∗ *g x* / (*x powr* (*p* + *1*) ∗ *m′*))
by (*simp add*: *field-simps powr-add*)
also from *x i c1-pos x1-pos x0-le-x1*
have ... ≤ (*x* − *?x′*) ∗ (*c1* ∗ *g x* / (*x powr* (*p* + *1*) ∗ *m′*))
unfolding *b′-def m′-def* **by** (*intro x0-hb-bound6 mult-right-mono mult-nonneg-nonneg*

divide-nonneg-nonneg g-nonneg) *simp-all*
also have *x powr* (*p* + *1*) ∗ *m′* =
max (*x powr* (*p* + *1*) ∗ (*bs* ! *i* + (*hs* ! *i*) *x* / *x*) *powr* (*p* + *1*)) (*x powr* (*p* + *1*) ∗ *1*)
unfolding *m′-def* **using** *x-pos* **by** (*subst max.commute*, *intro max-mult-left*) *simp*
also have (*x powr* (*p* + *1*) ∗ (*bs* ! *i* + (*hs* ! *i*) *x* / *x*) *powr* (*p* + *1*)) =
(*bs* ! *i* + (*hs* ! *i*) *x* / *x*) *powr* (*p* + *1*) ∗ *x powr* (*p* + *1*) **by** *simp*
also have ... = (*bs* ! *i* ∗ *x* + (*hs* ! *i*) *x*) *powr* (*p* + *1*)
using *x x1-pos step-pos*[*OF i x*] *x-pos i x0-le-x1 x-pos*
by (*subst powr-mult*[*symmetric*]) (*simp add*: *field-simps*, *simp*, *simp add*: *algebra-simps*)
also have *x powr* (*p* + *1*) ∗ *1* = *x powr* (*p* + *1*) **by** *simp*
also have (*x* − *?x′*) ∗ (*c1* ∗ *g x* / *?m*) = *integral* (λ-. *c1* ∗ *g x* / *?m*) *?x′ x*
using *x′-le-x* **by** (*subst integral-const*[*OF c1-g-m-nonneg*]) *auto*
also {
fix *u* **assume** *u*: *u* ≥ *?x′ u* ≤ *x*
have *u powr* (*p* + *1*) ≤ *?m* **using** *x u x′-pos* **by** (*intro powr-upper-bound mult-pos-pos*) *simp-all*
moreover from *x′-pos u* **have** *u* ≥ *0* **by** *simp*
moreover from *c1* **and** *u* **have** *c1* ∗ *g x* ≤ *g u* **by** *simp*
ultimately have *c1* ∗ *g x* ∗ *u powr* (*p* + *1*) ≤ *g u* ∗ *?m* **using** *c1 x u x0-hb-bound7*[*OF x i*]
by (*intro mult-mono g-nonneg*) *auto*
with *m-pos u step-pos*[*OF i x*]
have *c1* ∗ *g x* / *?m* ≤ *g u* / *u powr* (*p* + *1*) **by** (*simp add*: *field-simps*)
}
hence *integral* (λ-. *c1* ∗ *g x* / *?m*) *?x′ x* ≤ *integral* (λ*u*. *g u* / *u powr* (*p* + *1*)) *?x′ x*
using *x0-hb-bound7*[*OF x i*] *x′-le-x*
by (*intro integral-le ballI akra-bazzi-integrable integrable-const c1-g-m-nonneg*) *simp-all*
finally have *c3* ∗ *g x* ≤ *g-approx i x* **using** *x-pos*
unfolding *g-approx-def* **by** (*simp add*: *field-simps*)
}

thus *?thesis* **using** *that* ‹$c3 > 0$› **by** *blast*
qed

lemma *f-bounded-above*:
assumes c': $c' > 0$
obtains c **where** $\bigwedge x.\ x \geq x_0 \Longrightarrow x \leq x_1 \Longrightarrow f\ x \leq (1/2) * (c * f\text{-}approx\ x)$ $c \geq c'$ $c > 0$
proof−
obtain c **where** c: $\bigwedge x.\ x_0 \leq x \Longrightarrow x \leq x_1 \Longrightarrow f\text{-}approx\ x \geq c$ $c > 0$
by (*rule f-approx-bounded-below*) *blast*
have *fb1-nonneg*: $fb1 \geq 0$ **using** *f-base1*[*of* x_0] *f-nonneg*[*of* x_0] *x0-le-x1* **by** *simp*
{
fix x **assume** x: $x \geq x_0$ $x \leq x_1$
with *f-base1 x0-pos* **have** $f\ x \leq fb1$ **by** *simp*
moreover from c **and** x **have** $f\text{-}approx\ x \geq c$ **by** *blast*
ultimately have $f\ x * c \leq fb1 * f\text{-}approx\ x$ **using** c *fb1-nonneg* **by** (*intro mult-mono*) *simp-all*
also from *f-approx-nonneg* x **have** ... $\leq (fb1 + 1) * f\text{-}approx\ x$ **by** (*simp add*: *algebra-simps*)
finally have $f\ x \leq ((fb1{+}1) / c) * f\text{-}approx\ x$ **by** (*simp add*: *field-simps* c)
also have ... $\leq max\ ((fb1{+}1) / c)\ c' * f\text{-}approx\ x$
by (*intro mult-right-mono*) (*simp-all add*: *f-approx-nonneg* x)
finally have $f\ x \leq 1/2 * (max\ ((fb1{+}1) / c)\ c' * 2 * f\text{-}approx\ x)$ **by** *simp*
}
moreover have $max\ ((fb1{+}1) / c)\ c' * 2 \geq max\ ((fb1{+}1) / c)\ c'$
by (*subst mult-le-cancel-left1*) (*insert* c', *simp*)
hence $max\ ((fb1{+}1) / c)\ c' * 2 \geq c'$ **by** (*rule order.trans*[*OF max.cobounded2*])
moreover from *fb1-nonneg* **and** c **have** $(fb1{+}1) / c > 0$ **by** *simp*
hence $max\ ((fb1{+}1) / c)\ c' * 2 > 0$ **by** *simp*
ultimately show *?thesis* **by** (*rule that*)
qed

lemma *akra-bazzi-upper*:
obtains $c6$ **where** $\bigwedge x.\ x \geq x_0 \Longrightarrow f\ x \leq c6 * f\text{-}approx\ x$ $c6 > 0$
proof−
obtain $c3$ **where** $c3$: $\bigwedge x\ i.\ x \geq x_1 \Longrightarrow i < k \Longrightarrow c3 * g\ x \leq g\text{-}approx\ i\ x$ $c3 > 0$
by (*rule g-bounds1*) *blast*
hence $2 / c3 > 0$ **by** *simp*
then obtain $c6$ **where** $c6$: $\bigwedge x.\ x \geq x_0 \Longrightarrow x \leq x_1 \Longrightarrow f\ x \leq 1/2 * (c6 * f\text{-}approx\ x)$
$c6 \geq 2 / c3$ $c6 > 0$
by (*rule f-bounded-above*) *blast*

{
fix x :: *real* **assume** x: $x \geq x_0$
hence $f\ x \leq c6 * (1 - ln\ x\ powr\ (-e/2)) * f\text{-}approx\ x$

proof (*induction x rule*: *akra-bazzi-induct*)
case (*base x*)
from *base* **have** *f x* $\leq$ *1/2* $*$ (*c6* $*$ *f-approx x*) **by** (*intro c6*) *simp-all*
also have *1* $-$ *ln x powr* ($-e/2$) $\geq$ *1/2* **using** *asymptotics4 base* **by** *simp*
hence (*1* $-$ *ln x powr* ($-e/2$)) $*$ (*c6* $*$ *f-approx x*) $\geq$ *1/2* $*$ (*c6* $*$ *f-approx x*)
using *c6 f-approx-nonneg base x0-ge-1* **by** (*intro mult-right-mono mult-nonneg-nonneg*) *simp-all*
finally show *?case* **by** (*simp add*: *algebra-simps*)
next
case (*rec x*)
let *?a* = λi. *as*!*i* **and** *?b* = λi. *bs*!*i* **and** *?h* = λi. *hs*!*i*
let *?int* = *integral* (λu. *g u* / *u powr* (*p+1*)) x_0 *x*
let *?int1* = λi. *integral* (λu. *g u* / *u powr* (*p+1*)) x_0 (*?b i*$*$*x*+*?h i x*)
let *?int2* = λi. *integral* (λu. *g u* / *u powr* (*p+1*)) (*?b i*$*$*x*+*?h i x*) *x*
let *?l* = *ln x powr* ($-e/2$) **and** *?l'* = λi. *ln* (*?b i*$*$*x* + *?h i x*) *powr* ($-e/2$)

from *rec* **and** *x0-le-x1* **have** *x*: $x \geq x_0$ **by** *simp*
with *x0-pos* **have** *x-pos*: *x* > *0* **and** *x-nonneg*: *x* $\geq$ *0* **by** *simp-all*
from *c6 c3* **have** *c6* $*$ *c3* $\geq$ *2* **by** (*simp add*: *field-simps*)
have *f x* = ($\sum$ *i*<*k*. *?a i* $*$ *f* (*?b i*$*$*x* + *?h i x*)) + *g x* (**is** - = *?sum* + -)
using *f-rec*[*of x*] *rec.hyps x0-le-x1* **by** *simp*
also have *?sum* $\leq$ ($\sum$ *i*<*k*. (*?a i*$*$*?b i powr p*) $*$ (*c6*$*$*x powr p*$*$(*1* $-$ *?l*)$*$(*1* + *?int* $-$ *c3*$*$*g x*/*x powr p*))) (**is** - $\leq$ *?sum'*)
proof (*rule setsum-mono*, *clarify*)
fix *i* **assume** *i*: *i* < *k*
from *rec.hyps i* **have** x_0 < *bs* ! *i* $*$ *x* + (*hs* ! *i*) *x* **by** (*intro x0-hb-bound7*) *simp-all*
hence *1* + *?int1 i* $\geq$ *1* **by** (*intro f-approx-aux x0-hb-bound7*) *simp-all*
hence *int-nonneg*: *1* + *?int1 i* $\geq$ *0* **by** *simp*
have *l-le-1*: *ln x powr* $-(e/2)$ $\leq$ *1* **using** *asymptotics3*[*OF x*] **by** (*simp add*: *field-simps*)

from *i* **have** *f* (*?b i*$*$*x* + *?h i x*) $\leq$ *c6* $*$ (*1* $-$ *?l' i*) $*$ *f-approx* (*?b i*$*$*x* + *?h i x*)
by (*rule rec.IH*)
hence *?a i* $*$ *f* (*?b i*$*$*x* + *?h i x*) $\leq$ *?a i* $*$ ... **using** *a-ge-0 i*
by (*intro mult-left-mono*) *simp-all*
also have ... = (*c6*$*$*?a i*$*$(*1* + *?int1 i*)) $*$ ((*?b i*$*$*x* + *?h i x*) *powr p* $*$ (*1* $-$ *?l' i*))
unfolding *f-approx-def* **by** (*simp add*: *algebra-simps*)
also from *i rec.hyps c6 a-ge-0*
have ... $\leq$ (*c6*$*$*?a i*$*$(*1* + *?int1 i*)) $*$ ((*?b i*$*$*x*) *powr p* $*$ (*1* $-$ *?l*))
by (*intro mult-left-mono asymptotics2' mult-nonneg-nonneg int-nonneg*) *simp-all*
also have ... = (*1* + *?int1 i*) $*$ (*c6*$*$*?a i*$*$(*?b i*$*$*x*) *powr p* $*$ (*1* $-$ *?l*))
by (*simp add*: *algebra-simps*)
also from *rec.hyps i* **have** x_0 < *bs* ! *i* $*$ *x* + (*hs* ! *i*) *x* **by** (*intro x0-hb-bound7*) *simp-all*
hence *?int1 i* = *?int* $-$ *?int2 i*

apply (*subst eq-diff-eq*)
apply (*intro integral-combine akra-bazzi-integrable*)
apply (*insert rec.hyps step-le-x*[*OF i*, *of x*], *simp-all*)
done
also from *rec.hyps i* **have** *c3* $*$ *g x* $\leq$ *g-approx i x* **by** (*intro c3*) *simp-all*
hence *?int2 i* $\geq$ *c3*$*$*g x*/*x powr p* **unfolding** *g-approx-def* **using** *x-pos*
by (*simp add*: *field-simps*)
hence (*1* + (*?int* − *?int2 i*)) $*$ (*c6*$*$*?a i*$*$(*?b i*$*$*x*) *powr p* $*$ (*1* − *?l*)) $\leq$
(*1* + *?int* − *c3*$*$*g x*/*x powr p*) $*$ (*c6*$*$*?a i*$*$(*?b i*$*$*x*) *powr p* $*$ (*1* − *?l*))
using *i c6 a-ge-0 l-le-1*
by (*intro mult-right-mono mult-nonneg-nonneg*) (*simp-all add*: *field-simps*)
also have ... = (*?a i*$*$*?b i powr p*) $*$ (*c6*$*$*x powr p*$*$(*1* − *?l*) $*$ (*1* + *?int* −
c3$*$*g x*/*x powr p*))
using *b-pos*[*of bs*!*i*] *x x0-pos i* **by** (*subst powr-mult*) (*simp-all add*: *algebra-simps*)
finally show *?a i* $*$ *f* (*?b i*$*$*x* + *?h i x*) $\leq$ ... **.**
qed

hence *?sum* + *g x* $\leq$ *?sum'* + *g x* **by** *simp*
also have ... = *c6* $*$ *x powr p* $*$ (*1* − *?l*) $*$ (*1* + *?int* − *c3*$*$*g x*/*x powr p*) +
g x
by (*simp add*: *setsum-left-distrib*[*symmetric*] *p-props*)
also have ... = *c6* $*$ (*1* − *?l*) $*$ *f-approx x* − (*c6*$*$*c3*$*$(*1* − *?l*) − *1*) $*$ *g x*
unfolding *f-approx-def* **using** *x-pos* **by** (*simp add*: *field-simps*)
also {
from *c6 c3* **have** *c6*$*$*c3* $\geq$ *2* **by** (*simp add*: *field-simps*)
moreover have (*1* − *?l*) $\geq$ *1*/*2* **using** *asymptotics4*[*OF x*] **by** *simp*
ultimately have *c6*$*$*c3*$*$(*1* − *?l*) $\geq$ *2* $*$ (*1*/*2*) **by** (*intro mult-mono*) *simp-all*
with *x x-pos* **have** (*c6*$*$*c3*$*$(*1* − *?l*) − *1*) $*$ *g x* $\geq$ *0*
by (*intro mult-nonneg-nonneg g-nonneg*) *simp-all*
hence *c6* $*$ (*1* − *?l*) $*$ *f-approx x* − (*c6*$*$*c3*$*$(*1* − *?l*) − *1*) $*$ *g x* $\leq$
c6 $*$ (*1* − *?l*) $*$ *f-approx x* **by** (*simp add*: *algebra-simps*)
}
finally show *?case* **.**
qed
also from *x c6* **have** ... $\leq$ *c6* $*$ *1* $*$ *f-approx x*
by (*intro mult-left-mono mult-right-mono f-approx-nonneg*) *simp-all*
finally have *f x* $\leq$ *c6* $*$ *f-approx x* **by** *simp*
}
from *this* **and** *c6*(*3*) **show** *?thesis* **by** (*rule that*)
qed

lemma *akra-bazzi-bigo*:
f $\in$ *O*(λ*x*. *x powr p* $*$(*1* + *integral* (λ*u*. *g u* / *u powr* (*p* + *1*)) x_0 *x*))
apply (*fold f-approx-def*, *rule akra-bazzi-upper*, *erule landau-o.bigI*)
apply (*subst eventually-at-top-linorder*, *rule exI*[*of* - x_0])
apply (*simp add*: *f-nonneg f-approx-nonneg*)
done

end

end

4 The discrete Akra-Bazzi theorem

theory *Akra-Bazzi*
imports
 Complex-Main
 ../Landau-Symbols/Landau-Symbols
 Akra-Bazzi-Real
begin

lemma *ex-mono*: $(\exists x.\ P\ x) \Longrightarrow (\bigwedge x.\ P\ x \Longrightarrow Q\ x) \Longrightarrow (\exists x.\ Q\ x)$ **by** *blast*

lemma *x-over-ln-mono*:
 assumes $(e{::}real) > 0$
 assumes $x > exp\ e$
 assumes $x \leq y$
 shows $x\ /\ ln\ x\ powr\ e \leq y\ /\ ln\ y\ powr\ e$
proof (*rule DERIV-nonneg-imp-mono*[*of - - λx. x / ln x powr e*])
 fix t **assume** t: $t \in \{x..y\}$
 from *assms*(*1*) **have** $1 < exp\ e$ **by** *simp*
 from *this* **and** *assms*(*2*) **have** $x > 1$ **by** (*rule less-trans*)
 with t **have** t': $t > 1$ **by** *simp*
 from ‹$x > exp\ e$› **and** t **have** $t > exp\ e$ **by** *simp*
 with t' **have** $ln\ t > ln\ (exp\ e)$ **by** (*subst ln-less-cancel-iff*) *simp-all*
 hence t'': $ln\ t > e$ **by** *simp*
 show $((\lambda x.\ x\ /\ ln\ x\ powr\ e)$ *has-real-derivative*
  $(ln\ t - e)\ /\ ln\ t\ powr\ (e{+}1))\ (at\ t)$ **using** *assms* $t\ t'\ t''$
  by (*force intro*!: *derivative-eq-intros simp*: *powr-divide2*[*symmetric*] *field-simps powr-add*)
 from t'' **show** $(ln\ t - e)\ /\ ln\ t\ powr\ (e + 1) \geq 0$ **by** (*intro divide-nonneg-nonneg*) *simp-all*
qed (*simp-all add*: *assms*)

definition *akra-bazzi-term* :: $nat \Rightarrow nat \Rightarrow real \Rightarrow (nat \Rightarrow nat) \Rightarrow bool$ **where**
 akra-bazzi-term $x_0\ x_1\ b\ t =$
  $(\exists e\ h.\ e > 0 \wedge (\lambda x.\ h\ x) \in O(\lambda x.\ real\ x\ /\ ln\ (real\ x)\ powr\ (1{+}e)) \wedge$
   $(\forall x{\geq}x_1.\ t\ x \geq x_0 \wedge t\ x < x \wedge b{*}x + h\ x = real\ (t\ x)))$

lemma *akra-bazzi-termI* [*intro?*]:
 assumes $e > 0\ (\lambda x.\ h\ x) \in O(\lambda x.\ real\ x\ /\ ln\ (real\ x)\ powr\ (1{+}e))$
  $\bigwedge x.\ x \geq x_1 \Longrightarrow t\ x \geq x_0\ \bigwedge x.\ x \geq x_1 \Longrightarrow t\ x < x$
  $\bigwedge x.\ x \geq x_1 \Longrightarrow b{*}x + h\ x = real\ (t\ x)$
 shows *akra-bazzi-term* $x_0\ x_1\ b\ t$
 using *assms* **unfolding** *akra-bazzi-term-def* **by** *blast*

lemma *akra-bazzi-term-imp-less*:

assumes *akra-bazzi-term* x_0 x_1 *b t x* $\geq x_1$
shows *t x* $<$ *x*
using *assms* **unfolding** *akra-bazzi-term-def* **by** *blast*

lemma *akra-bazzi-term-imp-less'*:
assumes *akra-bazzi-term* x_0 (*Suc* x_1) *b t x* $> x_1$
shows *t x* $<$ *x*
using *assms* **unfolding** *akra-bazzi-term-def* **by** *force*

locale *akra-bazzi-recursion* =
fixes x_0 x_1 *k* :: *nat* **and** *as bs* :: *real list* **and** *ts* :: (*nat* $\Rightarrow$ *nat*) *list* **and** *f* :: *nat* $\Rightarrow$ *real*
assumes *k-not-0*: $k \neq 0$
and *length-as*: *length as* = *k*
and *length-bs*: *length bs* = *k*
and *length-ts*: *length ts* = *k*
and *a-ge-0*: $a \in$ *set as* $\Longrightarrow a \geq 0$
and *b-bounds*: $b \in$ *set bs* $\Longrightarrow b \in \{0<..<1\}$
and *ts*: $i <$ *length bs* $\Longrightarrow$ *akra-bazzi-term* x_0 x_1 (*bs*!*i*) (*ts*!*i*)
begin

sublocale *akra-bazzi-params k as bs*
using *length-as length-bs k-not-0 a-ge-0 b-bounds* **by** *unfold-locales*

lemma *ts-nonempty*: *ts* $\neq$ [] **using** *length-ts k-not-0* **by** (*cases ts*) *simp-all*

definition *e-hs* :: *real* $\times$ (*nat* $\Rightarrow$ *real*) *list* **where**
e-hs = (*SOME* (*e*,*hs*).
$e > 0 \wedge$ *length hs* = *k* $\wedge$ ($\forall h \in$*set hs*. (λx. *h x*) $\in O(\lambda x$. *real x* / *ln* (*real x*) *powr* ($1+e$))) $\wedge$
($\forall t \in$*set ts*. $\forall x \geq x_1$. *t x* $\geq x_0 \wedge$ *t x* $<$ *x*) $\wedge$
($\forall i<k$. $\forall x \geq x_1$. (*bs*!*i*)$*x$ + (*hs*!*i*) *x* = *real* ((*ts*!*i*) *x*))
)

definition *e* = (*case e-hs of* (*e*,-) $\Rightarrow$ *e*)
definition *hs* = (*case e-hs of* (-,*hs*) $\Rightarrow$ *hs*)

lemma *filterlim-powr-zero-cong*:
filterlim (λx. *P* (*x*::*real*) (*x powr* (*0*::*real*))) *F at-top* = *filterlim* (λx. *P x 1*) *F at-top*
apply (*rule filterlim-cong*[*OF refl refl*])
using *eventually-gt-at-top*[*of 0*::*real*] **by** *eventually-elim simp-all*

lemma *e-hs-aux*:
$0 < e \wedge$ *length hs* = *k* $\wedge$
($\forall h \in$*set hs*. (λx. *h x*) $\in O(\lambda x$. *real x* / *ln* (*real x*) *powr* ($1 + e$))) $\wedge$
($\forall t \in$*set ts*. $\forall x \geq x_1$. $x_0 \leq$ *t x* $\wedge$ *t x* $<$ *x*) $\wedge$

(∀ *i*<*k*. ∀ *x*≥x_1. (*bs*!*i*)∗*x* + (*hs*!*i*) *x* = *real* ((*ts*!*i*) *x*))
proof−
have *Ex* (λ(*e*,*hs*). *e* > *0* ∧ *length hs* = *k* ∧
(∀ *h*∈*set hs*. (λ*x*. *h x*) ∈ *O*(λ*x*. *real x* / *ln* (*real x*) *powr* (*1*+*e*))) ∧
(∀ *t*∈*set ts*. ∀ *x*≥x_1. *t x* ≥ x_0 ∧ *t x* < *x*) ∧
(∀ *i*<*k*. ∀ *x*≥x_1. (*bs*!*i*)∗*x* + (*hs*!*i*) *x* = *real* ((*ts*!*i*) *x*)))
proof−
from *ts* **have** *A*: ∀ *i*∈{..<*k*}. *akra-bazzi-term* x_0 x_1 (*bs*!*i*) (*ts*!*i*) **by** (*auto simp*: *length-bs*)
hence ∃ *e h*. (∀ *i*<*k*. *e i* > *0* ∧
(λ*x*. *h i x*) ∈ *O*(λ*x*. *real x* / *ln* (*real x*) *powr* (*1*+*e i*)) ∧
(∀ *x*≥x_1. (*ts*!*i*) *x* ≥ x_0 ∧ (*ts*!*i*) *x* < *x*) ∧
(∀ *i*<*k*. ∀ *x*≥x_1. (*bs*!*i*)∗*real x* + *h i x* = *real* ((*ts*!*i*) *x*)))
unfolding *akra-bazzi-term-def*
by (*subst* (*asm*) *bchoice-iff*, *subst* (*asm*) *bchoice-iff*) *blast*
then guess *ee* :: - ⇒ *real* **and** *hh* :: - ⇒ *nat* ⇒ *real*
by (*elim exE conjE*) **note** *eh* = *this*
def *e* ≡ *Min* {*ee i* |*i*. *i* < *k*} **and** *hs* ≡ *map hh* (*upt 0 k*)
have *e-pos*: *e* > *0* **unfolding** *e-def* **using** *eh k-not-0* **by** (*subst Min-gr-iff*) *auto*
moreover have *length hs* = *k* **unfolding** *hs-def* **by** (*simp-all add*: *length-ts*)
moreover have *hs-growth*: ∀ *h*∈*set hs*. (λ*x*. *h x*) ∈ *O*(λ*x*. *real x* / *ln* (*real x*) *powr* (*1*+*e*))
proof
fix *h* **assume** *h* ∈ *set hs*
then obtain *i* **where** *t*: *i* < *k h* = *hh i* **unfolding** *hs-def* **by** *force*
hence (λ*x*. *h x*) ∈ *O*(λ*x*. *real x* / *ln* (*real x*) *powr* (*1*+*ee i*)) **using** *eh* **by** *blast*
also from *t k-not-0* **have** *e* ≤ *ee i* **unfolding** *e-def* **by** (*subst Min-le-iff*) *auto*
hence (λ*x*::*nat*. *real x* / *ln* (*real x*) *powr* (*1*+*ee i*)) ∈ *O*(λ*x*. *real x* / *ln* (*real x*) *powr* (*1*+*e*))
by (*intro bigo-real-nat-transfer*) *auto*
finally show (λ*x*. *h x*) ∈ *O*(λ*x*. *real x* / *ln* (*real x*) *powr* (*1*+*e*)) **.**
qed
moreover have ∀ *t*∈*set ts*. (∀ *x*≥x_1. *t x* ≥ x_0 ∧ *t x* < *x*)
proof (*rule ballI*)
fix *t* **assume** *t* ∈ *set ts*
then obtain *i* **where** *i* < *k t* = *ts*!*i* **using** *length-ts* **by** (*subst* (*asm*) *in-set-conv-nth*) *auto*
with *eh* **show** ∀ *x*≥x_1. *t x* ≥ x_0 ∧ *t x* < *x* **unfolding** *hs-def* **by** *force*
qed
moreover have ∀ *i*<*k*. ∀ *x*≥x_1. (*bs*!*i*)∗*x* + (*hs*!*i*) *x* = *real* ((*ts*!*i*) *x*)
proof (*rule allI*, *rule impI*)
fix *i* **assume** *i*: *i* < *k*
with *eh* **show** ∀ *x*≥x_1. (*bs*!*i*)∗*x* + (*hs*!*i*) *x* = *real* ((*ts*!*i*) *x*)
using *length-ts* **unfolding** *hs-def* **by** *fastforce*
qed
ultimately show *?thesis* **by** *blast*

```
  qed
  from someI-ex[OF this, folded e-hs-def] show ?thesis
    unfolding e-def hs-def by (intro conjI) fastforce+
qed

lemma
  e-pos: e > 0 and length-hs: length hs = k and
  hs-growth: ⋀h. h ∈ set hs ⟹ (λx. h x) ∈ O(λx. real x / ln (real x) powr (1 +
e)) and
  step-ge-x0: ⋀t x. t ∈ set ts ⟹ x ≥ x₁ ⟹ x₀ ≤ t x and
  step-less: ⋀t x. t ∈ set ts ⟹ x ≥ x₁ ⟹ t x < x and
  decomp:    ⋀i x. i < k ⟹ x ≥ x₁ ⟹ (bs!i)*x + (hs!i) x = real ((ts!i) x)
by (insert e-hs-aux) simp-all

lemma h-in-hs [intro, simp]: i < k ⟹ hs ! i ∈ set hs
  by (rule nth-mem) (simp add: length-hs)

lemma t-in-ts [intro, simp]: i < k ⟹ ts ! i ∈ set ts
  by (rule nth-mem) (simp add: length-ts)

lemma x0-less-x1: x₀ < x₁ and x0-le-x1: x₀ ≤ x₁
proof−
  from ts-nonempty have x₀ ≤ hd ts x₁ using step-ge-x0[of hd ts x₁] by simp
  also from ts-nonempty have ... < x₁ by (intro step-less) simp-all
  finally show x₀ < x₁ by simp
  thus x₀ ≤ x₁ by simp
qed

lemma akra-bazzi-induct [consumes 1, case-names base rec]:
  assumes x ≥ x₀
  assumes base: ⋀x. x ≥ x₀ ⟹ x < x₁ ⟹ P x
  assumes rec:  ⋀x. x ≥ x₁ ⟹ (⋀t. t ∈ set ts ⟹ P (t x)) ⟹ P x
  shows   P x
proof (insert assms(1), induction x rule: less-induct)
  case (less x)
  with assms step-less step-ge-x0 show P x by (cases x < x₁) auto
qed

end

locale akra-bazzi-function = akra-bazzi-recursion +
  fixes integrable integral
  assumes integral: akra-bazzi-integral integrable integral
  fixes g :: nat ⇒ real
  assumes f-nonneg-base: x ≥ x₀ ⟹ x < x₁ ⟹ f x ≥ 0
  and     f-rec:         x ≥ x₁ ⟹ f x = g x + (∑i<k. as!i ∗ f ((ts!i) x))
  and     g-nonneg:      x ≥ x₁ ⟹ g x ≥ 0
  and     ex-pos-a:      ∃a∈set as. a > 0
begin
```

lemma *ex-pos-a'*: $\exists\, i{<}k.\ as!i > 0$
  using *ex-pos-a* **by** (*auto simp*: *in-set-conv-nth length-as*)

sublocale *akra-bazzi-params-nonzero*
  using *length-as length-bs ex-pos-a a-ge-0 b-bounds* **by** *unfold-locales*

definition *g-real* :: *real* ⇒ *real* **where** *g-real x* = *g* (*nat* $\lfloor x \rfloor$)

lemma *g-real-real*[*simp*]: *g-real* (*real x*) = *g x* **unfolding** *g-real-def* **by** *simp*

lemma *f-nonneg*: $x \geq x_0 \Longrightarrow f\,x \geq 0$
proof (*induction x rule*: *akra-bazzi-induct*)
  case (*base x*)
  with *f-nonneg-base* **show** $f\ x \geq 0$ **by** *simp*
next
  case (*rec x*)
  from *rec.hyps* **have** $g\ x \geq 0$ **by** (*intro g-nonneg*) *simp*
  moreover have ($\sum i{<}k.\ as!i{*}f\ ((ts!i)\ x)) \geq 0$ **using** *rec.hyps length-ts length-as*
    by (*intro setsum-nonneg ballI mult-nonneg-nonneg*[*OF a-ge-0 rec.IH*]) *simp-all*
  ultimately show $f\ x \geq 0$ **using** *rec.hyps* **by** (*simp add*: *f-rec*)
qed

definition *hs'* = *map* ($\lambda h\ x.\ h$ (*nat* $\lfloor x{::}real \rfloor$)) *hs*

lemma *length-hs'*: *length hs'* = *k* **unfolding** *hs'-def* **by** (*simp add*: *length-hs*)

lemma *hs'-real*: $i < k \Longrightarrow$ (*hs'*!*i*) (*real x*) = (*hs*!*i*) *x*
  unfolding *hs'-def* **by** (*simp add*: *length-hs*)

lemma *h-bound*:
  obtains *hb* **where** $hb > 0$ **and**
    eventually ($\lambda x.\ \forall h{\in}set\ hs'.\ |h\ x| \leq hb * x\ /\ ln\ x\ powr\ (1 + e)$) *at-top*
proof−
  have $\forall h{\in}set\ hs.\ \exists c{>}0.\ eventually\ (\lambda x.\ |h\ x| \leq c * real\ x\ /\ ln\ (real\ x)\ powr\ (1$
$+\ e))\ at\text{-}top$
  proof
    fix *h* **assume** *h*: $h \in set\ hs$
    hence ($\lambda x.\ h\ x$) $\in O(\lambda x.\ real\ x\ /\ ln\ (real\ x)\ powr\ (1 + e))$ **by** (*rule hs-growth*)
    thus $\exists c{>}0.\ eventually\ (\lambda x.\ |h\ x| \leq c * x\ /\ ln\ x\ powr\ (1 + e))\ at\text{-}top$
      unfolding *bigo-def* **by** *auto*
  qed
  from *bchoice*[*OF this*] **obtain** *hb* **where** *hb*:
    $\forall h{\in}set\ hs.\ hb\ h > 0 \wedge eventually\ (\lambda x.\ |h\ x| \leq hb\ h * real\ x\ /\ ln\ (real\ x)$
powr (1 + *e*)) *at-top* **by** *blast*
  def $hb' \equiv max\ 1\ (Max\ \{hb\ h\ |h.\ h \in set\ hs\})$
  have $hb' > 0$ **unfolding** *hb'-def* **by** *simp*
  moreover have $\forall h{\in}set\ hs.\ eventually\ (\lambda x.\ |h\ (nat\ \lfloor x \rfloor)| \leq hb' * x\ /\ ln\ x\ powr$

(*1* + *e*)) *at-top*
proof (*intro ballI*, *rule eventually-mp*[*OF always-eventually eventually-conj*], *clarify*)
fix *h* **assume** *h*: $h \in$ *set hs*
with *hb* **have** *hb-pos*: *hb h* > *0* **by** *auto*

show *eventually* (λx. x > *exp* (*1* + *e*) + *1*) *at-top* **by** (*rule eventually-gt-at-top*)
from *h hb* **have** *e*: *eventually* (λx. $|h$ (*nat* $\lfloor x$ :: *real*$\rfloor$)$|$ $\leq$
hb h $*$ *real* (*nat* $\lfloor x \rfloor$) / *ln* (*real* (*nat* $\lfloor x \rfloor$)) *powr* (*1* + *e*)) *at-top*
by (*intro eventually-natfloor*) *blast*
show *eventually* (λx. $|h$ (*nat* $\lfloor x$ :: *real*$\rfloor$)$|$ $\leq$ *hb'* $*$ *x* / *ln x powr* (*1* + *e*)) *at-top*
using *e eventually-gt-at-top*
proof *eventually-elim*
fix *x* :: *real* **assume** *x*: *x* > *exp* (*1* + *e*) + *1*

have *x'*: *x* > *0* **by** (*rule le-less-trans*[*OF - x*]) *simp-all*
assume $|h$ (*nat* $\lfloor x \rfloor$)$|$ $\leq$ *hb h*$*$*real* (*nat* $\lfloor x \rfloor$)/*ln* (*real* (*nat* $\lfloor x \rfloor$)) *powr* (*1* + *e*)
also {
from *x* **have** *exp* (*1* + *e*) < *real* (*nat* $\lfloor x \rfloor$) **by** *linarith*
moreover have *x* > *0* **by** (*rule le-less-trans*[*OF - x*]) *simp-all*
hence *real* (*nat* $\lfloor x \rfloor$) $\leq$ *x* **by** *simp*
ultimately have *real* (*nat* $\lfloor x \rfloor$)/*ln* (*real* (*nat* $\lfloor x \rfloor$)) *powr* (*1*+*e*) $\leq$ *x*/*ln x powr* (*1*+*e*)
using *e-pos* **by** (*intro x-over-ln-mono*) *simp-all*
from *hb-pos mult-left-mono*[*OF this*, *of hb h*]
have *hb h* $*$ *real* (*nat* $\lfloor x \rfloor$)/*ln* (*real* (*nat* $\lfloor x \rfloor$)) *powr* (*1*+*e*) $\leq$ *hb h* $*$ *x* / *ln x powr* (*1*+*e*)
by (*simp add*: *algebra-simps*)
}
also from *h* **have** *hb h* $\leq$ *hb'*
unfolding *hb'-deff-rec* **by** (*intro order.trans*[*OF Max.coboundedI max.cobounded2*]) *auto*
with *x'* **have** *hb h*$*$*x*/*ln x powr* (*1*+*e*) $\leq$ *hb'*$*$*x*/*ln x powr* (*1*+*e*)
by (*intro mult-right-mono divide-right-mono*) *simp-all*
finally show $|h$ (*nat* $\lfloor x \rfloor$)$|$ $\leq$ *hb'* $*$ *x* / *ln x powr* (*1* + *e*) **.**
qed
qed
hence $\forall$ *h*$\in$*set hs'*. *eventually* (λx. $|h\ x|$ $\leq$ *hb'* $*$ *x* / *ln x powr* (*1* + *e*)) *at-top*
by (*auto simp*: *hs'-def*)
hence *eventually* (λx. $\forall$ *h*$\in$*set hs'*. $|h\ x|$ $\leq$ *hb'* $*$ *x* / *ln x powr* (*1* + *e*)) *at-top*
by (*intro eventually-ball-finite*) *simp-all*
ultimately show *?thesis* **by** (*rule that*)
qed

lemma *C-bound*:
assumes $\bigwedge b.\ b \in$ *set bs* $\Longrightarrow$ *C* < *b hb* > *0*
shows *eventually* (λx::*real*. $\forall$ *b*$\in$*set bs*. *C*$*$*x* $\leq$ *b*$*$*x* − *hb*$*$*x*/*ln x powr* (*1*+*e*)) *at-top*
proof−

from *e-pos* **have** $((\lambda x.\ hb * ln\ x\ powr\ -(1+e)) \longrightarrow 0)$ *at-top*
  by (*intro tendsto-mult-right-zero tendsto-neg-powr ln-at-top*) *simp-all*
with *assms* **have** $\forall\, b{\in}set\ bs.\ eventually\ (\lambda x.\ |hb * ln\ x\ powr\ -(1+e)| < b - C)$ *at-top*
  by (*force simp*: *tendsto-iff dist-real-def*)
hence *eventually* $(\lambda x.\ \forall\, b{\in}set\ bs.\ |hb * ln\ x\ powr\ -(1+e)| < b - C)$ *at-top*
  by (*intro eventually-ball-finite*) *simp-all*
note A = *eventually-conj*[*OF this eventually-gt-at-top*]
show *?thesis* **using** A **apply** *eventually-elim*
proof *clarify*
  fix $x\ b$ :: *real* **assume** x: $x > 0$ **and** b: $b \in set\ bs$
  assume A: $\forall\, b{\in}set\ bs.\ |hb * ln\ x\ powr\ -(1+e)| < b - C$
  from b A *assms* **have** $hb * ln\ x\ powr\ -(1+e) < b - C$ **by** *simp*
    with x **have** $x * (hb * ln\ x\ powr\ -(1+e)) < x * (b - C)$ **by** (*intro mult-strict-left-mono*)
  thus $C{*}x \le b{*}x - hb{*}x\ /\ ln\ x\ powr\ (1+e)$
    by (*subst* (*asm*) *powr-minus*) (*simp-all add*: *field-simps*)
qed
qed

end

locale *akra-bazzi-lower* = *akra-bazzi-function* +
fixes g' :: *real* $\Rightarrow$ *real*
assumes *f-pos*: *eventually* $(\lambda x.\ f\ x > 0)$ *at-top*
and *g-growth2*: $\exists\, C\ c2.\ c2 > 0 \wedge C < Min\ (set\ bs) \wedge$
  eventually $(\lambda x.\ \forall\, u{\in}\{C{*}x..x\}.\ g'\ u \le c2 * g'\ x)$ *at-top*
and g'*-integrable*: $\exists\, a.\ \forall\, b{\ge}a.\ integrable\ (\lambda u.\ g'\ u\ /\ u\ powr\ (p + 1))\ a\ b$
and g'*-bounded*: *eventually* $(\lambda a{::}real.\ (\forall\, b{>}a.\ \exists\, c.\ \forall\, x{\in}\{a..b\}.\ g'\ x \le c))$ *at-top*
and *g-bigomega*: $g \in \Omega(\lambda x.\ g'\ (real\ x))$
and g'*-nonneg*: *eventually* $(\lambda x.\ g'\ x \ge 0)$ *at-top*
begin

definition $gc2 \equiv SOME\ gc2.\ gc2 > 0 \wedge eventually\ (\lambda x.\ g\ x \ge gc2 * g'\ (real\ x))$ *at-top*

lemma *gc2*: $gc2 > 0$ *eventually* $(\lambda x.\ g\ x \ge gc2 * g'\ (real\ x))$ *at-top*
proof−
from *g-bigomega* **guess** c **by** (*elim landau-omega.bigE*) **note** c = *this*
from g'*-nonneg* **have** *eventually* $(\lambda x{::}nat.\ g'\ (real\ x) \ge 0)$ *at-top* **by** (*rule eventually-nat-real*)
with $c(2)$ **have** *eventually* $(\lambda x.\ g\ x \ge c * g'\ (real\ x))$ *at-top*
  using *eventually-ge-at-top*[*of* x_1] **by** *eventually-elim* (*insert g-nonneg*, *simp-all*)
with $c(1)$ **have** $\exists\, gc2.\ gc2 > 0 \wedge eventually\ (\lambda x.\ g\ x \ge gc2 * g'\ (real\ x))$ *at-top* **by** *blast*
from *someI-ex*[*OF this*] **show** $gc2 > 0$ *eventually* $(\lambda x.\ g\ x \ge gc2 * g'\ (real\ x))$

at-top
  unfolding *gc2-def* **by** *blast+*
qed

definition $gx0 \equiv max\ x_1\ (SOME\ gx0.\ \forall x \geq gx0.\ g\ x \geq gc2 * g'\ (real\ x) \wedge f\ x > 0 \wedge g'\ (real\ x) \geq 0)$
definition $gx1 \equiv max\ gx0\ (SOME\ gx1.\ \forall x \geq gx1.\ \forall i<k.\ (ts!i)\ x \geq gx0)$

lemma *gx0*:
 assumes $x \geq gx0$
 shows $g\ x \geq gc2 * g'\ (real\ x)$ $f\ x > 0$ $g'\ (real\ x) \geq 0$
proof−
 from *eventually-conj*[*OF gc2*(*2*) *eventually-conj*[*OF f-pos eventually-nat-real*[*OF g′-nonneg*]]]
  have $\exists gx0.\ \forall x \geq gx0.\ g\ x \geq gc2 * g'\ (real\ x) \wedge f\ x > 0 \wedge g'\ (real\ x) \geq 0$
  by (*simp add*: *eventually-at-top-linorder*)
 note *someI-ex*[*OF this*]
 moreover have $x \geq (SOME\ gx0.\ \forall x \geq gx0.\ g\ x \geq gc2 * g'\ (real\ x) \wedge f\ x > 0 \wedge g'\ (real\ x) \geq 0)$
  using *assms* **unfolding** *gx0-def* **by** *simp*
 ultimately show $g\ x \geq gc2 * g'\ (real\ x)$ $f\ x > 0$ $g'\ (real\ x) \geq 0$ **unfolding** *gx0-def* **by** *blast+*
qed

lemma *gx1*:
 assumes $x \geq gx1$ $i < k$
 shows $(ts!i)\ x \geq gx0$
proof−
 def $mb \equiv Min\ (set\ bs)/2$
 from *b-bounds bs-nonempty* **have** *mb-pos*: $mb > 0$ **unfolding** *mb-def* **by** *simp*
 from *h-bound* **guess** *hb* **. note** *hb* = *this*
 from *e-pos* **have** $((\lambda x.\ hb * ln\ x\ powr\ -(1 + e)) \longrightarrow 0)$ *at-top*
  by (*intro tendsto-mult-right-zero tendsto-neg-powr ln-at-top*) *simp-all*
 moreover note *mb-pos*
 ultimately have *eventually* $(\lambda x.\ hb * ln\ x\ powr\ -(1 + e) < mb)$ *at-top* **using** *hb*(*1*)
  by (*subst* (*asm*) *tendsto-iff*) (*simp-all add*: *dist-real-def*)

 from *eventually-nat-real*[*OF hb*(*2*)] *eventually-nat-real*[*OF this*]
   eventually-ge-at-top eventually-ge-at-top
 have *eventually* $(\lambda x.\ \forall i<k.\ (ts!i)\ x \geq gx0)$ *at-top* **apply** *eventually-elim*
 proof *clarify*
  fix $i\ x :: nat$ **assume** *A*: $hb * ln\ (real\ x)\ powr\ -(1+e) < mb$ **and** *i*: $i < k$
  assume *B*: $\forall h \in set\ hs'.\ |h\ (real\ x)| \leq hb * real\ x\ /\ ln\ (real\ x)\ powr\ (1+e)$
  with *i* **have** *B′*: $|(hs'!i)\ (real\ x)| \leq hb * real\ x\ /\ ln\ (real\ x)\ powr\ (1+e)$
   using *length-hs′*[*symmetric*] **by** *auto*
  assume *C*: $x \geq nat\ \lceil gx0/mb \rceil$
  hence *C′*: $real\ gx0/mb \leq real\ x$ **by** *linarith*
  assume *D*: $x \geq x_1$

from *mb-pos* **have** *real gx0* = *mb* ∗ (*real gx0*/*mb*) **by** *simp*
also from *i bs-nonempty* **have** *mb* ≤ *bs*!*i*/*2* **unfolding** *mb-def* **by** *simp*
hence *mb* ∗ (*real gx0*/*mb*) ≤ *bs*!*i*/*2* ∗ *x*
using *C′ i b-bounds*[*of bs*!*i*] *mb-pos* **by** (*intro mult-mono*) *simp-all*
also have ... = *bs*!*i*∗*x* + −*bs*!*i*/*2* ∗ *x* **by** *simp*
also {
have −(*hs*!*i*) *x* ≤ |(*hs*!*i*) *x*| **by** *simp*
also from *i B′ length-hs* **have** |(*hs*!*i*) *x*| ≤ *hb* ∗ *real x* / *ln* (*real x*) *powr* (*1*+*e*)
by (*simp add*: *hs′-def*)
also from *A* **have** *hb* / *ln x powr* (*1*+*e*) ≤ *mb*
by (*subst* (*asm*) *powr-minus*) (*simp add*: *field-simps*)
hence *hb* / *ln x powr* (*1*+*e*) ∗ *x* ≤ *mb* ∗ *x* **by** (*intro mult-right-mono*) *simp-all*
hence *hb* ∗ *x* / *ln x powr* (*1*+*e*) ≤ *mb* ∗ *x* **by** *simp*
also from *i* **have** ... ≤ (*bs*!*i*/*2*) ∗ *x* **unfolding** *mb-def* **by** (*intro mult-right-mono*) *simp-all*
finally have −*bs*!*i*/*2* ∗ *x* ≤ (*hs*!*i*) *x* **by** *simp*
}
also have *bs*!*i*∗*real x* + (*hs*!*i*) *x* = *real* ((*ts*!*i*) *x*) **using** *i D decomp* **by** *simp*
finally show (*ts*!*i*) *x* ≥ *gx0* **by** *simp*
qed
hence ∃ *gx1*. ∀ *x*≥*gx1*. ∀ *i*<*k*. *gx0* ≤ (*ts*!*i*) *x* (**is** *Ex ?P*)
by (*simp add*: *eventually-at-top-linorder*)
from *someI-ex*[*OF this*] **have** *?P* (*SOME x*. *?P x*) **.**
moreover have ⋀*x*. *x* ≥ *gx1* ⟹ *x* ≥ (*SOME x*. *?P x*) **unfolding** *gx1-def* **by** *simp*
ultimately have *?P gx1* **by** *blast*
with *assms* **show** *?thesis* **by** *blast*
qed

lemma *gx0-ge-x1*: *gx0* ≥ x_1 **unfolding** *gx0-def* **by** *simp*
lemma *gx0-le-gx1*: *gx0* ≤ *gx1* **unfolding** *gx1-def* **by** *simp*

function *f2′* :: *nat* ⇒ *real* **where**
x < *gx1* ⟹ *f2′ x* = *max 0* (*f x* / *gc2*)
| *x* ≥ *gx1* ⟹ *f2′ x* = *g′* (*real x*) + (∑ *i*<*k*. *as*!*i*∗*f2′* ((*ts*!*i*) *x*))
using *le-less-linear* **by** (*blast*, *simp-all*)
termination by (*relation Wellfounded.measure* (λ*x*. *x*))
(*insert gx0-le-gx1 gx0-ge-x1*, *simp-all add*: *step-less*)

lemma *f2′-nonneg*: *x* ≥ *gx0* ⟹ *f2′ x* ≥ *0*
by (*induction x rule*: *f2′.induct*)
(*auto intro*!: *add-nonneg-nonneg setsum-nonneg gx0 gx1 mult-nonneg-nonneg*[*OF a-ge-0*])

lemma *f2′-le-f*: *x* ≥ x_0 ⟹ *gc2* ∗ *f2′ x* ≤ *f x*
proof (*induction rule*: *f2′.induct*)
case (*1 x*)

with *gc2 f-nonneg* **show** *?case* **by** (*simp add*: *max-def field-simps*)
next
case *prems*: (*2 x*)
with *gx0 gx0-le-gx1* **have** $gc2 * g'$ (*real x*) $\leq g\ x$ **by** *force*
moreover from *step-ge-x0 prems*(*1*) *gx0-ge-x1 gx0-le-gx1*
have $\bigwedge i.\ i < k \Longrightarrow x_0 \leq (ts!i)\ x$ **by** *simp*
hence $\bigwedge i.\ i < k \Longrightarrow as!i * (gc2 * f2'\ ((ts!i)\ x)) \leq as!i * f\ ((ts!i)\ x)$
using *prems*(*1*) **by** (*intro mult-left-mono a-ge-0 prems*(*2*)) *auto*
hence $gc2 * (\sum i<k.\ as!i*f2'\ ((ts!i)\ x)) \leq (\sum i<k.\ as!i*f\ ((ts!i)\ x))$
by (*subst setsum-right-distrib*, *intro setsum-mono*) (*simp-all add*: *algebra-simps*)
ultimately show *?case* **using** *prems*(*1*) *gx0-ge-x1 gx0-le-gx1*
by (*simp-all add*: *algebra-simps f-rec*)
qed

lemma *f2'-pos*: *eventually* ($\lambda x.\ f2'\ x > 0$) *at-top*
proof (*subst eventually-at-top-linorder*, *intro exI allI impI*)
fix *x* :: *nat* **assume** $x \geq gx0$
thus $f2'\ x > 0$
proof (*induction x rule*: *f2'.induct*)
case (*1 x*)
with *gc2 gx0*(*2*)[*of x*] **show** *?case* **by** (*simp add*: *max-def field-simps*)
next
case *prems*: (*2 x*)
have $(\sum i<k.\ as!i*f2'\ ((ts!i)\ x)) > 0$
proof (*rule setsum-pos'*)
from *ex-pos-a'* **guess** *i* **by** (*elim exE conjE*) **note** *i* = *this*
with *prems*(*1*) *gx0 gx1* **have** $as!i * f2'\ ((ts!i)\ x) > 0$
by (*intro mult-pos-pos prems*(*2*)) *simp-all*
with *i* **show** $\exists\, i\in\{..<k\}.\ as!i * f2'\ ((ts!i)\ x) > 0$ **by** *blast*
next
fix *i* **assume** *i*: $i \in \{..<k\}$
with *prems*(*1*) **have** $f2'\ ((ts!i)\ x) > 0$ **by** (*intro prems*(*2*) *gx1*) *simp-all*
with *i* **show** $as!i * f2'\ ((ts!i)\ x) \geq 0$ **by** (*intro mult-nonneg-nonneg*[*OF a-ge-0*]) *simp-all*
qed *simp-all*
with *prems*(*1*) *gx0-le-gx1* **show** *?case* **by** (*auto intro*!: *add-nonneg-pos gx0*)
qed
qed

lemma *bigomega-f-aux*:
obtains *a* **where** $a \geq A\ \forall a'\geq a.\ a' \in \mathbb{N} \longrightarrow$
$f \in \Omega(\lambda x.\ x\ powr\ p * (1 + integral\ (\lambda u.\ g'\ u\ /\ u\ powr\ (p + 1))\ a'\ x))$
proof−
from *g'-integrable* **guess** *a0* **by** (*elim exE*) **note** *a0* = *this*
from *h-bound* **guess** *hb* **. note** *hb* = *this*
moreover from *g-growth2* **guess** *C c2* **by** (*elim conjE exE*) **note** *C* = *this*
hence *eventually* ($\lambda x.\ \forall b\in set\ bs.\ C*x \leq b*x - hb*x/ln\ x\ powr\ (1 + e)$) *at-top*
using *hb*(*1*) *bs-nonempty* **by** (*intro C-bound*) *simp-all*

moreover from *b-bounds hb(1) e-pos*
have *eventually* ($\lambda x.\ \forall b\in set\ bs.\ akra\text{-}bazzi\text{-}asymptotics\ b\ hb\ e\ p\ x$) *at-top*
by (*rule akra-bazzi-asymptotics*)
moreover note *g′-bounded C(3) g′-nonneg eventually-natfloor*[*OF f2′-pos*] *eventually-natfloor*[*OF gc2(2)*]
ultimately have *eventually* ($\lambda x.\ (\forall h\in set\ hs'.\ |h\ x| \le hb*x/ln\ x\ powr\ (1+e)) \wedge$
$(\forall b\in set\ bs.\ C*x \le b*x - hb*x/ln\ x\ powr\ (1+e)) \wedge$
$(\forall b\in set\ bs.\ akra\text{-}bazzi\text{-}asymptotics\ b\ hb\ e\ p\ x) \wedge$
$(\forall b>x.\ \exists c.\ \forall x\in\{x..b\}.\ g'\ x \le c) \wedge f2'\ (nat\ \lfloor x\rfloor) > 0 \wedge$
$(\forall u\in\{C * x..x\}.\ g'\ u \le c2 * g'\ x) \wedge$
$g'\ x \ge 0)$ *at-top*
by (*intro eventually-conj*) (*force elim*!: *eventually-conjE*)+
then have $\exists X.\ (\forall x\ge X.\ (\forall h\in set\ hs'.\ |h\ x| \le hb*x/ln\ x\ powr\ (1+e)) \wedge$
$(\forall b\in set\ bs.\ C*x \le b*x - hb*x/ln\ x\ powr\ (1+e)) \wedge$
$(\forall b\in set\ bs.\ akra\text{-}bazzi\text{-}asymptotics\ b\ hb\ e\ p\ x) \wedge$
$(\forall b>x.\ \exists c.\ \forall x\in\{x..b\}.\ g'\ x \le c) \wedge$
$(\forall u\in\{C * x..x\}.\ g'\ u \le c2 * g'\ x) \wedge$
$f2'\ (nat\ \lfloor x\rfloor) > 0 \wedge g'\ x \ge 0)$
by (*subst* (*asm*) *eventually-at-top-linorder*) (*erule ex-mono*, *blast*)
then guess *X* **by** (*elim exE conjE*) **note** $X = this$

def $x_0'\text{-}min \equiv max\ A\ (max\ X\ (max\ a0\ (max\ gx1\ (max\ 1\ (real\ x_1 + 1)))))$
{
fix x_0' :: *real* **assume** *x0′-props*: $x_0' \ge x_0'\text{-}min\ x_0' \in \mathbb{N}$
hence *x0′-ge-x1*: $x_0' \ge real\ (x_1+1)$ **and** *x0′-ge-1*: $x_0' \ge 1$ **and** *x0′-ge-X*: $x_0' \ge X$
unfolding x_0'*-min-def* **by** *linarith*+
hence *x0′-pos*: $x_0' > 0$ **and** *x0′-nonneg*: $x_0' \ge 0$ **by** *simp-all*
have *x0′*: $\forall x\ge x_0'.\ (\forall h\in set\ hs'.\ |h\ x| \le hb*x/ln\ x\ powr\ (1+e))$
$\forall x\ge x_0'.\ (\forall b\in set\ bs.\ C*x \le b*x - hb*x/ln\ x\ powr\ (1+e))$
$\forall x\ge x_0'.\ (\forall b\in set\ bs.\ akra\text{-}bazzi\text{-}asymptotics\ b\ hb\ e\ p\ x)$
$\forall a\ge x_0'.\ \forall b>a.\ \exists c.\ \forall x\in\{a..b\}.\ g'\ x \le c$
$\forall x\ge x_0'.\ \forall u\in\{C * x..x\}.\ g'\ u \le c2 * g'\ x$
$\forall x\ge x_0'.\ f2'\ (nat\ \lfloor x\rfloor) > 0\ \forall x\ge x_0'.\ g'\ x \ge 0$
using *X x0′-ge-X* **by** *auto*
from *x0′-props(2)* **have** *x0′-int*: $real\ (nat\ \lfloor x_0'\rfloor) = x_0'$ **by** (*rule real-natfloor-nat*)
from *x0′-props* **have** *x0′-ge-gx1*: $x_0' \ge gx1$ **and** *x0′-ge-a0*: $x_0' \ge a0$
unfolding x_0'*-min-def* **by** *simp-all*
with *gx0-le-gx1* **have** *f2′-nonneg*: $\bigwedge x.\ x \ge x_0' \Longrightarrow f2'\ x \ge 0$ **by** (*force intro*!: *f2′-nonneg*)

def $bm \equiv Min\ (set\ bs)$
def $x_1' \equiv 2 * x_0' * inverse\ bm$
def $fb2 \equiv Min\ \{f2'\ x\ |x.\ x \in \{x_0'..x_1'\}\}$
def $gb2 \equiv SOME\ c.\ \forall x\in\{x_0'..x_1'\}.\ g'\ x \le c$

from *b-bounds bs-nonempty* **have** $bm > 0\ bm < 1$ **unfolding** *bm-def* **by** *auto*
hence $1 < 2 * inverse\ bm$ **by** (*simp add*: *field-simps*)
from *mult-strict-left-mono*[*OF this x0′-pos*]

have *x0'-lt-x1'*: $x_0' < x_1'$ **and** *x0'-le-x1'*: $x_0' \leq x_1'$ **unfolding** *x_1'-def* **by** *simp-all*

from *x0-le-x1 x0'-ge-x1* **have** *ge-x0'D*: $\bigwedge x.\ x_0' \leq \mathit{real}\ x \Longrightarrow x_0 \leq x$ **by** *simp*
from *x0'-ge-x1 x0'-le-x1'* **have** *gt-x1'D*: $\bigwedge x.\ x_1' < \mathit{real}\ x \Longrightarrow x_1 \leq x$ **by** *simp*

have *x0'-x1'*: $\forall b{\in}\mathit{set}\ bs.\ 2 * x_0' * \mathit{inverse}\ b \leq x_1'$
proof
  fix *b* **assume** *b*: $b \in \mathit{set}\ bs$
  hence $bm \leq b$ **by** (*simp add*: *bm-def*)
  moreover from *b bs-nonempty b-bounds* **have** $bm > 0$ $b > 0$ **unfolding** *bm-def* **by** *auto*
  ultimately have $\mathit{inverse}\ b \leq \mathit{inverse}\ bm$ **by** *simp*
  with *x0'-nonneg* **show** $2 * x_0' * \mathit{inverse}\ b \leq x_1'$
    unfolding *x_1'-def* **by** (*intro mult-left-mono*) *simp-all*
qed

note *f-nonneg' = f-nonneg*
have $\bigwedge x.\ \mathit{real}\ x \geq x_0' \Longrightarrow x \geq \mathit{nat}\ \lfloor x_0' \rfloor$ $\bigwedge x.\ \mathit{real}\ x \leq x_1' \Longrightarrow x \leq \mathit{nat}\ \lceil x_1' \rceil$ **by** *linarith+*
hence $\{x\ |x.\ \mathit{real}\ x \in \{x_0'..x_1'\}\} \subseteq \{x\ |x.\ x \in \{\mathit{nat}\ \lfloor x_0' \rfloor..\mathit{nat}\ \lceil x_1' \rceil\}\}$ **by** *auto*
hence *finite* $\{x\ |x{::}\mathit{nat}.\ \mathit{real}\ x \in \{x_0'..x_1'\}\}$ **by** (*rule finite-subset*) *auto*
hence *fin*: *finite* $\{f2'\ x\ |x{::}\mathit{nat}.\ \mathit{real}\ x \in \{x_0'..x_1'\}\}$ **by** *force*

note *facts = hs'-real e-pos length-hs' length-as length-bs k-not-0 a-ge-0 p-props x0'-ge-1*
    f2'-nonneg f-rec[*OF gt-x1'D*] *x0' x0'-int x0'-x1' gc2*(*1*) *decomp*
from *b-bounds x0'-le-x1' x0'-ge-gx1 gx0-le-gx1 x0'-ge-x1*
  interpret *abr*: *akra-bazzi-nat-to-real as bs hs' k* x_0' x_1' *hb e p f2' g'*
  by (*unfold-locales*) (*auto simp*: *facts simp del*: *f2'.simps intro*!: *f2'.simps*(*2*))

have *f'-nat*: $\bigwedge x{::}\mathit{nat}.\ \mathit{abr.f'}\ (\mathit{real}\ x) = \mathit{f2'}\ x$
proof−
  fix $x :: \mathit{nat}$ **show** *abr.f'* (*real* (*x*::*nat*)) = *f2' x*
  proof (*induction real x arbitrary*: *x rule*: *abr.f'.induct*)
    case (*2 x*)
    note *x = this*(*1*) **and** *IH = this*(*2*)
    from *x* **have** *abr.f'* (*real x*) = *g'* (*real x*) + ($\sum$ *i*<*k*. *as*!*i*∗*abr.f'* (*bs*!*i*∗*real x* + (*hs*!*i*) *x*))
      by (*auto simp*: *gt-x1'D hs'-real g-real-def intro*!: *setsum.cong*)
    also have ($\sum$ *i*<*k*. *as*!*i*∗*abr.f'* (*bs*!*i*∗*real x* + (*hs*!*i*) *x*)) =
        ($\sum$ *i*<*k*. *as*!*i*∗*f2'* ((*ts*!*i*) *x*))
    proof (*rule setsum.cong*, *simp*, *clarify*)
      fix *i* **assume** *i*: $i < k$
      from *i x x0'-le-x1' x0'-ge-x1* **have** *bs*!*i* ∗ *real x* + (*hs*!*i*) *x* = *real* ((*ts*!*i*) *x*)
        by (*intro decomp*) *simp-all*
      also from *i this* **have** *abr.f'* ... = *f2'* ((*ts*!*i*) *x*)
        by (*subst IH*[*of i*]) (*simp-all add*: *hs'-real*)
      finally show *as*!*i*∗*abr.f'* (*bs*!*i*∗*real x*+(*hs*!*i*) *x*) = *as*!*i*∗*f2'* ((*ts*!*i*) *x*) **by**

simp
qed
also have $g'\ x + ... = f2'\ x$ **using** *x x0'-ge-gx1 x0'-le-x1'*
by (*intro f2'.simps(2)[symmetric] gt-x1'D*) *simp-all*
finally show *?case* **.**
qed *simp*
qed
interpret *akra-bazzi-integral integrable integral* **by** (*rule integral*)
interpret *akra-bazzi-real-lower as bs hs'* $k\ x_0'\ x_1'$ *hb e p*
integrable integral abr.f' g' C fb2 gb2 c2
proof *unfold-locales*
fix x **assume** $x \geq x_0'\ x \leq x_1'$
thus $abr.f'\ x \geq 0$ **by** (*intro abr.f'-base*) *simp-all*
next
fix x **assume** $x{:}x \geq x_0'$
show *integrable* $(\lambda x.\ g'\ x\ /\ x\ powr\ (p + 1))\ x_0'\ x$
by (*rule integrable-subinterval[of - a0 x]*) (*insert a0 x0'-ge-a0 x*, *auto*)
next
fix x **assume** x: $x \geq x_0'\ x \leq x_1'$
have $x_0' = real\ (nat\ \lfloor x_0' \rfloor)$ **by** (*simp add*: *x0'-int*)
also from x **have** $... \leq real\ (nat\ \lfloor x \rfloor)$ **by** (*auto intro*!: *nat-mono floor-mono*)
finally have $x_0' \leq real\ (nat\ \lfloor x \rfloor)$ **.**
moreover have $real\ (nat\ \lfloor x \rfloor) \leq x_1'$ **using** *x x0'-ge-1* **by** *linarith*
ultimately have $f2'\ (nat\ \lfloor x \rfloor) \in \{f2'\ x\ |x.\ real\ x \in \{x_0'..x_1'\}\}$ **by** *force*
from *fin* **and** *this* **have** $f2'\ (nat\ \lfloor x \rfloor) \geq fb2$ **unfolding** *fb2-def* **by** (*rule Min-le*)
with x **show** $abr.f'\ x \geq fb2$ **by** *simp*
next
from *x0'-int x0'-le-x1'* **have** $\exists x{::}nat.\ real\ x \geq x_0' \wedge real\ x \leq x_1'$
by (*intro exI[of - nat* $\lfloor x_0' \rfloor$*]*) *simp-all*
moreover {
fix x :: *nat* **assume** $real\ x \geq x_0' \wedge real\ x \leq x_1'$
with *x0'(6)* **have** $f2'\ (nat\ \lfloor real\ x \rfloor) > 0$ **by** *blast*
hence $f2'\ x > 0$ **by** *simp*
}
ultimately show $fb2 > 0$ **unfolding** *fb2-def* **using** *fin* **by** (*subst Min-gr-iff*)
auto
next
fix x **assume** x: $x_0' \leq x\ x \leq x_1'$
with *x0'(4) x0'-lt-x1'* **have** $\exists\, c.\ \forall x{\in}\{x_0'..x_1'\}.\ g'\ x \leq c$ **by** *force*
from *someI-ex[OF this]* x **show** $g'\ x \leq gb2$ **unfolding** *gb2-def* **by** *simp*
qed (*insert g-nonneg integral x0'(2) C x0'-le-x1' x0'-ge-x1*, *simp-all add*: *facts*)

from *akra-bazzi-lower* **guess** *c5* **. note** $c5 = this$
have *eventually* $(\lambda x.\ |f\ x| \geq gc2 * c5 * |f\text{-}approx\ (real\ x)|)$ *at-top*
proof (*unfold eventually-at-top-linorder*, *intro exI allI impI*)
fix x :: *nat* **assume** $x \geq nat\ \lceil x_0' \rceil$
hence x: $real\ x \geq x_0'$ **by** *linarith*
note *c5(1)[OF x]*
also have $abr.f'\ (real\ x) = f2'\ x$ **by** (*rule f'-nat*)

also have $gc2 * ... \leq f\, x$ **using** x *x0'-ge-x1 x0-le-x1* **by** (*intro f2'-le-f*) *simp-all*
also have $f\, x = |f\, x|$ **using** x *f-nonneg' x0'-ge-x1 x0-le-x1* **by** *simp*
finally show $gc2 * c5 * |\textit{f-approx}\ (\textit{real}\ x)| \leq |f\, x|$
using *gc2 f-approx-nonneg*[*OF x*] **by** (*simp add*: *algebra-simps*)
qed
hence $f \in \Omega(\lambda x.\ \textit{f-approx}\ (\textit{real}\ x))$ **using** *gc2*(*1*) *f-nonneg' f-approx-nonneg*
by (*intro landau-omega.bigI*[*of gc2 * c5*] *eventually-conj*
mult-pos-pos c5 eventually-nat-real) (*auto simp*: *eventually-at-top-linorder*)
note *this*[*unfolded f-approx-def*]
}
moreover have x_0*'-min* $\geq A$ **unfolding** x_0*'-min-def gx0-ge-x1* **by** *simp*
ultimately show *?thesis* **by** (*intro that*) *auto*
qed

lemma *bigomega-f*:
obtains a **where** $a \geq A$ $f \in \Omega(\lambda x.\ x\ \textit{powr}\ p *(1 + \textit{integral}\ (\lambda u.\ g'\ u\ /\ u\ \textit{powr}\ (p+1))\ a\ x))$
proof−
from *bigomega-f-aux*[*of A*] **guess** a **. note** $a = this$
def $a' \equiv \textit{real}\ (\textit{max}\ (\textit{nat}\ \lceil a\rceil)\ 0) + 1$
note a
moreover have $a' \in \mathbb{N}$ **by** (*auto simp*: *max-def a'-def*)
moreover have $a' \geq a + 1$ **unfolding** *a'-def* **by** *linarith*
moreover from *this* **and** a **have** $a' \geq A$ **by** *simp*
ultimately show *?thesis* **by** (*intro that*[*of a'*]) *auto*
qed

end

locale *akra-bazzi-upper* = *akra-bazzi-function* +
fixes $g' :: \textit{real} \Rightarrow \textit{real}$
assumes *g'-integrable*: $\exists a.\ \forall b{\geq}a.\ \textit{integrable}\ (\lambda u.\ g'\ u\ /\ u\ \textit{powr}\ (p + 1))\ a\ b$
and *g-growth1*: $\exists C\ c1.\ c1 > 0 \wedge C < \textit{Min}\ (\textit{set}\ bs) \wedge$
$\textit{eventually}\ (\lambda x.\ \forall u{\in}\{C{*}x..x\}.\ g'\ u \geq c1 * g'\ x)\ \textit{at-top}$
and *g-bigo*: $g \in O(g')$
and *g'-nonneg*: $\textit{eventually}\ (\lambda x.\ g'\ x \geq 0)\ \textit{at-top}$
begin

definition $gc1 \equiv \textit{SOME}\ gc1.\ gc1 > 0 \wedge \textit{eventually}\ (\lambda x.\ g\ x \leq gc1 * g'\ (\textit{real}\ x))\ \textit{at-top}$

lemma *gc1*: $gc1 > 0$ $\textit{eventually}\ (\lambda x.\ g\ x \leq gc1 * g'\ (\textit{real}\ x))\ \textit{at-top}$
proof−
from *g-bigo* **guess** c **by** (*elim landau-o.bigE*) **note** $c = this$
from *g'-nonneg* **have** $\textit{eventually}\ (\lambda x{::}\textit{nat}.\ g'\ (\textit{real}\ x) \geq 0)\ \textit{at-top}$ **by** (*rule eventually-nat-real*)

with $c(2)$ **have** *eventually* $(\lambda x.\ g\ x \le c * g'\ (real\ x))$ *at-top*
using *eventually-ge-at-top*[*of* x_1] **by** *eventually-elim* (*insert g-nonneg*, *simp-all*)
with $c(1)$ **have** $\exists\, gc1.\ gc1 > 0 \wedge$ *eventually* $(\lambda x.\ g\ x \le gc1 * g'\ (real\ x))$ *at-top*
by *blast*
from *someI-ex*[*OF this*] **show** $gc1 > 0$ *eventually* $(\lambda x.\ g\ x \le gc1 * g'\ (real\ x))$ *at-top*
unfolding *gc1-def* **by** *blast+*
qed

definition $gx3 \equiv max\ x_1\ (SOME\ gx0.\ \forall x{\ge}gx0.\ g\ x \le gc1 * g'\ (real\ x))$

lemma *gx3*:
assumes $x \ge gx3$
shows $g\ x \le gc1 * g'\ (real\ x)$
proof−
from *gc1*(*2*) **have** $\exists\, gx3.\ \forall x{\ge}gx3.\ g\ x \le gc1 * g'\ (real\ x)$ **by** (*simp add*: *eventually-at-top-linorder*)
note *someI-ex*[*OF this*]
moreover have $x \ge (SOME\ gx0.\ \forall x{\ge}gx0.\ g\ x \le gc1 * g'\ (real\ x))$
using *assms* **unfolding** *gx3-def* **by** *simp*
ultimately show $g\ x \le gc1 * g'\ (real\ x)$ **unfolding** *gx3-def* **by** *blast*
qed

lemma *gx3-ge-x1*: $gx3 \ge x_1$ **unfolding** *gx3-def* **by** *simp*

function $f' :: nat \Rightarrow real$ **where**
$x < gx3 \Longrightarrow f'\ x = max\ 0\ (f\ x\ /\ gc1)$
$|\ x \ge gx3 \Longrightarrow f'\ x = g'\ (real\ x) + (\sum i{<}k.\ as!i{*}f'\ ((ts!i)\ x))$
using *le-less-linear* **by** (*blast*, *simp-all*)
termination by (*relation Wellfounded.measure* $(\lambda x.\ x)$)
(*insert gx3-ge-x1*, *simp-all add*: *step-less*)

lemma *f'-ge-f*: $x \ge x_0 \Longrightarrow gc1 * f'\ x \ge f\ x$
proof (*induction rule*: *f'.induct*)
case (*1 x*)
with *gc1 f-nonneg* **show** *?case* **by** (*simp add*: *max-def field-simps*)
next
case *prems*: (*2 x*)
with *gx3* **have** $gc1 * g'\ (real\ x) \ge g\ x$ **by** *force*
moreover from *step-ge-x0 prems*(*1*) *gx3-ge-x1*
have $\bigwedge i.\ i < k \Longrightarrow x_0 \le nat\ \lfloor (ts!i)\ x \rfloor$ **by** (*intro le-nat-floor*) *simp*
hence $\bigwedge i.\ i < k \Longrightarrow as!i * (gc1 * f'\ ((ts!i)\ x)) \ge as!i * f\ ((ts!i)\ x)$
using *prems*(*1*) **by** (*intro mult-left-mono a-ge-0 prems*(*2*)) *auto*
hence $gc1 * (\sum i{<}k.\ as!i{*}f'\ ((ts!i)\ x)) \ge (\sum i{<}k.\ as!i{*}f\ ((ts!i)\ x))$
by (*subst setsum-right-distrib*, *intro setsum-mono*) (*simp-all add*: *algebra-simps*)
ultimately show *?case* **using** *prems*(*1*) *gx3-ge-x1*
by (*simp-all add*: *algebra-simps f-rec*)
qed

```
lemma bigo-f-aux:
  obtains a where a ≥ A ∀ a'≥a. a' ∈ ℕ ⟶
    f ∈ O(λx. x powr p *(1 + integral (λu. g' u / u powr (p + 1)) a' x))
proof−
  from g'-integrable guess a0 by (elim exE) note a0 = this
  from h-bound guess hb . note hb = this
  moreover from g-growth1 guess C c1 by (elim conjE exE) note C = this
  hence eventually (λx. ∀ b∈set bs. C*x ≤ b*x − hb*x/ln x powr (1 + e)) at-top
    using hb(1) bs-nonempty by (intro C-bound) simp-all
  moreover from b-bounds hb(1) e-pos
    have eventually (λx. ∀ b∈set bs. akra-bazzi-asymptotics b hb e p x) at-top
    by (rule akra-bazzi-asymptotics)
  moreover note gc1(2) C(3) g'-nonneg
  ultimately have eventually (λx. (∀ h∈set hs'. |h x| ≤ hb*x/ln x powr (1+e)) ∧
                   (∀ b∈set bs. C*x ≤ b*x − hb*x/ln x powr (1+e)) ∧
                   (∀ b∈set bs. akra-bazzi-asymptotics b hb e p x) ∧
                   (∀ u∈{C*x..x}. g' u ≥ c1 * g' x) ∧ g' x ≥ 0) at-top
    by (intro eventually-conj) (force elim!: eventually-conjE)+
  then have ∃ X. (∀ x≥X. (∀ h∈set hs'. |h x| ≤ hb*x/ln x powr (1+e)) ∧
                   (∀ b∈set bs. C*x ≤ b*x − hb*x/ln x powr (1+e)) ∧
                   (∀ b∈set bs. akra-bazzi-asymptotics b hb e p x) ∧
                   (∀ u∈{C*x..x}. g' u ≥ c1 * g' x) ∧ g' x ≥ 0)
    by (subst (asm) eventually-at-top-linorder) fast
  then guess X by (elim exE conjE) note X = this

  def x_0'-min ≡ max A (max X (max 1 (max a0 (max gx3 (real x_1 + 1)))))
  {
  fix x_0' :: real assume x0'-props: x_0' ≥ x_0'-min x_0' ∈ ℕ
  hence x0'-ge-x1: x_0' ≥ real (x_1+1) and x0'-ge-1: x_0' ≥ 1 and x0'-ge-X: x_0'
≥ X
    unfolding x_0'-min-def by linarith+
  hence x0'-pos: x_0' > 0 and  x0'-nonneg: x_0' ≥ 0 by simp-all
  have x0': ∀ x≥x_0'. (∀ h∈set hs'. |h x| ≤ hb*x/ln x powr (1+e))
          ∀ x≥x_0'. (∀ b∈set bs. C*x ≤ b*x − hb*x/ln x powr (1+e))
          ∀ x≥x_0'. (∀ b∈set bs. akra-bazzi-asymptotics b hb e p x)
          ∀ x≥x_0'. ∀ u∈{C*x..x}. g' u ≥ c1 * g' x ∀ x≥x_0'. g' x ≥ 0
          using X x0'-ge-X by auto
 from x0'-props(2) have x0'-int: real (nat ⌊x_0'⌋) = x_0' by (rule real-natfloor-nat)
  from x0'-props have x0'-ge-gx0: x_0' ≥ gx3 and x0'-ge-a0: x_0' ≥ a0
    unfolding x_0'-min-def by simp-all
  hence f'-nonneg: ⋀x. x ≥ x_0' ⟹ f' x ≥ 0
    using order.trans[OF f-nonneg f'-ge-f] gc1(1) x0'-ge-x1 x0-le-x1
    by (simp add: zero-le-mult-iff del: f'.simps)

  def bm ≡ Min (set bs)
  def x_1' ≡ 2 * x_0' * inverse bm
  def fb1 ≡ Max {f' x |x. x ∈ {x_0'..x_1'}}

from b-bounds bs-nonempty have bm > 0 bm < 1 unfolding bm-def by auto
```

hence $1 < 2 * inverse\ bm$ **by** $(simp\ add: field\text{-}simps)$
from $mult\text{-}strict\text{-}left\text{-}mono[OF\ this\ x0'\text{-}pos]$
have $x0'\text{-}lt\text{-}x1'$: $x_0' < x_1'$ **and** $x0'\text{-}le\text{-}x1'$: $x_0' \leq x_1'$ **unfolding** $x_1'\text{-}def$ **by** $simp\text{-}all$

from $x0\text{-}le\text{-}x1\ x0'\text{-}ge\text{-}x1$ **have** $ge\text{-}x0'D$: $\bigwedge x.\ x_0' \leq real\ x \Longrightarrow x_0 \leq x$ **by** $simp$
from $x0'\text{-}ge\text{-}x1\ x0'\text{-}le\text{-}x1'$ **have** $gt\text{-}x1'D$: $\bigwedge x.\ x_1' < real\ x \Longrightarrow x_1 \leq x$ **by** $simp$

have $x0'\text{-}x1'$: $\forall b \in set\ bs.\ 2 * x_0' * inverse\ b \leq x_1'$
proof
fix b **assume** b: $b \in set\ bs$
hence $bm \leq b$ **by** $(simp\ add: bm\text{-}def)$
moreover from $b\ b\text{-}bounds\ bs\text{-}nonempty$ **have** $bm > 0\ b > 0$ **unfolding** $bm\text{-}def$ **by** $auto$
ultimately have $inverse\ b \leq inverse\ bm$ **by** $simp$
with $x0'\text{-}nonneg$ **show** $2 * x_0' * inverse\ b \leq x_1'$
unfolding $x_1'\text{-}def$ **by** $(intro\ mult\text{-}left\text{-}mono)\ simp\text{-}all$
qed

note $f\text{-}nonneg' = f\text{-}nonneg$
have $\bigwedge x.\ real\ x \geq x_0' \Longrightarrow x \geq nat\ \lfloor x_0' \rfloor\ \bigwedge x.\ real\ x \leq x_1' \Longrightarrow x \leq nat\ \lceil x_1' \rceil$ **by** $linarith+$
hence $\{x\ |x.\ real\ x \in \{x_0'..x_1'\}\} \subseteq \{x\ |x.\ x \in \{nat\ \lfloor x_0' \rfloor..nat\ \lceil x_1' \rceil\}\}$ **by** $auto$
hence $finite\ \{x\ |x::nat.\ real\ x \in \{x_0'..x_1'\}\}$ **by** $(rule\ finite\text{-}subset)\ auto$
hence fin: $finite\ \{f'\ x\ |x::nat.\ real\ x \in \{x_0'..x_1'\}\}$ **by** $force$

note $facts = hs'\text{-}real\ e\text{-}pos\ length\text{-}hs'\ length\text{-}as\ length\text{-}bs\ k\text{-}not\text{-}0\ a\text{-}ge\text{-}0\ p\text{-}props\ x0'\text{-}ge\text{-}1$
$f'\text{-}nonneg\ f\text{-}rec[OF\ gt\text{-}x1'D]\ x0'\ x0'\text{-}int\ x0'\text{-}x1'\ gc1(1)\ decomp$
from $b\text{-}bounds\ x0'\text{-}le\text{-}x1'\ x0'\text{-}ge\text{-}gx0\ x0'\text{-}ge\text{-}x1$
interpret abr: $akra\text{-}bazzi\text{-}nat\text{-}to\text{-}real\ as\ bs\ hs'\ k\ x_0'\ x_1'\ hb\ e\ p\ f'\ g'$
by $(unfold\text{-}locales)\ (auto\ simp\ add: facts\ simp\ del: f'.simps\ intro!: f'.simps(2))$

have $f'\text{-}nat$: $\bigwedge x::nat.\ abr.f'\ (real\ x) = f'\ x$
proof$-$
fix $x :: nat$ **show** $abr.f'\ (real\ (x::nat)) = f'\ x$
proof $(induction\ real\ x\ arbitrary: x\ rule: abr.f'.induct)$
case $(2\ x)$
note $x = this(1)$ **and** $IH = this(2)$
from x **have** $abr.f'\ (real\ x) = g'\ (real\ x) + (\sum i<k.\ as!i*abr.f'\ (bs!i*real\ x + (hs!i)\ x))$
by $(auto\ simp: gt\text{-}x1'D\ hs'\text{-}real\ intro!: setsum.cong)$
also have $(\sum i<k.\ as!i*abr.f'\ (bs!i*real\ x + (hs!i)\ x)) = (\sum i<k.\ as!i*f'\ ((ts!i)\ x))$
proof $(rule\ setsum.cong,\ simp,\ clarify)$
fix i **assume** i: $i < k$
from $i\ x\ x0'\text{-}le\text{-}x1'\ x0'\text{-}ge\text{-}x1$ **have** $bs!i * real\ x + (hs!i)\ x = real\ ((ts!i)\ x)$
by $(intro\ decomp)\ simp\text{-}all$
also from $i\ this$ **have** $abr.f'\ ... = f'\ ((ts!i)\ x)$

```
        by (subst IH[of i]) (simp-all add: hs'-real)
      finally show as!i*abr.f' (bs!i*real x+(hs!i) x) = as!i*f' ((ts!i) x) by simp
    qed
    also from x have g' x + ... = f' x using x0'-le-x1' x0'-ge-gx0 by simp
    finally show ?case .
  qed simp
qed

interpret akra-bazzi-integral integrable integral by (rule integral)
interpret akra-bazzi-real-upper as bs hs' k x₀' x₁' hb e p integrable integral abr.f'
g' C fb1 c1
proof (unfold-locales)
  fix x assume x ≥ x₀' x ≤ x₁'
  thus abr.f' x ≥ 0 by (intro abr.f'-base) simp-all
next
  fix x assume x:x ≥ x₀'
  show integrable (λx. g' x / x powr (p + 1)) x₀' x
    by (rule integrable-subinterval[of - a0 x]) (insert a0 x0'-ge-a0 x, auto)
next
  fix x assume x: x ≥ x₀' x ≤ x₁'
  have x₀' = real (nat ⌊x₀'⌋) by (simp add: x0'-int)
  also from x have ... ≤ real (nat ⌊x⌋) by (auto intro!: nat-mono floor-mono)
  finally have x₀' ≤ real (nat ⌊x⌋) .
  moreover have real (nat ⌊x⌋) ≤ x₁' using x x0'-ge-1 by linarith
  ultimately have f' (nat ⌊x⌋) ∈ {f' x |x. real x ∈ {x₀'..x₁'}} by force
 from fin and this have f' (nat ⌊x⌋) ≤ fb1 unfolding fb1-def by (rule Max-ge)
  with x show abr.f' x ≤ fb1 by simp
qed (insert x0'(2) x0'-le-x1' x0'-ge-x1 C, simp-all add: facts)

from akra-bazzi-upper guess c6 . note c6 = this
{
  fix x :: nat assume x ≥ nat ⌈x₀⌉
  hence x: real x ≥ x₀' by linarith
  have f x ≤ gc1 ∗ f' x using x x0'-ge-x1 x0-le-x1 by (intro f'-ge-f) simp-all
  also have f' x = abr.f' (real x) by (simp add: f'-nat)
  also note c6(1)[OF x]
  also from f-nonneg' x x0'-ge-x1 x0-le-x1 have f x = |f x| by simp
   also from f-approx-nonneg x have f-approx (real x) = |f-approx (real x)| by
simp
   finally have gc1 ∗ c6 ∗ |f-approx (real x)| ≥ |f x| using gc1 by (simp add:
algebra-simps)
}
hence eventually (λx. |f x| ≤ gc1 ∗ c6 ∗ |f-approx (real x)|) at-top
  using eventually-ge-at-top[of nat ⌈x₀⌉] by (auto elim!: eventually-mono)
hence f ∈ O(λx. f-approx (real x)) using gc1(1) f-nonneg' f-approx-nonneg
  by (intro landau-o.bigI[of gc1 ∗ c6] eventually-conj
      mult-pos-pos c6 eventually-nat-real) (auto simp: eventually-at-top-linorder)
note this[unfolded f-approx-def]
}
```

moreover have x_0'*-min* $\geq$ *A* **unfolding** x_0'*-min-def gx3-ge-x1* **by** *simp*
ultimately show *?thesis* **by** (*intro that*) *auto*
qed

lemma *bigo-f*:
obtains *a* **where** $a > A$ $f \in O(\lambda x.\ x\ powr\ p\ *(1 + integral\ (\lambda u.\ g'\ u\ /\ u\ powr\ (p + 1))\ a\ x))$
proof−
from *bigo-f-aux*[*of A*] **guess** *a* **. note** $a = this$
def $a' \equiv real\ (max\ (nat\ \lceil a\rceil)\ 0) + 1$
note *a*
moreover have $a' \in \mathbb{N}$ **by** (*auto simp*: *max-def a'-def*)
moreover have $a' \geq a + 1$ **unfolding** *a'-def* **by** *linarith*
moreover from *this* **and** *a* **have** $a' > A$ **by** *simp*
ultimately show *?thesis* **by** (*intro that*[*of a'*]) *auto*
qed

end

locale *akra-bazzi* = *akra-bazzi-function* +
fixes $g' :: real \Rightarrow real$
assumes *f-pos*: *eventually* $(\lambda x.\ f\ x > 0)$ *at-top*
and *g'-nonneg*: *eventually* $(\lambda x.\ g'\ x \geq 0)$ *at-top*
assumes *g'-integrable*: $\exists a.\ \forall b\geq a.\ integrable\ (\lambda u.\ g'\ u\ /\ u\ powr\ (p + 1))\ a\ b$
and *g-growth1*: $\exists C\ c1.\ c1 > 0 \land C < Min\ (set\ bs) \land$
eventually $(\lambda x.\ \forall u\in\{C*x..x\}.\ g'\ u \geq c1 * g'\ x)$ *at-top*
and *g-growth2*: $\exists C\ c2.\ c2 > 0 \land C < Min\ (set\ bs) \land$
eventually $(\lambda x.\ \forall u\in\{C*x..x\}.\ g'\ u \leq c2 * g'\ x)$ *at-top*
and *g-bounded*: *eventually* $(\lambda a{::}real.\ (\forall b>a.\ \exists c.\ \forall x\in\{a..b\}.\ g'\ x \leq c))$ *at-top*
and *g-bigtheta*: $g \in \Theta(g')$
begin

sublocale *akra-bazzi-lower* **using** *f-pos g-growth2 g-bounded*
bigthetaD2[*OF g-bigtheta*] *g'-nonneg g'-integrable* **by** *unfold-locales*
sublocale *akra-bazzi-upper* **using** *g-growth1 bigthetaD1*[*OF g-bigtheta*]
g'-nonneg g'-integrable **by** *unfold-locales*

lemma *bigtheta-f*:
obtains *a* **where** $a > A$ $f \in \Theta(\lambda x.\ x\ powr\ p\ *(1 + integral\ (\lambda u.\ g'\ u\ /\ u\ powr\ (p + 1))\ a\ x))$
proof−
from *bigo-f-aux*[*of A*] **guess** *a* **. note** $a = this$
moreover from *bigomega-f-aux*[*of A*] **guess** *b* **. note** $b = this$
let $?a = real\ (max\ (max\ (nat\ \lceil a\rceil)\ (nat\ \lceil b\rceil))\ 0) + 1$
have $?a \in \mathbb{N}$ **by** (*auto simp*: *max-def*)
moreover have $?a \geq a$ $?a \geq b$ **by** *linarith+*
ultimately have $f \in \Theta(\lambda x.\ x\ powr\ p\ *(1 + integral\ (\lambda u.\ g'\ u\ /\ u\ powr\ (p + 1))\ ?a\ x))$
using *a b* **by** (*intro bigthetaI*) *blast+*

moreover from *a b* **have** *?a* $>$ *A* **by** *linarith*
ultimately show *?thesis* **by** (*intro that*[*of ?a*]) *simp-all*
qed

end

named-theorems *akra-bazzi-term-intros introduction rules for Akra−Bazzi terms*

lemma *akra-bazzi-term-floor-add* [*akra-bazzi-term-intros*]:
assumes $(b{::}real) > 0$ $b < 1$ $real\ x_0 \le b * real\ x_1 + c$ $c < (1 - b) * real\ x_1$ $x_1 > 0$
shows *akra-bazzi-term* x_0 x_1 b $(\lambda x.\ nat\ \lfloor b{*}real\ x + c\rfloor)$
proof (*rule akra-bazzi-termI*[*OF zero-less-one*])
fix x **assume** x: $x \ge x_1$
from *assms x* **have** $real\ x_0 \le b * real\ x_1 + c$ **by** *simp*
also from *x assms* **have** ... $\le b * real\ x + c$ **by** *auto*
finally have *step-ge-x0*: $b * real\ x + c \ge real\ x_0$ **by** *simp*
thus $nat\ \lfloor b * real\ x + c\rfloor \ge x_0$ **by** (*subst le-nat-iff*) (*simp-all add*: *le-floor-iff*)

from *assms x* **have** $c < (1 - b) * real\ x_1$ **by** *simp*
also from *assms x* **have** ... $\le (1 - b) * real\ x$ **by** (*intro mult-left-mono*) *simp-all*
finally show $nat\ \lfloor b * real\ x + c\rfloor < x$ **using** *assms step-ge-x0*
by (*subst nat-less-iff*) (*simp-all add*: *floor-less-iff algebra-simps*)

from *step-ge-x0* **have** $real\text{-}of\text{-}int\ \lfloor c + b * real\ x\rfloor = real\text{-}of\text{-}int\ (nat\ \lfloor c + b * real\ x\rfloor)$ **by** *linarith*
thus $(b * real\ x) + (\lfloor b * real\ x + c\rfloor - (b * real\ x)) =$
$real\ (nat\ \lfloor b * real\ x + c\rfloor)$ **by** *linarith*
next
have $(\lambda x{::}nat.\ real\text{-}of\text{-}int\ \lfloor b * real\ x + c\rfloor - b * real\ x) \in O(\lambda\text{-}.\ |c| + 1)$
by (*intro landau-o.big-mono always-eventually allI*) *linarith*
also have $(\lambda\text{-}{::}nat.\ |c| + 1) \in O(\lambda x.\ real\ x\ /\ ln\ (real\ x)\ powr\ (1 + 1))$ **by** *force*
finally show $(\lambda x{::}nat.\ real\text{-}of\text{-}int\ \lfloor b * real\ x + c\rfloor - b * real\ x) \in$
$O(\lambda x.\ real\ x\ /\ ln\ (real\ x)\ powr\ (1{+}1))$ **.**
qed

lemma *akra-bazzi-term-floor-add′* [*akra-bazzi-term-intros*]:
assumes $(b{::}real) > 0$ $b < 1$ $real\ x_0 \le b * real\ x_1 + real\ c$ $real\ c < (1 - b) * real\ x_1$ $x_1 > 0$
shows *akra-bazzi-term* x_0 x_1 b $(\lambda x.\ nat\ \lfloor b{*}real\ x\rfloor + c)$
proof−
from *assms* **have** *akra-bazzi-term* x_0 x_1 b $(\lambda x.\ nat\ \lfloor b{*}real\ x + real\ c\rfloor)$
by (*rule akra-bazzi-term-floor-add*)
also have $(\lambda x.\ nat\ \lfloor b{*}real\ x + real\ c\rfloor) = (\lambda x{::}nat.\ nat\ \lfloor b{*}real\ x\rfloor + c)$
proof
fix $x :: nat$
have $\lfloor b * real\ x + real\ c\rfloor = \lfloor b * real\ x\rfloor + int\ c$ **by** *linarith*
also from *assms* **have** nat ... $= nat\ \lfloor b * real\ x\rfloor + c$ **by** (*simp add*: *nat-add-distrib*)

finally show *nat* $\lfloor b * \mathit{real}\ x + \mathit{real}\ c\rfloor = \mathit{nat}\ \lfloor b * \mathit{real}\ x\rfloor + c$ **.**
qed
finally show *?thesis* **.**
qed

lemma *akra-bazzi-term-floor-subtract* [*akra-bazzi-term-intros*]:
assumes $(b::\mathit{real}) > 0$ $b < 1$ $\mathit{real}\ x_0 \le b * \mathit{real}\ x_1 - c$ $0 < c + (1 - b) * \mathit{real}\ x_1$ $x_1 > 0$
shows *akra-bazzi-term* x_0 x_1 b $(\lambda x.\ \mathit{nat}\ \lfloor b*\mathit{real}\ x - c\rfloor)$
by (*subst diff-conv-add-uminus*, *rule akra-bazzi-term-floor-add*, *insert assms*) *simp-all*

lemma *akra-bazzi-term-floor-subtract′* [*akra-bazzi-term-intros*]:
assumes $(b::\mathit{real}) > 0$ $b < 1$ $\mathit{real}\ x_0 \le b * \mathit{real}\ x_1 - \mathit{real}\ c$ $0 < \mathit{real}\ c + (1 - b) * \mathit{real}\ x_1$ $x_1 > 0$
shows *akra-bazzi-term* x_0 x_1 b $(\lambda x.\ \mathit{nat}\ \lfloor b*\mathit{real}\ x\rfloor - c)$
proof−
from *assms* **have** *akra-bazzi-term* x_0 x_1 b $(\lambda x.\ \mathit{nat}\ \lfloor b*\mathit{real}\ x - \mathit{real}\ c\rfloor)$
by (*intro akra-bazzi-term-floor-subtract*) *simp-all*
also have $(\lambda x.\ \mathit{nat}\ \lfloor b*\mathit{real}\ x - \mathit{real}\ c\rfloor) = (\lambda x::\mathit{nat}.\ \mathit{nat}\ \lfloor b*\mathit{real}\ x\rfloor - c)$
proof
fix $x :: \mathit{nat}$
have $\lfloor b * \mathit{real}\ x - \mathit{real}\ c\rfloor = \lfloor b * \mathit{real}\ x\rfloor - \mathit{int}\ c$ **by** *linarith*
also from *assms* **have** *nat* ... $= \mathit{nat}\ \lfloor b * \mathit{real}\ x\rfloor - c$ **by** (*simp add*: *nat-diff-distrib*)
finally show *nat* $\lfloor b * \mathit{real}\ x - \mathit{real}\ c\rfloor = \mathit{nat}\ \lfloor b * \mathit{real}\ x\rfloor - c$ **.**
qed
finally show *?thesis* **.**
qed

lemma *akra-bazzi-term-floor* [*akra-bazzi-term-intros*]:
assumes $(b::\mathit{real}) > 0$ $b < 1$ $\mathit{real}\ x_0 \le b * \mathit{real}\ x_1$ $0 < (1 - b) * \mathit{real}\ x_1$ $x_1 > 0$
shows *akra-bazzi-term* x_0 x_1 b $(\lambda x.\ \mathit{nat}\ \lfloor b*\mathit{real}\ x\rfloor)$
using *assms akra-bazzi-term-floor-add*[**where** $c = 0$] **by** *simp*

lemma *akra-bazzi-term-ceiling-add* [*akra-bazzi-term-intros*]:
assumes $(b::\mathit{real}) > 0$ $b < 1$ $\mathit{real}\ x_0 \le b * \mathit{real}\ x_1 + c$ $c + 1 \le (1 - b) * x_1$
shows *akra-bazzi-term* x_0 x_1 b $(\lambda x.\ \mathit{nat}\ \lceil b*\mathit{real}\ x + c\rceil)$
proof (*rule akra-bazzi-termI*[*OF zero-less-one*])
fix x **assume** x: $x \ge x_1$
have $0 \le \mathit{real}\ x_0$ **by** *simp*
also from *assms* **have** $\mathit{real}\ x_0 \le b * \mathit{real}\ x_1 + c$ **by** *simp*
also from *assms* x **have** $b * \mathit{real}\ x_1 \le b * \mathit{real}\ x$ **by** (*intro mult-left-mono*) *simp-all*
hence $b * \mathit{real}\ x_1 + c \le b * \mathit{real}\ x + c$ **by** *simp*
also have $b * \mathit{real}\ x + c \le \mathit{real\text{-}of\text{-}int}\ \lceil b * \mathit{real}\ x + c\rceil$ **by** *linarith*
finally have *bx-nonneg*: $\mathit{real\text{-}of\text{-}int}\ \lceil b * \mathit{real}\ x + c\rceil \ge 0$ **.**

have $c + 1 \leq (1 - b) * x_1$ **by** *fact*
also have $(1 - b) * x_1 \leq (1 - b) * x$ **using** *assms x* **by** (*intro mult-left-mono*) *simp-all*
finally have $b * real\, x + c + 1 \leq real\, x$ **using** *assms* **by** (*simp add*: *algebra-simps*)
with *bx-nonneg* **show** $nat \lceil b * real\ x + c \rceil < x$ **by** (*subst nat-less-iff*) (*simp-all add*: *ceiling-less-iff*)

have $real\ x_0 \leq b * real\ x_1 + c$ **by** *fact*
also have ... $\leq$ *real-of-int* $\lceil ... \rceil$ **by** *linarith*
also have $x_1 \leq x$ **by** *fact*
finally show $x_0 \leq nat \lceil b * real\, x + c \rceil$ **using** *assms* **by** (*force simp*: *ceiling-mono*)

show $b * real\ x + (\lceil b * real\ x + c \rceil - b * real\ x) = real\ (nat\ \lceil b * real\ x + c \rceil)$
using *assms bx-nonneg* **by** *simp*
next
have $(\lambda x{::}nat.\ real\text{-}of\text{-}int\ \lceil b * real\ x + c \rceil - b * real\ x) \in O(\lambda\text{-}.\ |c| + 1)$
by (*intro landau-o.big-mono always-eventually allI*) *linarith*
also have $(\lambda\text{-}{::}nat.\ |c| + 1) \in O(\lambda x.\ real\ x\ /\ ln\ (real\ x)\ powr\ (1 + 1))$ **by** *force*
finally show $(\lambda x{::}nat.\ real\text{-}of\text{-}int\ \lceil b * real\ x + c \rceil - b * real\ x) \in$
$O(\lambda x.\ real\ x\ /\ ln\ (real\ x)\ powr\ (1{+}1))$ **.**
qed

lemma *akra-bazzi-term-ceiling-add'* [*akra-bazzi-term-intros*]:
assumes $(b{::}real) > 0$ $b < 1$ $real\ x_0 \leq b * real\ x_1 + real\ c$ $real\ c + 1 \leq (1 - b) * x_1$
shows *akra-bazzi-term* $x_0\ x_1\ b\ (\lambda x.\ nat\ \lceil b{*}real\ x \rceil + c)$
proof−
from *assms* **have** *akra-bazzi-term* $x_0\ x_1\ b\ (\lambda x.\ nat\ \lceil b{*}real\ x + real\ c \rceil)$
by (*rule akra-bazzi-term-ceiling-add*)
also have $(\lambda x.\ nat\ \lceil b{*}real\ x + real\ c \rceil) = (\lambda x{::}nat.\ nat\ \lceil b{*}real\ x \rceil + c)$
proof
fix $x :: nat$
from *assms* **have** $0 \leq b * real\ x$ **by** *simp*
also have $b * real\ x \leq real\text{-}of\text{-}int\ \lceil b * real\ x \rceil$ **by** *linarith*
finally have *bx-nonneg*: $\lceil b * real\ x \rceil \geq 0$ **by** *simp*

have $\lceil b * real\ x + real\ c \rceil = \lceil b * real\ x \rceil + int\ c$ **by** *linarith*
also from *assms bx-nonneg* **have** *nat* ... $= nat\ \lceil b * real\ x \rceil + c$
by (*subst nat-add-distrib*) *simp-all*
finally show $nat\ \lceil b * real\ x + real\ c \rceil = nat\ \lceil b * real\ x \rceil + c$ **.**
qed
finally show *?thesis* **.**
qed

lemma *akra-bazzi-term-ceiling-subtract* [*akra-bazzi-term-intros*]:
assumes $(b{::}real) > 0$ $b < 1$ $real\ x_0 \leq b * real\ x_1 - c$ $1 \leq c + (1 - b) * x_1$
shows *akra-bazzi-term* $x_0\ x_1\ b\ (\lambda x.\ nat\ \lceil b{*}real\ x - c \rceil)$
by (*subst diff-conv-add-uminus*, *rule akra-bazzi-term-ceiling-add*, *insert assms*)

simp-all

lemma *akra-bazzi-term-ceiling-subtract'* [*akra-bazzi-term-intros*]:
assumes $(b::real) > 0$ $b < 1$ *real* $x_0 \le b * real\ x_1 - real\ c$ $1 \le real\ c + (1 - b) * x_1$
shows *akra-bazzi-term* x_0 x_1 b $(\lambda x.\ nat\ \lceil b{*}real\ x\rceil - c)$
proof−
from *assms* **have** *akra-bazzi-term* x_0 x_1 b $(\lambda x.\ nat\ \lceil b{*}real\ x - real\ c\rceil)$
by (*intro akra-bazzi-term-ceiling-subtract*) *simp-all*
also have $(\lambda x.\ nat\ \lceil b{*}real\ x - real\ c\rceil) = (\lambda x{::}nat.\ nat\ \lceil b{*}real\ x\rceil - c)$
proof
fix $x :: nat$
from *assms* **have** $0 \le b * real\ x$ **by** *simp*
also have $b * real\ x \le real\text{-}of\text{-}int\ \lceil b * real\ x\rceil$ **by** *linarith*
finally have *bx-nonneg*: $\lceil b * real\ x\rceil \ge 0$ **by** *simp*

have $\lceil b * real\ x - real\ c\rceil = \lceil b * real\ x\rceil - int\ c$ **by** *linarith*
also from *assms bx-nonneg* **have** $nat\ ... = nat\ \lceil b * real\ x\rceil - c$ **by** *simp*
finally show $nat\ \lceil b * real\ x - real\ c\rceil = nat\ \lceil b * real\ x\rceil - c$ **.**
qed
finally show *?thesis* **.**
qed

lemma *akra-bazzi-term-ceiling* [*akra-bazzi-term-intros*]:
assumes $(b::real) > 0$ $b < 1$ *real* $x_0 \le b * real\ x_1$ $1 \le (1 - b) * x_1$
shows *akra-bazzi-term* x_0 x_1 b $(\lambda x.\ nat\ \lceil b{*}real\ x\rceil)$
using *assms akra-bazzi-term-ceiling-add*[**where** $c = 0$] **by** *simp*

end

5 The Master theorem

theory *Master-Theorem*
imports
~~/src/HOL/Multivariate-Analysis/Multivariate-Analysis
../Landau-Symbols/Landau-Symbols
Akra-Bazzi-Library
Akra-Bazzi
begin

lemma *fundamental-theorem-of-calculus-real*:
$a \le b \Longrightarrow \forall x{\in}\{a..b\}.\ (f\ \text{has-real-derivative}\ f'\ x)\ (at\ x\ within\ \{a..b\}) \Longrightarrow$
$(f'\ \text{has-integral}\ (f\ b - f\ a))\ \{a..b\}$
by (*intro fundamental-theorem-of-calculus ballI*)
(*simp-all add*: *has-field-derivative-iff-has-vector-derivative*[*symmetric*])

lemma *integral-powr*:
$y \ne -1 \Longrightarrow a \le b \Longrightarrow a > 0 \Longrightarrow integral\ \{a..b\}\ (\lambda x.\ x\ powr\ y :: real) =$

inverse $(y + 1) * (b$ *powr* $(y + 1) - a$ *powr* $(y + 1))$
by (*subst right-diff-distrib*, *intro integral-unique fundamental-theorem-of-calculus-real*)
(*auto intro*!: *derivative-eq-intros*)

lemma *integral-ln-powr-over-x*:
$y \neq -1 \Longrightarrow a \leq b \Longrightarrow a > 1 \Longrightarrow$ *integral* $\{a..b\}$ $(\lambda x.\ ln\ x\ powr\ y\ /\ x :: real)$ $=$
inverse $(y + 1) * (ln\ b\ powr\ (y + 1) - ln\ a\ powr\ (y + 1))$
by (*subst right-diff-distrib*, *intro integral-unique fundamental-theorem-of-calculus-real*)
(*auto intro*!: *derivative-eq-intros*)

lemma *integral-one-over-x-ln-x*:
$a \leq b \Longrightarrow a > 1 \Longrightarrow$ *integral* $\{a..b\}$ $(\lambda x.\ inverse\ (x * ln\ x) :: real) = ln\ (ln\ b) - ln\ (ln\ a)$
by (*intro integral-unique fundamental-theorem-of-calculus-real*)
(*auto intro*!: *derivative-eq-intros simp*: *field-simps*)

lemma *akra-bazzi-integral-kurzweil-henstock*:
akra-bazzi-integral $(\lambda f\ a\ b.\ f$ *integrable-on* $\{a..b\})$ $(\lambda f\ a\ b.$ *integral* $\{a..b\}\ f)$
apply *unfold-locales*
apply (*rule integrable-const-ivl*)
apply *simp*
apply (*erule integrable-subinterval-real*, *simp*)
apply (*blast intro*!: *integral-le*)
apply (*rule integral-combine*, *simp-all*) []
done

locale *master-theorem-function* = *akra-bazzi-recursion* +
fixes $g :: nat \Rightarrow real$
assumes *f-nonneg-base*: $x \geq x_0 \Longrightarrow x < x_1 \Longrightarrow f\ x \geq 0$
and *f-rec*: $x \geq x_1 \Longrightarrow f\ x = g\ x + (\sum i{<}k.\ as!i * f\ ((ts!i)\ x))$
and *g-nonneg*: $x \geq x_1 \Longrightarrow g\ x \geq 0$
and *ex-pos-a*: $\exists\, a{\in}set\ as.\ a > 0$
begin

interpretation *akra-bazzi-integral* $\lambda f\ a\ b.\ f$ *integrable-on* $\{a..b\}$ $\lambda f\ a\ b.$ *integral* $\{a..b\}\ f$
by (*rule akra-bazzi-integral-kurzweil-henstock*)

sublocale *akra-bazzi-function* $x_0\ x_1\ k\ as\ bs\ ts\ f$ $\lambda f\ a\ b.\ f$ *integrable-on* $\{a..b\}$
$\lambda f\ a\ b.$ *integral* $\{a..b\}\ f\ g$
using *f-nonneg-base f-rec g-nonneg ex-pos-a* **by** *unfold-locales*

context
begin

private lemma *g-nonneg'*: *eventually* $(\lambda x.\ g\ x \geq 0)$ *at-top*
using *g-nonneg* **by** (*force simp*: *eventually-at-top-linorder*)

```
private lemma g-pos:
  assumes g ∈ Ω(h)
  assumes eventually (λx. h x > 0) at-top
  shows   eventually (λx. g x > 0) at-top
proof−
  from landau-omega.bigE-nonneg[OF assms(1) g-nonneg'] guess c . note c =
this
  from assms(2) c(2) show ?thesis
    by eventually-elim (rule less-le-trans[OF mult-pos-pos[OF c(1)]], simp-all)
qed

private lemma f-pos:
  assumes g ∈ Ω(h)
  assumes eventually (λx. h x > 0) at-top
  shows   eventually (λx. f x > 0) at-top
  using g-pos[OF assms(1,2)] eventually-ge-at-top[of x1]
  by (eventually-elim) (subst f-rec, insert step-ge-x0,
       auto intro!: add-pos-nonneg setsum-nonneg mult-nonneg-nonneg[OF a-ge-0]
f-nonneg)

lemma bs-lower-bound: ∃ C>0. ∀ b∈set bs. C < b
proof (intro exI conjI ballI)
  from b-pos show A: Min (set bs) / 2 > 0 by auto
  fix b assume b: b ∈ set bs
  from A have Min (set bs) / 2 < Min (set bs) by simp
  also from b have ... ≤ b by simp
  finally show Min (set bs) / 2 < b .
qed

private lemma powr-growth2:
  ∃ C c2. 0 < c2 ∧ C < Min (set bs) ∧
      eventually (λx. ∀ u∈{C ∗ x..x}. c2 ∗ x powr p' ≥ u powr p') at-top
proof (intro exI conjI allI ballI)
  def C ≡ Min (set bs) / 2
  from b-bounds bs-nonempty have C-pos: C > 0 unfolding C-def by auto
  thus C < Min (set bs) unfolding C-def by simp
  show max (C powr p') 1 > 0 by simp
  show eventually (λx. ∀ u∈{C ∗ x..x}.
    max ((Min (set bs)/2) powr p') 1 ∗ x powr p' ≥ u powr p') at-top
    using eventually-gt-at-top[of 0::real] apply eventually-elim
  proof clarify
    fix x u assume x: x > 0 and u ∈ {C∗x..x}
    hence u: u ≥ C∗x u ≤ x unfolding C-def by  simp-all
    from u have u powr p' ≤ max ((C∗x) powr p') (x powr p') using C-pos x
      by (intro powr-upper-bound mult-pos-pos) simp-all
   also from u x C-pos have max ((C∗x) powr p') (x powr p') = x powr p' ∗ max
(C powr p') 1
      by (subst max-mult-left) (simp-all add: powr-mult algebra-simps)
```

```
    finally show u powr p' ≤ max ((Min (set bs)/2) powr p') 1 * x powr p'
      by (simp add: C-def algebra-simps)
  qed
qed

private lemma powr-growth1:
  ∃ C c1. 0 < c1 ∧ C < Min (set bs) ∧
      eventually (λx. ∀ u∈{C * x..x}. c1 * x powr p' ≤ u powr p') at-top
proof (intro exI conjI allI ballI)
  def C ≡ Min (set bs) / 2
  from b-bounds bs-nonempty have C-pos: C > 0 unfolding C-def by auto
  thus C < Min (set bs) unfolding C-def by simp
  from C-pos show min (C powr p') 1 > 0 by simp
  show eventually (λx. ∀ u∈{C * x..x}.
          min ((Min (set bs)/2) powr p') 1 * x powr p' ≤ u powr p') at-top
    using eventually-gt-at-top[of 0::real] apply eventually-elim
  proof clarify
    fix x u assume x: x > 0 and u ∈ {C*x..x}
    hence u: u ≥ C*x u ≤ x unfolding C-def by  simp-all
    from u x C-pos have x powr p' * min (C powr p') 1 = min ((C*x) powr p')
(x powr p')
      by (subst min-mult-left) (simp-all add: powr-mult algebra-simps)
    also from u have u powr p' ≥ min ((C*x) powr p') (x powr p') using C-pos x
      by (intro powr-lower-bound mult-pos-pos) simp-all
    finally show u powr p' ≥ min ((Min (set bs)/2) powr p') 1 * x powr p'
      by (simp add: C-def algebra-simps)
  qed
qed

private lemma powr-ln-powr-lower-bound:
  a > 1 ⟹ a ≤ x ⟹ x ≤ b ⟹
     min (a powr p) (b powr p) * min (ln a powr p') (ln b powr p') ≤ x powr p *
ln x powr p'
 using assms by (intro mult-mono powr-lower-bound) (auto intro: min.coboundedI1)

private lemma powr-ln-powr-upper-bound:
  a > 1 ⟹ a ≤ x ⟹ x ≤ b ⟹
     max (a powr p) (b powr p) * max (ln a powr p') (ln b powr p') ≥ x powr p *
ln x powr p'
 using assms by (intro mult-mono powr-upper-bound) (auto intro: max.coboundedI1)

private lemma powr-ln-powr-upper-bound':
  eventually (λa. ∀ b>a. ∃ c. ∀ x∈{a..b}. x powr p * ln x powr p' ≤ c) at-top
  by (subst eventually-at-top-dense) (force intro: powr-ln-powr-upper-bound)

private lemma powr-upper-bound':
  eventually (λa::real. ∀ b>a. ∃ c. ∀ x∈{a..b}. x powr p' ≤ c) at-top
  by (subst eventually-at-top-dense) (force intro: powr-upper-bound)
```

lemmas *bounds* =
powr-ln-powr-lower-bound powr-ln-powr-upper-bound powr-ln-powr-upper-bound′ powr-upper-bound′

private lemma *eventually-ln-const*:
assumes $(C::real) > 0$
shows *eventually* $(\lambda x.\ ln\ (C*x)\ /\ ln\ x > 1/2)$ *at-top*
proof−
from *tendstoD*[*OF tendsto-ln-over-ln*[*of C 1*], *of 1/2*] *assms*
have *eventually* $(\lambda x.\ |ln\ (C*x)\ /\ ln\ x\ -\ 1| < 1/2)$ *at-top* **by** (*simp add*: *dist-real-def*)
thus *?thesis* **by** *eventually-elim linarith*
qed

private lemma *powr-ln-powr-growth1*: $\exists\ C\ c1.\ 0 < c1 \wedge C < Min\ (set\ bs) \wedge$
eventually $(\lambda x.\ \forall u{\in}\{C * x..x\}.\ c1 * (x\ powr\ r * ln\ x\ powr\ r') \le u\ powr\ r * ln\ u\ powr\ r')$ *at-top*
proof (*intro exI conjI*)
let $?C = Min\ (set\ bs)\ /\ 2$ **and** $?f = \lambda x.\ x\ powr\ r * ln\ x\ powr\ r'$
def $C \equiv\ ?C$
from *b-bounds* **have** *C-pos*: $C > 0$ **unfolding** *C-def* **by** *simp*
let $?T = min\ (C\ powr\ r)\ (1\ powr\ r) * min\ ((1/2)\ powr\ r')\ (1\ powr\ r')$
from *C-pos* **show** $?T > 0$ **unfolding** *min-def* **by** (*auto split*: *if-split*)
from *bs-nonempty b-bounds* **have** *C-pos*: $C > 0$ **unfolding** *C-def* **by** *simp*
thus $C < Min\ (set\ bs)$ **by** (*simp add*: *C-def*)

show *eventually* $(\lambda x.\ \forall u{\in}\{C*x..x\}.\ ?T *\ ?f\ x \le\ ?f\ u)$ *at-top*
using *eventually-gt-at-top*[*of max 1* (*inverse C*)] *eventually-ln-const*[*OF C-pos*]
apply *eventually-elim*
proof *clarify*
fix *x u* **assume** *x*: $x > max\ 1\ (inverse\ C)$ **and** *u*: $u \in \{C*x..x\}$
hence *x′*: $x > 1$ **by** (*simp add*: *field-simps*)
with *C-pos* **have** *x-pos*: $x > 0$ **by** (*simp add*: *field-simps*)
from *x u C-pos* **have** *u′*: $u > 1$ **by** (*simp add*: *field-simps*)
assume *A*: $ln\ (C*x)\ /\ ln\ x > 1/2$
have $min\ (C\ powr\ r)\ (1\ powr\ r) \le (u/x)\ powr\ r$
using *x u u′ C-pos* **by** (*intro powr-lower-bound*) (*simp-all add*: *field-simps*)
moreover {
note *A*
also from *C-pos x′ u u′* **have** $ln\ (C*x) \le ln\ u$ **by** (*subst ln-le-cancel-iff*) *simp-all*
with *x′* **have** $ln\ (C*x)\ /\ ln\ x \le ln\ u\ /\ ln\ x$ **by** (*simp add*: *field-simps*)
finally have $min\ ((1/2)\ powr\ r')\ (1\ powr\ r') \le (ln\ u\ /\ ln\ x)\ powr\ r'$
using *x u u′ C-pos A* **by** (*intro powr-lower-bound*) *simp-all*
}
ultimately have $?T \le (u/x)\ powr\ r * (ln\ u\ /\ ln\ x)\ powr\ r'$
using *x-pos* **by** (*intro mult-mono*) *simp-all*
also from *x u u′* **have** ... = $?f\ u\ /\ ?f\ x$ **by** (*simp add*: *powr-divide*)

finally show *?T* ∗ *?f x* ≤ *?f u* **using** *x′* **by** (*simp add*: *field-simps*)
qed
qed

private lemma *powr-ln-powr-growth2*: ∃ *C c1*. *0* < *c1* ∧ *C* < *Min* (*set bs*) ∧
eventually (λ*x*. ∀ *u*∈{*C* ∗ *x*..*x*}. *c1* ∗ (*x powr r* ∗ *ln x powr r′*) ≥ *u powr r* ∗ *ln u powr r′*) *at-top*
proof (*intro exI conjI*)
let *?C* = *Min* (*set bs*) / *2* **and** *?f* = λ*x*. *x powr r* ∗ *ln x powr r′*
def *C* ≡ *?C*
let *?T* = *max* (*C powr r*) (*1 powr r*) ∗ *max* ((*1/2*) *powr r′*) (*1 powr r′*)
show *?T* > *0* **by** *simp*
from *b-bounds bs-nonempty* **have** *C-pos*: *C* > *0* **unfolding** *C-def* **by** *simp*
thus *C* < *Min* (*set bs*) **by** (*simp add*: *C-def*)

show *eventually* (λ*x*. ∀ *u*∈{*C*∗*x*..*x*}. *?T* ∗ *?f x* ≥ *?f u*) *at-top*
using *eventually-gt-at-top*[*of max 1* (*inverse C*)] *eventually-ln-const*[*OF C-pos*]
apply *eventually-elim*
proof *clarify*
fix *x u* **assume** *x*: *x* > *max 1* (*inverse C*) **and** *u*: *u* ∈ {*C*∗*x*..*x*}
hence *x′*: *x* > *1* **by** (*simp add*: *field-simps*)
with *C-pos* **have** *x-pos*: *x* > *0* **by** (*simp add*: *field-simps*)
from *x u C-pos* **have** *u′*: *u* > *1* **by** (*simp add*: *field-simps*)
assume *A*: *ln* (*C*∗*x*) / *ln x* > *1/2*
from *x u u′* **have** *?f u* / *?f x* = (*u/x*) *powr r* ∗ (*ln u/ln x*) *powr r′* **by** (*simp add*: *powr-divide*)
also {
have (*u/x*) *powr r* ≤ *max* (*C powr r*) (*1 powr r*)
using *x u u′ C-pos* **by** (*intro powr-upper-bound*) (*simp-all add*: *field-simps*)
moreover {
note *A*
also from *C-pos x′ u u′* **have** *ln* (*C*∗*x*) ≤ *ln u* **by** (*subst ln-le-cancel-iff*) *simp-all*
with *x′* **have** *ln* (*C*∗*x*) / *ln x* ≤ *ln u* / *ln x* **by** (*simp add*: *field-simps*)
finally have (*ln u* / *ln x*) *powr r′* ≤ *max* ((*1/2*) *powr r′*) (*1 powr r′*)
using *x u u′ C-pos A* **by** (*intro powr-upper-bound*) *simp-all*
} **ultimately have** (*u/x*) *powr r* ∗ (*ln u* / *ln x*) *powr r′* ≤ *?T*
using *x-pos* **by** (*intro mult-mono*) *simp-all*
}
finally show *?T* ∗ *?f x* ≥ *?f u* **using** *x′* **by** (*simp add*: *field-simps*)
qed
qed

lemmas *growths* = *powr-growth1 powr-growth2 powr-ln-powr-growth1 powr-ln-powr-growth2*

private lemma *master-integrable*:
∃ *a*::*real*. ∀ *b*≥*a*. (λ*u*. *u powr r* ∗ *ln u powr s* / *u powr t*) *integrable-on* {*a*..*b*}
∃ *a*::*real*. ∀ *b*≥*a*. (λ*u*. *u powr r* / *u powr s*) *integrable-on* {*a*..*b*}

by (*rule exI*[*of* - *2*], *force intro*!: *integrable-continuous-real continuous-intros*)+

private lemma *master-integral*:
fixes *a p p'* :: *real*
assumes *p*: $p \neq p'$ **and** *a*: $a > 0$
obtains *c d* **where** $c \neq 0$ $p > p' \longrightarrow d \neq 0$
$(\lambda x{::}nat.\ x\ powr\ p * (1 + integral\ \{a..x\}\ (\lambda u.\ u\ powr\ p' / u\ powr\ (p+1)))) \in$
$\Theta(\lambda x{::}nat.\ d * x\ powr\ p + c * x\ powr\ p')$
proof−
def $e \equiv a\ powr\ (p' - p)$
from *assms* **have** *e*: $e \geq 0$ **by** (*simp add*: *e-def*)
def $c \equiv inverse\ (p' - p)$ **and** $d \equiv 1 - inverse\ (p' - p) * e$
have $c \neq 0$ **and** $p > p' \longrightarrow d \neq 0$
using *e p a* **unfolding** *c-def d-def* **by** (*auto simp*: *field-simps*)
thus *?thesis*
apply (*rule that*) **apply** (*rule bigtheta-real-nat-transfer*, *rule bigthetaI-cong*)
using *eventually-ge-at-top*[*of a*]
proof *eventually-elim*
fix *x* **assume** *x*: $x \geq a$
hence $integral\ \{a..x\}\ (\lambda u.\ u\ powr\ p' / u\ powr\ (p+1)) =$
$integral\ \{a..x\}\ (\lambda u.\ u\ powr\ (p' - (p + 1)))$
by (*intro integral-cong*) (*simp-all add*: *powr-divide2*)
also have ... $= inverse\ (p' - p) * (x\ powr\ (p' - p) - a\ powr\ (p' - p))$
using *p x0-less-x1 a x* **by** (*simp add*: *integral-powr*)
also have $x\ powr\ p * (1 + ...) = d * x\ powr\ p + c * x\ powr\ p'$
using *p* **unfolding** *c-def d-def* **by** (*simp add*: *algebra-simps powr-divide2*[*symmetric*] *e-def*)
finally show $x\ powr\ p * (1 + integral\ \{a..x\}\ (\lambda u.\ u\ powr\ p' / u\ powr\ (p+1)))$ $=$
$d * x\ powr\ p + c * x\ powr\ p'$ **.**
qed
qed

private lemma *master-integral'*:
fixes *a p p'* :: *real*
assumes *p'*: $p' \neq 0$ **and** *a*: $a > 1$
obtains *c d* :: *real* **where** $p' < 0 \longrightarrow c \neq 0$ $d \neq 0$
$(\lambda x{::}nat.\ x\ powr\ p * (1 + integral\ \{a..x\}\ (\lambda u.\ u\ powr\ p * ln\ u\ powr\ (p'-1) / u\ powr\ (p+1)))) \in$
$\Theta(\lambda x{::}nat.\ c * x\ powr\ p + d * x\ powr\ p * ln\ x\ powr\ p')$
proof−
def $e \equiv ln\ a\ powr\ p'$
from *assms* **have** *e*: $e > 0$ **by** (*simp add*: *e-def*)
def $c \equiv 1 - inverse\ p' * e$ **and** $d \equiv inverse\ p'$
from *assms e* **have** $p' < 0 \longrightarrow c \neq 0$ $d \neq 0$ **unfolding** *c-def d-def* **by** (*auto simp*: *field-simps*)
thus *?thesis*
apply (*rule that*) **apply** (*rule landau-real-nat-transfer*, *rule bigthetaI-cong*)
using *eventually-ge-at-top*[*of a*]

proof *eventually-elim*
fix *x* :: *real* **assume** *x*: $x \geq a$
have *integral* $\{a..x\}$ ($\lambda u.$ *u powr p* $*$ *ln u powr* $(p' - 1)$ / *u powr* $(p + 1)$) =
integral $\{a..x\}$ ($\lambda u.$ *ln u powr* $(p' - 1)$ / *u*) **using** *x a x0-less-x1*
by (*intro integral-cong*) (*simp-all add*: *powr-add*)
also have ... = *inverse* p' $*$ (*ln x powr* p' $-$ *ln a powr* p')
using p' *x0-less-x1 a*(*1*) *x* **by** (*simp add*: *integral-ln-powr-over-x*)
also have *x powr p* $*$ (*1* + ...) = *c* $*$ *x powr p* + *d* $*$ *x powr p* $*$ *ln x powr* p'
using p' **by** (*simp add*: *algebra-simps c-def d-def e-def*)
finally show *x powr p* $*$ (*1*+*integral* $\{a..x\}$ ($\lambda u.$ *u powr p* $*$ *ln u powr* (p'−*1*) / *u powr* (*p*+*1*))) =
c $*$ *x powr p* + *d* $*$ *x powr p* $*$ *ln x powr* p' **.**
qed
qed

private lemma *master-integral''*:
fixes *a p* p' :: *real*
assumes *a*: $a > 1$
shows (λx::*nat*. *x powr p* $*$ (*1* + *integral* $\{a..x\}$ ($\lambda u.$ *u powr p* $*$ *ln u powr* − *1*/*u powr* (*p*+*1*)))) $\in$
Θ(λx::*nat*. *x powr p* $*$ *ln* (*ln x*))
proof (*rule landau-real-nat-transfer*)
have (λx::*real*. *x powr p* $*$ (*1* + *integral* $\{a..x\}$ ($\lambda u.$ *u powr p* $*$ *ln u powr* − *1*/*u powr* (*p*+*1*)))) $\in$
Θ(λx::*real*. (*1* − *ln* (*ln a*)) $*$ *x powr p* + *x powr p* $*$ *ln* (*ln x*)) (**is** *?f* $\in$ -)
apply (*rule bigthetaI-cong*) **using** *eventually-ge-at-top*[*of a*]
proof *eventually-elim*
fix *x* **assume** *x*: $x \geq a$
have *integral* $\{a..x\}$ ($\lambda u.$ *u powr p* $*$ *ln u powr* −*1* / *u powr* (*p* + *1*)) =
integral $\{a..x\}$ ($\lambda u.$ *inverse* (*u* $*$ *ln u*)) **using** *x a x0-less-x1*
by (*intro integral-cong*) (*simp-all add*: *powr-add powr-minus field-simps*)
also have ... = *ln* (*ln x*) − *ln* (*ln a*)
using *x0-less-x1 a*(*1*) *x* **by** (*subst integral-one-over-x-ln-x*) *simp-all*
also have *x powr p* $*$ (*1* + ...) = (*1* − *ln* (*ln a*)) $*$ *x powr p* + *x powr p* $*$ *ln* (*ln x*)
by (*simp add*: *algebra-simps*)
finally show *x powr p* $*$ (*1* + *integral* $\{a..x\}$ ($\lambda u.$ *u powr p* $*$ *ln u powr* − *1* / *u powr* (*p*+*1*))) =
(*1* − *ln* (*ln a*)) $*$ *x powr p* + *x powr p* $*$ *ln* (*ln x*) **.**
qed
also have ($\lambda x.$ (*1* − *ln* (*ln a*)) $*$ *x powr p* + *x powr p* $*$ *ln* (*ln x*)) $\in$
Θ($\lambda x.$ *x powr p* $*$ *ln* (*ln x*)) **by** *simp*
finally show *?f* $\in$ Θ($\lambda a.$ *a powr p* $*$ *ln* (*ln a*)) **.**
qed

lemma *master1-bigo*:
assumes *g-bigo*: *g* $\in$ *O*($\lambda x.$ *real x powr* p')

assumes *less-p'*: $(\sum i{<}k.\ as!i * bs!i\ powr\ p') > 1$
shows $f \in O(\lambda x.\ real\ x\ powr\ p)$
proof−
interpret *akra-bazzi-upper* x_0 x_1 *k as bs ts f*
$\lambda f\ a\ b.\ f$ *integrable-on* $\{a..b\}$ $\lambda f\ a\ b.$ *integral* $\{a..b\}$ *f g* $\lambda x.\ x\ powr\ p'$
using *assms growths g-bigo master-integrable* **by** *unfold-locales* (*assumption* | *simp*)+
from *less-p'* **have** *less-p*: $p' < p$ **by** (*rule p-greaterI*)
from *bigo-f*[*of 0*] **guess** *a* **. note** *a = this*
note *a*(*2*)
also from *a*(*1*) *less-p x0-less-x1* **have** $p \neq p'$ **by** *simp-all*
from *master-integral*[*OF this a*(*1*)] **guess** *c d* **. note** *cd = this*
note *cd*(*3*)
also from *cd*(*1*,*2*) *less-p*
have $(\lambda x{::}nat.\ d * real\ x\ powr\ p + c * real\ x\ powr\ p') \in \Theta(\lambda x.\ real\ x\ powr\ p)$ **by** *force*
finally show $f \in O(\lambda x{::}nat.\ x\ powr\ p)$ **.**
qed

lemma *master1*:
assumes *g-bigo*: $g \in O(\lambda x.\ real\ x\ powr\ p')$
assumes *less-p'*: $(\sum i{<}k.\ as!i * bs!i\ powr\ p') > 1$
assumes *f-pos*: *eventually* $(\lambda x.\ f\ x > 0)$ *at-top*
shows $f \in \Theta(\lambda x.\ real\ x\ powr\ p)$
proof (*rule bigthetaI*)
interpret *akra-bazzi-lower* x_0 x_1 *k as bs ts f*
$\lambda f\ a\ b.\ f$ *integrable-on* $\{a..b\}$ $\lambda f\ a\ b.$ *integral* $\{a..b\}$ *f g* λ-. *0*
using *assms*(*1*,*3*) *bs-lower-bound* **by** *unfold-locales* (*auto intro*: *always-eventually*)
from *bigomega-f* **show** $f \in \Omega(\lambda x.\ real\ x\ powr\ p)$ **by** *force*
qed (*fact master1-bigo*[*OF g-bigo less-p'*])

lemma *master2-3*:
assumes *g-bigtheta*: $g \in \Theta(\lambda x.\ real\ x\ powr\ p * ln\ (real\ x)\ powr\ (p' - 1))$
assumes *p'*: $p' > 0$
shows $f \in \Theta(\lambda x.\ real\ x\ powr\ p * ln\ (real\ x)\ powr\ p')$
proof−
have *eventually* $(\lambda x{::}real.\ x\ powr\ p * ln\ x\ powr\ (p' - 1) > 0)$ *at-top*
using *eventually-gt-at-top*[*of 1::real*] **by** *eventually-elim simp*
hence *eventually* $(\lambda x.\ f\ x > 0)$ *at-top*
by (*rule f-pos*[*OF bigthetaD2*[*OF g-bigtheta*] *eventually-nat-real*])
then interpret *akra-bazzi* x_0 x_1 *k as bs ts f*
$\lambda f\ a\ b.\ f$ *integrable-on* $\{a..b\}$ $\lambda f\ a\ b.$ *integral* $\{a..b\}$ *f g* $\lambda x.\ x\ powr\ p * ln\ x\ powr\ (p' - 1)$
using *assms growths bounds master-integrable* **by** *unfold-locales* (*assumption* | *simp*)+
from *bigtheta-f*[*of 1*] **guess** *a* **. note** *a = this*
note *a*(*2*)
also from *a*(*1*) *p'* **have** $p' \neq 0$ **by** *simp-all*

from *master-integral'*[*OF this a*(*1*), *of p*] **guess** *c d* **. note** *cd* = *this*
note *cd*(*3*)
also have (λx::*nat*. *c* $*$ *real x powr p* + *d* $*$ *real x powr p* $*$ *ln* (*real x*) *powr p'*) $\in$
Θ(λx::*nat*. *x powr p* $*$ *ln x powr p'*) **using** *cd*(*1*,*2*) *p'* **by** *force*
finally show *f* $\in$ Θ(λx. *real x powr p* $*$ *ln* (*real x*) *powr p'*) **.**
qed

lemma *master2-1*:
assumes *g-bigtheta*: *g* $\in$ Θ(λx. *real x powr p* $*$ *ln* (*real x*) *powr p'*)
assumes *p'*: *p'* < −*1*
shows *f* $\in$ Θ(λx. *real x powr p*)
proof−
have *eventually* (λx::*real*. *x powr p* $*$ *ln x powr p'* > *0*) *at-top*
using *eventually-gt-at-top*[*of 1*::*real*] **by** *eventually-elim simp*
hence *eventually* (λx. *f x* > *0*) *at-top*
by (*rule f-pos*[*OF bigthetaD2*[*OF g-bigtheta*] *eventually-nat-real*])
then interpret *akra-bazzi* x_0 x_1 *k as bs ts f*
λ*f a b*. *f integrable-on* {*a*..*b*} λ*f a b*. *integral* {*a*..*b*} *f g* λx. *x powr p* $*$ *ln x powr p'*
using *assms growths bounds master-integrable* **by** *unfold-locales* (*assumption* | *simp*)+
from *bigtheta-f*[*of 1*] **guess** *a* **. note** *a* = *this*
note *a*(*2*)
also from *a*(*1*) *p'* **have** *A*: *p'* + *1* ≠ *0* **by** *simp-all*
obtain *c d* :: *real* **where** *cd*: *c* ≠ *0* *d* ≠ *0* **and**
(λx::*nat*. *x powr p* $*$ (*1* + *integral* {*a*..*x*} (λu. *u powr p* $*$ *ln u powr p'*/ *u powr* (*p*+*1*)))) $\in$
Θ(λx::*nat*. *c* $*$ *x powr p* + *d* $*$ *x powr p* $*$ *ln x powr* (*p'* + *1*))
by (*rule master-integral'*[*OF A a*(*1*), *of p*]) (*insert p'*, *simp*)
note *this*(*3*)
also have (λx::*nat*. *c* $*$ *real x powr p* + *d* $*$ *real x powr p* $*$ *ln* (*real x*) *powr* (*p'* + *1*)) $\in$
Θ(λx::*nat*. *x powr p*) **using** *cd*(*1*,*2*) *p'* **by** *force*
finally show *f* $\in$ Θ(λx::*nat*. *x powr p*) **.**
qed

lemma *master2-2*:
assumes *g-bigtheta*: *g* $\in$ Θ(λx. *real x powr p* / *ln* (*real x*))
shows *f* $\in$ Θ(λx. *real x powr p* $*$ *ln* (*ln* (*real x*)))
proof−
have *eventually* (λx::*real*. *x powr p* / *ln x* > *0*) *at-top*
using *eventually-gt-at-top*[*of 1*::*real*] **by** *eventually-elim simp*
hence *eventually* (λx. *f x* > *0*) *at-top*
by (*rule f-pos*[*OF bigthetaD2*[*OF g-bigtheta*] *eventually-nat-real*])
moreover from *g-bigtheta* **have** *g-bigtheta'*: *g* $\in$ Θ(λx. *real x powr p* $*$ *ln* (*real x*) *powr* −*1*)
by (*rule landau-theta.trans*, *intro landau-real-nat-transfer*) *simp*
ultimately interpret *akra-bazzi* x_0 x_1 *k as bs ts f*

$\lambda f\ a\ b.\ f$ *integrable-on* $\{a..b\}$ $\lambda f\ a\ b.$ *integral* $\{a..b\}$ $f\ g$ $\lambda x.\ x$ *powr* $p * ln\ x$ *powr* -1
using *assms growths bounds master-integrable* **by** *unfold-locales* (*assumption* | *simp*)+
from *bigtheta-f*[*of 1*] **guess** *a* **. note** $a = this$
note $a(2)$
also note *master-integral''*[*OF* $a(1)$]
finally show $f \in \Theta(\lambda x{::}nat.\ x\ powr\ p * ln\ (ln\ x))$ **.**
qed

lemma *master3*:
assumes *g-bigtheta*: $g \in \Theta(\lambda x.\ real\ x\ powr\ p')$
assumes *p'-greater'*: $(\sum i{<}k.\ as!i * bs!i\ powr\ p') < 1$
shows $f \in \Theta(\lambda x.\ real\ x\ powr\ p')$
proof−
have *eventually* $(\lambda x{::}real.\ x\ powr\ p' > 0)$ *at-top*
using *eventually-gt-at-top*[*of 1::real*] **by** *eventually-elim simp*
hence *eventually* $(\lambda x.\ f\ x > 0)$ *at-top*
by (*rule f-pos*[*OF bigthetaD2*[*OF g-bigtheta*] *eventually-nat-real*])
then interpret *akra-bazzi* x_0 x_1 *k as bs ts f*
$\lambda f\ a\ b.\ f$ *integrable-on* $\{a..b\}$ $\lambda f\ a\ b.$ *integral* $\{a..b\}$ $f\ g$ $\lambda x.\ x\ powr\ p'$
using *assms growths bounds master-integrable* **by** *unfold-locales* (*assumption* | *simp*)+
from *p'-greater'* **have** *p'-greater*: $p' > p$ **by** (*rule p-lessI*)
from *bigtheta-f*[*of 0*] **guess** *a* **. note** $a = this$
note $a(2)$
also from *p'-greater* **have** $p \neq p'$ **by** *simp*
from *master-integral*[*OF this* $a(1)$] **guess** *c d* **. note** $cd = this$
note $cd(3)$
also have $(\lambda x{::}nat.\ d * x\ powr\ p + c * x\ powr\ p') \in \Theta(\lambda x{::}real.\ x\ powr\ p')$
using *p'-greater* $cd(1,2)$ **by** *force*
finally show $f \in \Theta(\lambda x.\ real\ x\ powr\ p')$ **.**
qed

end
end

end

6 Evaluating expressions with rational numerals

theory *Eval-Numeral*
imports
Complex-Main
begin

lemma *real-numeral-to-Ratreal*:
$(0{::}real) = Ratreal\ (Frct\ (0,\ 1))$
$(1{::}real) = Ratreal\ (Frct\ (1,\ 1))$

$(numeral\ x :: real) = Ratreal\ (Frct\ (numeral\ x,\ 1))$
$(1::int) = numeral\ Num.One$
by (*simp-all add*: *rat-number-collapse*)

lemma *real-equals-code*: $Ratreal\ x = Ratreal\ y \longleftrightarrow x = y$
by *simp*

lemma *Rat-normalize-idempotent*: $Rat.normalize\ (Rat.normalize\ x) = Rat.normalize\ x$
apply (*cases Rat.normalize x*)
using *Rat.normalize-stable*[*OF normalize-denom-pos normalize-coprime*] **apply** *auto*
done

lemma *uminus-pow-Numeral1*: $(-(x::-::monoid\text{-}mult))\ \hat{}\ Numeral1 = -x$ **by** *simp*

lemmas *power-numeral-simps* = *power-0 uminus-pow-Numeral1 power-minus-Bit0 power-minus-Bit1*

lemma *Fract-normalize*: $Fract\ (fst\ (Rat.normalize\ (x,y)))\ (snd\ (Rat.normalize\ (x,y))) = Fract\ x\ y$
by (*rule quotient-of-inject*) (*simp add*: *quotient-of-Fract Rat-normalize-idempotent*)

lemma *Frct-add*: $Frct\ (a,\ numeral\ b) + Frct\ (c,\ numeral\ d) =$
$Frct\ (Rat.normalize\ (a * numeral\ d + c * numeral\ b,\ numeral\ (b*d)))$
by (*auto simp*: *rat-number-collapse Fract-normalize*)

lemma *Frct-uminus*: $-(Frct\ (a,b)) = Frct\ (-a,b)$ **by** *simp*

lemma *Frct-diff*: $Frct\ (a,\ numeral\ b) - Frct\ (c,\ numeral\ d) =$
$Frct\ (Rat.normalize\ (a * numeral\ d - c * numeral\ b,\ numeral\ (b*d)))$
by (*auto simp*: *rat-number-collapse Fract-normalize*)

lemma *Frct-mult*: $Frct\ (a,\ numeral\ b) * Frct\ (c,\ numeral\ d) = Frct\ (a*c,\ numeral\ (b*d))$
by *simp*

lemma *Frct-inverse*: $inverse\ (Frct\ (a,\ b)) = Frct\ (b,\ a)$ **by** *simp*

lemma *Frct-divide*: $Frct\ (a,\ numeral\ b)\ /\ Frct\ (c,\ numeral\ d) = Frct\ (a*numeral\ d,\ numeral\ b * c)$
by *simp*

lemma *Frct-pow*: $Frct\ (a,\ numeral\ b)\ \hat{}\ c = Frct\ (a\ \hat{}\ c,\ numeral\ b\ \hat{}\ c)$
by (*induction c*) (*simp-all add*: *rat-number-collapse*)

lemma *Frct-less*: $Frct\ (a,\ numeral\ b) < Frct\ (c,\ numeral\ d) \longleftrightarrow a * numeral\ d$

< c * numeral b
  by *simp*

lemma *Frct-le*: *Frct (a, numeral b) ≤ Frct (c, numeral d) ⟷ a * numeral d ≤ c * numeral b*
  by *simp*

lemma *Frct-equals*: *Frct (a, numeral b) = Frct (c, numeral d) ⟷ a * numeral d = c * numeral b*
apply (*intro iffI antisym*)
apply (*subst Frct-le*[*symmetric*], *simp*)+
apply (*subst Frct-le*, *simp*)+
done

lemma *real-power-code*: (*Ratreal x*) ^ *y* = *Ratreal* (*x* ^ *y*) **by** (*simp add*: *of-rat-power*)

lemmas *real-arith-code* =
  real-plus-code real-minus-code real-times-code real-uminus-code real-inverse-code
  real-divide-code real-power-code real-less-code real-less-eq-code real-equals-code

lemmas *rat-arith-code* =
  Frct-add Frct-uminus Frct-diff Frct-mult Frct-inverse Frct-divide Frct-pow
  Frct-less Frct-le Frct-equals

lemma *one-to-numeral*: *1 = Numeral1* **by** *simp*

lemma *gcd-1-int'*: *gcd 1 x* = (*1::int*)
  by (*fact coprime-1-left*)

lemma *gcd-numeral-red*: *gcd* (*numeral x::int*) (*numeral y*) = *gcd* (*numeral y*) (*numeral x mod numeral y*)
  by (*fact gcd-red-int*)

lemma *divmod-one*:
  divmod (*Num.One*) (*Num.One*) = (*Numeral1*, *0*)
  divmod (*Num.One*) (*Num.Bit0 x*) = (*0*, *Numeral1*)
  divmod (*Num.One*) (*Num.Bit1 x*) = (*0*, *Numeral1*)
  divmod x (*Num.One*) = (*numeral x*, *0*)
  unfolding *divmod-def* **by** *simp-all*

lemmas *divmod-numeral-simps* =
  div-0 div-by-0 mod-0 mod-by-0
  semiring-numeral-div-class.fst-divmod [*symmetric*]
  semiring-numeral-div-class.snd-divmod [*symmetric*]
  divmod-cancel
  divmod-steps [*simplified rel-simps if-True*] *divmod-trivial*
  rel-simps

lemma *Suc-0-to-numeral*: *Suc 0* = *Numeral1* **by** *simp*
lemmas *Suc-to-numeral* = *Suc-0-to-numeral Num.Suc-1 Num.Suc-numeral*

lemma *rat-powr*:
0 powr y = *0*
x > *0* ⟹ *x powr Ratreal* (*Frct* (*0*, *Numeral1*)) = *Ratreal* (*Frct* (*Numeral1*, *Numeral1*))
x > *0* ⟹ *x powr Ratreal* (*Frct* (*numeral a*, *Numeral1*)) = *x* ˆ *numeral a*
x > *0* ⟹ *x powr Ratreal* (*Frct* (−*numeral a*, *Numeral1*)) = *inverse* (*x* ˆ *numeral a*)
by (*simp-all add*: *rat-number-collapse powr-numeral powr-minus*)

lemmas *eval-numeral-simps* =
real-numeral-to-Ratreal real-arith-code rat-arith-code Num.arith-simps
Rat.normalize-def fst-conv snd-conv gcd-0-int gcd-0-left-int gcd-1-int gcd-1-int′
gcd-neg1-int gcd-neg2-int gcd-numeral-red zmod-numeral-Bit0 zmod-numeral-Bit1 power-numeral-simps
divmod-numeral-simps one-to-numeral Groups.Let-0 Num.Let-numeral Suc-to-numeral power-numeral
greaterThanLessThan-iff atLeastAtMost-iff atLeastLessThan-iff greaterThanAtMost-iff rat-powr
Num.pow.simps Num.sqr.simps Product-Type.split of-int-numeral of-int-neg-numeral of-nat-numeral

ML ⟪
```
signature EVAL-NUMERAL =
sig
  val eval-numeral-tac : Proof.context -> int -> tactic
end

structure Eval-Numeral : EVAL-NUMERAL =
struct

fun eval-numeral-tac ctxt =
  let
    val ctxt' = put-simpset HOL-ss ctxt addsimps @{thms eval-numeral-simps}
  in
    SELECT-GOAL (SOLVE (Simplifier.simp-tac ctxt' 1))
  end

end
```
⟫

lemma *212543875486595895 12∗31421352363246435745388 4361∗2342523623324234∗5643274385872417347 4*
12561712738645824362329316482973164398214286 powr 2 /
(*113024631297842312 3*+*231212374631082764842731842∗1224743783894243623474512512 63*)
>
(*12313244512931247243543279768645745929475829310651205623844*::*real*)
by (*tactic* ⟪ *Eval-Numeral.eval-numeral-tac* @{*context*} *1* ⟫)

end

7 The proof methods

7.1 Master theorem and termination

theory *Akra-Bazzi-Method*
imports
Complex-Main
Akra-Bazzi
Master-Theorem
Eval-Numeral
begin

lemma *landau-symbol-ge-3-cong*:
assumes *landau-symbol L*
assumes $\bigwedge x::'a::linordered\text{-}semidom.\ x \geq 3 \Longrightarrow f\ x = g\ x$
shows $L(f) = L(g)$
apply (*rule landau-symbol.cong*[*OF assms*(*1*)])
apply (*subst eventually-at-top-linorder*, *rule exI*[*of - 3*], *simp add*: *assms*(*2*))
done

lemma *exp-1-lt-3*: *exp* (*1*::*real*) < *3*
proof−
from *taylor-up*[*of 3* λ-. *exp exp 0 1 0*]
obtain *t* :: *real* **where** *t* > *0 t* < *1 exp 1* = *5/2* + *exp t* / *6* **by** (*auto simp*: *eval-nat-numeral*)
note *this*(*3*)
also from ‹*t* < *1*› **have** *exp t* < *exp 1* **by** *simp*
finally show *exp* (*1*::*real*) < *3* **by** (*simp add*: *field-simps*)
qed

lemma *ln-ln-pos*:
assumes (*x*::*real*) ≥ *3*
shows *ln* (*ln x*) > *0*
proof (*subst ln-gt-zero-iff*)
from *assms exp-1-lt-3* **have** *ln x* > *ln* (*exp 1*) **by** (*intro ln-mono-strict*) *simp-all*
thus *ln x* > *0 ln x* > *1* **by** *simp-all*
qed

definition *akra-bazzi-terms* **where**
akra-bazzi-terms x_0 x_1 *bs ts* = (∀ *i*<*length bs*. *akra-bazzi-term* x_0 x_1 (*bs*!*i*) (*ts*!*i*))

lemma *akra-bazzi-termsI*:
(⋀*i*. *i* < *length bs* ⟹ *akra-bazzi-term* x_0 x_1 (*bs*!*i*) (*ts*!*i*)) ⟹ *akra-bazzi-terms* x_0 x_1 *bs ts*
unfolding *akra-bazzi-terms-def* **by** *blast*

lemma *master-theorem-functionI*:
  assumes $\forall x \in \{x_0..<x_1\}.\ f\ x \geq 0$
  assumes $\forall x \geq x_1.\ f\ x = g\ x + (\sum i<k.\ as\ !\ i * f\ ((ts\ !\ i)\ x))$
  assumes $\forall x \geq x_1.\ g\ x \geq 0$
  assumes $\forall a \in set\ as.\ a \geq 0$
  assumes *list-ex* $(\lambda a.\ a > 0)$ *as*
  assumes $\forall b \in set\ bs.\ b \in \{0<..<1\}$
  assumes $k \neq 0$
  assumes *length as* $= k$
  assumes *length bs* $= k$
  assumes *length ts* $= k$
  assumes *akra-bazzi-terms* $x_0\ x_1$ *bs ts*
  shows *master-theorem-function* $x_0\ x_1$ *k as bs ts f g*
using *assms* **unfolding** *akra-bazzi-terms-def* **by** *unfold-locales* (*auto simp*: *list-ex-iff*)

lemma *akra-bazzi-term-measure*:
  $x \geq x_1 \Longrightarrow$ *akra-bazzi-term* $0\ x_1\ b\ t \Longrightarrow (t\ x,\ x) \in$ *Wellfounded.measure* $(\lambda n::nat.\ n)$
  $x > x_1 \Longrightarrow$ *akra-bazzi-term* 0 $(Suc\ x_1)$ $b\ t \Longrightarrow (t\ x,\ x) \in$ *Wellfounded.measure* $(\lambda n::nat.\ n)$
  unfolding *akra-bazzi-term-def* **by** *auto*

lemma *measure-prod-conv*:
  $((a,\ b),\ (c,\ d)) \in$ *Wellfounded.measure* $(\lambda x.\ t\ (fst\ x)) \longleftrightarrow (a,\ c) \in$ *Wellfounded.measure t*
  $((e,\ f),\ (g,\ h)) \in$ *Wellfounded.measure* $(\lambda x.\ t\ (snd\ x)) \longleftrightarrow (f,\ h) \in$ *Wellfounded.measure t*
by *simp-all*

lemmas *measure-prod-conv'* = *measure-prod-conv*[**where** $t = \lambda x.\ x$]

lemma *akra-bazzi-termination-simps*:
  fixes $x :: nat$
  shows $a * real\ x\ /\ b = a/b * real\ x$ $real\ x\ /\ b = 1/b * real\ x$
  by *simp-all*

lemma *akra-bazzi-params-nonzeroI*:
  length as = *length bs* $\Longrightarrow$
  $(\forall a \in set\ as.\ a \geq 0) \Longrightarrow (\forall b \in set\ bs.\ b \in \{0<..<1\}) \Longrightarrow (\exists a \in set\ as.\ a > 0) \Longrightarrow$
  akra-bazzi-params-nonzero (*length as*) *as bs* **by** (*unfold-locales*, *simp-all*) []

lemmas *akra-bazzi-p-rel-intros* =
  akra-bazzi-params-nonzero.p-lessI[*rotated*, *OF* - *akra-bazzi-params-nonzeroI*]
  akra-bazzi-params-nonzero.p-greaterI[*rotated*, *OF* - *akra-bazzi-params-nonzeroI*]
  akra-bazzi-params-nonzero.p-leI[*rotated*, *OF* - *akra-bazzi-params-nonzeroI*]
  akra-bazzi-params-nonzero.p-geI[*rotated*, *OF* - *akra-bazzi-params-nonzeroI*]
  akra-bazzi-params-nonzero.p-boundsI[*rotated*, *OF* - *akra-bazzi-params-nonzeroI*]
  akra-bazzi-params-nonzero.p-boundsI'[*rotated*, *OF* - *akra-bazzi-params-nonzeroI*]

lemma *eval-length*: *length* [] = *0* *length* (*x* # *xs*) = *Suc* (*length xs*) **by** *simp-all*

lemma *eval-akra-bazzi-setsum*:
($\sum$ *i*<*0*. *as*!*i* ∗ *bs*!*i* *powr x*) = *0*
($\sum$ *i*<*Suc 0*. (*a*#*as*)!*i* ∗ (*b*#*bs*)!*i* *powr x*) = *a* ∗ *b* *powr x*
($\sum$ *i*<*Suc k*. (*a*#*as*)!*i* ∗ (*b*#*bs*)!*i* *powr x*) = *a* ∗ *b* *powr x* + ($\sum$ *i*<*k*. *as*!*i* ∗ *bs*!*i* *powr x*)
apply *simp*
apply *simp*
apply (*induction k arbitrary*: *a as b bs*)
apply *simp-all*
done

lemma *eval-akra-bazzi-setsum'*:
($\sum$ *i*<*0*. *as*!*i* ∗ *f* ((*ts*!*i*) *x*)) = *0*
($\sum$ *i*<*Suc 0*. (*a*#*as*)!*i* ∗ *f* (((*t*#*ts*)!*i*) *x*)) = *a* ∗ *f* (*t x*)
($\sum$ *i*<*Suc k*. (*a*#*as*)!*i* ∗ *f* (((*t*#*ts*)!*i*) *x*)) = *a* ∗ *f* (*t x*) + ($\sum$ *i*<*k*. *as*!*i* ∗ *f* ((*ts*!*i*) *x*))
apply *simp*
apply *simp*
apply (*induction k arbitrary*: *a as t ts*)
apply (*simp-all add*: *algebra-simps*)
done

lemma *akra-bazzi-termsI'*:
akra-bazzi-terms x_0 x_1 [] []
akra-bazzi-term x_0 x_1 *b t* ⟹ *akra-bazzi-terms* x_0 x_1 *bs ts* ⟹ *akra-bazzi-terms* x_0 x_1 (*b*#*bs*) (*t*#*ts*)
unfolding *akra-bazzi-terms-def* **using** *less-Suc-eq-0-disj* **by** *auto*

lemma *ball-set-intros*: (∀ *x*∈*set* []. *P x*) *P x* ⟹ (∀ *x*∈*set xs*. *P x*) ⟹ (∀ *x*∈*set* (*x*#*xs*). *P x*)
by *auto*

lemma *ball-set-simps*: (∀ *x*∈*set* []. *P x*) = *True* (∀ *x*∈*set* (*x*#*xs*). *P x*) = (*P x* ∧ (∀ *x*∈*set xs*. *P x*))
by *auto*

lemma *bex-set-simps*: (∃ *x*∈*set* []. *P x*) = *False* (∃ *x*∈*set* (*x*#*xs*). *P x*) = (*P x* ∨ (∃ *x*∈*set xs*. *P x*))
by *auto*

lemma *eval-akra-bazzi-le-list-ex*:
list-ex P (*x*#*y*#*xs*) ⟷ *P x* ∨ *list-ex P* (*y*#*xs*)
list-ex P [*x*] ⟷ *P x*
list-ex P [] ⟷ *False*
by (*auto simp*: *list-ex-iff*)

lemma *eval-akra-bazzi-le-listsum*:
  $x \le listsum\ [] \longleftrightarrow x \le 0$ $x \le listsum\ (y\#ys) \longleftrightarrow x \le y + listsum\ ys$
  $x \le z + listsum\ [] \longleftrightarrow x \le z$ $x \le z + listsum\ (y\#ys) \longleftrightarrow x \le z + y + listsum\ ys$
  by (*simp-all add*: *algebra-simps*)

lemma *atLeastLessThanE*: $x \in \{a..<b\} \Longrightarrow (x \ge a \Longrightarrow x < b \Longrightarrow P) \Longrightarrow P$ **by** *simp*

lemma *master-theorem-preprocess*:
  $\Theta(\lambda n::nat.\ 1) = \Theta(\lambda n::nat.\ real\ n\ powr\ 0)$
  $\Theta(\lambda n::nat.\ real\ n) = \Theta(\lambda n::nat.\ real\ n\ powr\ 1)$
  $O(\lambda n::nat.\ 1) = O(\lambda n::nat.\ real\ n\ powr\ 0)$
  $O(\lambda n::nat.\ real\ n) = O(\lambda n::nat.\ real\ n\ powr\ 1)$

  $\Theta(\lambda n::nat.\ ln\ (ln\ (real\ n))) = \Theta(\lambda n::nat.\ real\ n\ powr\ 0 * ln\ (ln\ (real\ n)))$
  $\Theta(\lambda n::nat.\ real\ n * ln\ (ln\ (real\ n))) = \Theta(\lambda n::nat.\ real\ n\ powr\ 1 * ln\ (ln\ (real\ n)))$

  $\Theta(\lambda n::nat.\ ln\ (real\ n)) = \Theta(\lambda n::nat.\ real\ n\ powr\ 0 * ln\ (real\ n)\ powr\ 1)$
  $\Theta(\lambda n::nat.\ real\ n * ln\ (real\ n)) = \Theta(\lambda n::nat.\ real\ n\ powr\ 1 * ln\ (real\ n)\ powr\ 1)$
  $\Theta(\lambda n::nat.\ real\ n\ powr\ p * ln\ (real\ n)) = \Theta(\lambda n::nat.\ real\ n\ powr\ p * ln\ (real\ n)\ powr\ 1)$
  $\Theta(\lambda n::nat.\ ln\ (real\ n)\ powr\ p') = \Theta(\lambda n::nat.\ real\ n\ powr\ 0 * ln\ (real\ n)\ powr\ p')$
  $\Theta(\lambda n::nat.\ real\ n * ln\ (real\ n)\ powr\ p') = \Theta(\lambda n::nat.\ real\ n\ powr\ 1 * ln\ (real\ n)\ powr\ p')$
apply (*simp-all*)
apply (*simp-all cong*: *landau-symbols*[*THEN landau-symbol-ge-3-cong*])?
done

lemma *akra-bazzi-term-imp-size-less*:
  $x_1 \le x \Longrightarrow akra\text{-}bazzi\text{-}term\ 0\ x_1\ b\ t \Longrightarrow size\ (t\ x) < size\ x$
  $x_1 < x \Longrightarrow akra\text{-}bazzi\text{-}term\ 0\ (Suc\ x_1)\ b\ t \Longrightarrow size\ (t\ x) < size\ x$
  by (*simp-all add*: *akra-bazzi-term-imp-less*)

definition $CLAMP\ (f :: nat \Rightarrow real)\ x = (if\ x < 3\ then\ 0\ else\ f\ x)$
definition $CLAMP'\ (f :: nat \Rightarrow real)\ x = (if\ x < 3\ then\ 0\ else\ f\ x)$
definition $MASTER\text{-}BOUND\ a\ b\ c\ x = real\ x\ powr\ a * ln\ (real\ x)\ powr\ b * ln\ (ln\ (real\ x))\ powr\ c$
definition $MASTER\text{-}BOUND'\ a\ b\ x = real\ x\ powr\ a * ln\ (real\ x)\ powr\ b$
definition $MASTER\text{-}BOUND''\ a\ x = real\ x\ powr\ a$

lemma *ln-1-imp-less-3*:
  $ln\ x = (1::real) \Longrightarrow x < 3$
proof−
  assume $ln\ x = 1$
  also have $(1::real) \le ln\ (exp\ 1)$ **by** *simp*
  finally have $ln\ x \le ln\ (exp\ 1)$ **by** *simp*
  hence $x \le exp\ 1$

by (*cases* $x > 0$) (*force simp del*: *ln-exp simp add*: *not-less intro*: *order.trans*)+
also have ... < 3 **by** (*rule exp-1-lt-3*)
finally show *?thesis* **.**
qed

lemma *ln-1-imp-less-3'*: $ln\ (real\ (x{::}nat)) = 1 \Longrightarrow x < 3$ **by** (*drule ln-1-imp-less-3*) *simp*

lemma *ln-ln-nonneg*: $x \geq (3{::}real) \Longrightarrow ln\ (ln\ x) \geq 0$ **using** *ln-ln-pos*[*of x*] **by** *simp*
lemma *ln-ln-nonneg'*: $x \geq (3{::}nat) \Longrightarrow ln\ (ln\ (real\ x)) \geq 0$ **using** *ln-ln-pos*[*of real x*] **by** *simp*

lemma *MASTER-BOUND-postproc*:
CLAMP (*MASTER-BOUND' a 0*) = *CLAMP* (*MASTER-BOUND'' a*)
CLAMP (*MASTER-BOUND' a 1*) = *CLAMP* (λx. *CLAMP* (*MASTER-BOUND'' a*) $x *$ *CLAMP* (λx. *ln* (*real x*)) *x*)
CLAMP (*MASTER-BOUND' a* (*numeral n*)) =
CLAMP (λx. *CLAMP* (*MASTER-BOUND'' a*) $x *$ *CLAMP* (λx. *ln* (*real x*) ^ *numeral n*) *x*)
CLAMP (*MASTER-BOUND' a* (-1)) =
CLAMP (λx. *CLAMP* (*MASTER-BOUND'' a*) x / *CLAMP* (λx. *ln* (*real x*)) *x*)
CLAMP (*MASTER-BOUND' a* ($-$*numeral n*)) =
CLAMP (λx. *CLAMP* (*MASTER-BOUND'' a*) x / *CLAMP* (λx. *ln* (*real x*) ^ *numeral n*) *x*)
CLAMP (*MASTER-BOUND' a b*) =
CLAMP (λx. *CLAMP* (*MASTER-BOUND'' a*) $x *$ *CLAMP* (λx. *ln* (*real x*) *powr b*) *x*)

CLAMP (*MASTER-BOUND'' 0*) = *CLAMP* (λx. *1*)
CLAMP (*MASTER-BOUND'' 1*) = *CLAMP* (λx. (*real x*))
CLAMP (*MASTER-BOUND''* (*numeral n*)) = *CLAMP* (λx. (*real x*) ^ *numeral n*)
CLAMP (*MASTER-BOUND''* (-1)) = *CLAMP* (λx. *1* / (*real x*))
CLAMP (*MASTER-BOUND''* ($-$*numeral n*)) = *CLAMP* (λx. *1* / (*real x*) ^ *numeral n*)
CLAMP (*MASTER-BOUND'' a*) = *CLAMP* (λx. (*real x*) *powr a*)

and *MASTER-BOUND-UNCLAMP*:
CLAMP (λx. *CLAMP f x* $*$ *CLAMP g x*) = *CLAMP* (λx. *f x* $*$ *g x*)
CLAMP (λx. *CLAMP f x* / *CLAMP g x*) = *CLAMP* (λx. *f x* / *g x*)
CLAMP (*CLAMP f*) = *CLAMP f*
unfolding *CLAMP-def*[*abs-def*] *MASTER-BOUND'-def*[*abs-def*] *MASTER-BOUND''-def*[*abs-def*]
by (*rule ext*, *simp add*: *powr-numeral powr-minus divide-inverse*)+

context
begin

private lemma *CLAMP-*: *landau-symbol L* $\Longrightarrow$ *L*(*f*::*nat* $\Rightarrow$ *real*) $\equiv$ *L*(λx. *CLAMP f x*)
  using *eventually-ge-at-top*[*of 3*::*nat*] **unfolding** *CLAMP-def*[*abs-def*]
  by (*intro landau-symbol.cong eq-reflection*) (*auto elim*!: *eventually-mono*)

private lemma *UNCLAMP'-*: *landau-symbol L* $\Longrightarrow$ *L*(*CLAMP'* (*MASTER-BOUND a b c*)) $\equiv$ *L*(*MASTER-BOUND a b c*)
  using *eventually-ge-at-top*[*of 3*::*nat*] **unfolding** *CLAMP'-def*[*abs-def*] *CLAMP-def*[*abs-def*]
  by (*auto intro*!: *landau-symbol.cong eq-reflection elim*!: *eventually-mono*)

private lemma *UNCLAMP-*: *landau-symbol L* $\Longrightarrow$ *L*(*CLAMP f*) $\equiv$ *L*(*f*)
  using *eventually-ge-at-top*[*of 3*::*nat*] **unfolding** *CLAMP'-def*[*abs-def*] *CLAMP-def*[*abs-def*]
  by (*auto intro*!: *landau-symbol.cong eq-reflection elim*!: *eventually-mono*)

lemmas *CLAMP* = *landau-symbols*[*THEN CLAMP-*]
lemmas *UNCLAMP'* = *landau-symbols*[*THEN UNCLAMP'-*]
lemmas *UNCLAMP* = *landau-symbols*[*THEN UNCLAMP-*]
end

lemma *propagate-CLAMP*:
  CLAMP (λx. *f x* $*$ *g x*) = *CLAMP'* (λx. *CLAMP f x* $*$ *CLAMP g x*)
  CLAMP (λx. *f x* / *g x*) = *CLAMP'* (λx. *CLAMP f x* / *CLAMP g x*)
  CLAMP (λx. *inverse* (*f x*)) = *CLAMP'* (λx. *inverse* (*CLAMP f x*))
  CLAMP (λx. *real x*) = *CLAMP'* (*MASTER-BOUND 1 0 0*)
  CLAMP (λx. *real x powr a*) = *CLAMP'* (*MASTER-BOUND a 0 0*)
  CLAMP (λx. *real x* ^ *a'*) = *CLAMP'* (*MASTER-BOUND* (*real a'*) *0 0*)
  CLAMP (λx. *ln* (*real x*)) = *CLAMP'* (*MASTER-BOUND 0 1 0*)
  CLAMP (λx. *ln* (*real x*) *powr b*) = *CLAMP'* (*MASTER-BOUND 0 b 0*)
  CLAMP (λx. *ln* (*real x*) ^ *b'*) = *CLAMP'* (*MASTER-BOUND 0* (*real b'*) *0*)
  CLAMP (λx. *ln* (*ln* (*real x*))) = *CLAMP'* (*MASTER-BOUND 0 0 1*)
  CLAMP (λx. *ln* (*ln* (*real x*)) *powr c*) = *CLAMP'* (*MASTER-BOUND 0 0 c*)
  CLAMP (λx. *ln* (*ln* (*real x*)) ^ *c'*) = *CLAMP'* (*MASTER-BOUND 0 0* (*real c'*))
  CLAMP' (*CLAMP f*) = *CLAMP' f*
  CLAMP' (λx. *CLAMP'* (*MASTER-BOUND a1 b1 c1*) *x* $*$ *CLAMP'* (*MASTER-BOUND a2 b2 c2*) *x*) =
      CLAMP' (*MASTER-BOUND* (*a1*+*a2*) (*b1*+*b2*) (*c1*+*c2*))
  CLAMP' (λx. *CLAMP'* (*MASTER-BOUND a1 b1 c1*) *x* / *CLAMP'* (*MASTER-BOUND a2 b2 c2*) *x*) =
      CLAMP' (*MASTER-BOUND* (*a1*−*a2*) (*b1*−*b2*) (*c1*−*c2*))
  CLAMP' (λx. *inverse* (*MASTER-BOUND a1 b1 c1 x*)) = *CLAMP'* (*MASTER-BOUND* (−*a1*) (−*b1*) (−*c1*))
by (*insert ln-1-imp-less-3'*)
   (*rule ext*, *simp add*: *CLAMP-def CLAMP'-def MASTER-BOUND-def*
     powr-realpow powr-one[*OF ln-ln-nonneg'*] *powr-realpow*[*OF ln-ln-pos*] *powr-add*
      powr-divide2[*symmetric*] *powr-minus*)+

lemma *numeral-assoc-simps*:
  ((*a*::*real*) + *numeral b*) + *numeral c* = *a* + *numeral* (*b* + *c*)

$(a + \textit{numeral } b) - \textit{numeral } c = a + \textit{neg-numeral-class.sub } b\ c$
$(a - \textit{numeral } b) + \textit{numeral } c = a + \textit{neg-numeral-class.sub } c\ b$
$(a - \textit{numeral } b) - \textit{numeral } c = a - \textit{numeral } (b + c)$ **by** *simp-all*

lemmas *CLAMP-aux* =
arith-simps numeral-assoc-simps of-nat-power of-nat-mult of-nat-numeral
one-add-one one-to-numeral

lemmas *CLAMP-postproc* = *numeral-One*

context *master-theorem-function*
begin

lemma *master1-bigo-automation*:
assumes $g \in O(\lambda x.\ \textit{real } x \textit{ powr } p')$ $1 < (\sum i{<}k.\ \textit{as} ! i * \textit{bs} ! i \textit{ powr } p')$
shows $f \in O(\textit{MASTER-BOUND } p\ 0\ 0)$
proof−
have $\textit{MASTER-BOUND } p\ 0\ 0 \in \Theta(\lambda x{::}\textit{nat}.\ x \textit{ powr } p)$ **unfolding** *MASTER-BOUND-def*[*abs-def*]
using *eventually-ge-at-top*[*of 3*::*real*]
by (*intro landau-real-nat-transfer*, *intro bigthetaI-cong*)
(*auto elim*!: *eventually-mono dest*!: *ln-1-imp-less-3*)
from *landau-o.big.cong-bigtheta*[*OF this*] *master1-bigo*[*OF assms*] **show** *?thesis*
by *simp*
qed

lemma *master1-automation*:
assumes $g \in O(\textit{MASTER-BOUND}''\ p')$ $1 < (\sum i{<}k.\ \textit{as} ! i * \textit{bs} ! i \textit{ powr } p')$
eventually $(\lambda x.\ f\ x > 0)$ *at-top*
shows $f \in \Theta(\textit{MASTER-BOUND } p\ 0\ 0)$
proof−
have *A*: $\textit{MASTER-BOUND } p\ 0\ 0 \in \Theta(\lambda x{::}\textit{nat}.\ x \textit{ powr } p)$ **unfolding** *MASTER-BOUND-def*[*abs-def*]
using *eventually-ge-at-top*[*of 3*::*real*]
by (*intro landau-real-nat-transfer*, *intro bigthetaI-cong*)
(*auto elim*!: *eventually-mono dest*!: *ln-1-imp-less-3*)
have *B*: $O(\textit{MASTER-BOUND}''\ p') = O(\lambda x{::}\textit{nat}.\ \textit{real } x \textit{ powr } p')$
using *eventually-ge-at-top*[*of 2*::*nat*]
by (*intro landau-o.big.cong*) (*auto elim*!: *eventually-mono simp*: *MASTER-BOUND''-def*)
from *landau-theta.cong-bigtheta*[*OF A*] *B assms*(*1*) *master1*[*OF - assms*(*2*−)]
show *?thesis* **by** *simp*
qed

lemma *master2-1-automation*:
assumes $g \in \Theta(\textit{MASTER-BOUND}'\ p\ p')$ $p' < -1$
shows $f \in \Theta(\textit{MASTER-BOUND } p\ 0\ 0)$
proof−
have *A*: $\textit{MASTER-BOUND } p\ 0\ 0 \in \Theta(\lambda x{::}\textit{nat}.\ x \textit{ powr } p)$ **unfolding** *MASTER-BOUND-def*[*abs-def*]
using *eventually-ge-at-top*[*of 3*::*real*]
by (*intro landau-real-nat-transfer*, *intro bigthetaI-cong*)
(*auto elim*!: *eventually-mono dest*!: *ln-1-imp-less-3*)

have B: $\Theta(MASTER\text{-}BOUND'\ p\ p') = \Theta(\lambda x{::}nat.\ real\ x\ powr\ p * ln\ (real\ x)\ powr\ p')$
by (*subst CLAMP*, (*subst MASTER-BOUND-postproc MASTER-BOUND-UNCLAMP*)+, *simp only*: *UNCLAMP*)
from *landau-theta.cong-bigtheta*[*OF A*] *B assms*(*1*) *master2-1*[*OF - assms*(*2*−)]
show *?thesis* **by** *simp*
qed

lemma *master2-2-automation*:
assumes $g \in \Theta(MASTER\text{-}BOUND'\ p\ (-1))$
shows $f \in \Theta(MASTER\text{-}BOUND\ p\ 0\ 1)$
proof−
have A: $MASTER\text{-}BOUND\ p\ 0\ 1 \in \Theta(\lambda x{::}nat.\ x\ powr\ p * ln\ (ln\ x))$ **unfolding** *MASTER-BOUND-def*[*abs-def*]
using *eventually-ge-at-top*[*of 3::real*]
apply (*intro landau-real-nat-transfer*, *intro bigthetaI-cong*)
apply (*elim eventually-mono*, *subst powr-one*[*OF ln-ln-nonneg*])
apply *simp-all*
done
have B: $\Theta(MASTER\text{-}BOUND'\ p\ (-1)) = \Theta(\lambda x{::}nat.\ real\ x\ powr\ p\ /\ ln\ (real\ x))$
by (*subst CLAMP*, (*subst MASTER-BOUND-postproc MASTER-BOUND-UNCLAMP*)+, *simp only*: *UNCLAMP*)
from *landau-theta.cong-bigtheta*[*OF A*] *B assms*(*1*) *master2-2* **show** *?thesis* **by** *simp*
qed

lemma *master2-3-automation*:
assumes $g \in \Theta(MASTER\text{-}BOUND'\ p\ (p' - 1))\ p' > 0$
shows $f \in \Theta(MASTER\text{-}BOUND\ p\ p'\ 0)$
proof−
have A: $MASTER\text{-}BOUND\ p\ p'\ 0 \in \Theta(\lambda x{::}nat.\ x\ powr\ p * ln\ x\ powr\ p')$ **unfolding** *MASTER-BOUND-def*[*abs-def*]
using *eventually-ge-at-top*[*of 3::real*]
apply (*intro landau-real-nat-transfer*, *intro bigthetaI-cong*)
apply (*elim eventually-mono*, *auto dest*: *ln-1-imp-less-3*)
done
have B: $\Theta(MASTER\text{-}BOUND'\ p\ (p' - 1)) = \Theta(\lambda x{::}nat.\ real\ x\ powr\ p * ln\ x\ powr\ (p' - 1))$
by (*subst CLAMP*, (*subst MASTER-BOUND-postproc MASTER-BOUND-UNCLAMP*)+, *simp only*: *UNCLAMP*)
from *landau-theta.cong-bigtheta*[*OF A*] *B assms*(*1*) *master2-3*[*OF - assms*(*2*−)]
show *?thesis* **by** *simp*
qed

lemma *master3-automation*:
assumes $g \in \Theta(MASTER\text{-}BOUND''\ p')\ 1 > (\sum i{<}k.\ as\ !\ i * bs\ !\ i\ powr\ p')$
shows $f \in \Theta(MASTER\text{-}BOUND\ p'\ 0\ 0)$
proof−

have *A*: *MASTER-BOUND p' 0 0* ∈ Θ(λ*x*::*nat*. *x powr p'*) **unfolding** *MASTER-BOUND-def*[*abs-def*]
  using *eventually-ge-at-top*[*of 3*::*real*]
  apply (*intro landau-real-nat-transfer*, *intro bigthetaI-cong*)
  apply (*elim eventually-mono*, *auto dest*: *ln-1-imp-less-3*)
  done
have *B*: Θ(*MASTER-BOUND'' p'*) = Θ(λ*x*::*nat*. *real x powr p'*)
  by (*subst CLAMP*, (*subst MASTER-BOUND-postproc*)+, *simp only*: *UN-CLAMP*)
from *landau-theta.cong-bigtheta*[*OF A*] *B assms*(*1*) *master3*[*OF - assms*(*2*−)]
show *?thesis* **by** *simp*
qed

lemmas *master-automation* =
  master1-automation master2-1-automation master2-2-automation
  master2-2-automation master3-automation

ML ⟪

```
fun generalize-master-thm ctxt thm =
  let
    val ([p'], ctxt') = Variable.variant-fixes [p''] ctxt
    val p' = Free (p', HOLogic.realT)
    val a = @{term nth as} $ Bound 0
    val b = @{term Transcendental.powr :: real => real => real} $
            (@{term nth bs} $ Bound 0) $ p'
    val f = Abs (i, HOLogic.natT, @{term op * :: real => real => real} $ a $ b)
    val setsum = @{term setsum :: (nat => real) => nat set => real} $ f $ @{term {..<k}}
    val prop = HOLogic.mk-Trueprop (HOLogic.mk-eq (setsum, @{term 1::real}))
    val cprop = Thm.cterm-of ctxt' prop
  in
    thm
    |> Local-Defs.unfold ctxt' [Thm.assume cprop RS @{thm p-unique}]
    |> Thm.implies-intr cprop
    |> rotate-prems 1
    |> singleton (Variable.export ctxt' ctxt)
  end

fun generalize-master-thm' (binding, thm) ctxt =
  Local-Theory.note ((binding, []), [generalize-master-thm ctxt thm]) ctxt |> snd
```

⟫

local-setup ⟪

```
  fold generalize-master-thm'
    [(@{binding master1-automation'}, @{thm master1-automation}),
     (@{binding master1-bigo-automation'}, @{thm master1-bigo-automation}),
     (@{binding master2-1-automation'}, @{thm master2-1-automation}),
```

(@{*binding master2-2-automation'*}, @{*thm master2-2-automation*}),
(@{*binding master2-3-automation'*}, @{*thm master2-3-automation*}),
(@{*binding master3-automation'*}, @{*thm master3-automation*})]
⟫

end

definition *arith-consts* $(x :: real)$ $(y :: nat)$ =
$(if\ \neg\ (-x) + 3\ /\ x * 5 - 1 \le x \wedge True \vee True \longrightarrow True\ then$
$x < inverse\ 3\ powr\ 21\ else\ x = real\ (Suc\ 0\ \hat{}\ 2 +$
$(if\ 42 - x \le 1 \wedge 1\ div\ y = y\ mod\ 2 \vee y < Numeral1\ then\ 0\ else\ 0)) + Numeral1)$

ML-file *akra-bazzi.ML*

hide-const *arith-consts*

method-setup *master-theorem* = ⟪
Akra-Bazzi.setup-master-theorem
⟫ *automatically apply the Master theorem for recursive functions*

method-setup *akra-bazzi-termination* = ⟪
Scan.succeed (fn ctxt => SIMPLE-METHOD' (Akra-Bazzi.akra-bazzi-termination-tac ctxt))
⟫ *prove termination of Akra−Bazzi functions*

hide-const *CLAMP CLAMP' MASTER-BOUND MASTER-BOUND' MASTER-BOUND''*

end
theory *Akra-Bazzi-Approximation*
imports
Complex-Main
Akra-Bazzi-Method
~~/src/HOL/Decision-Procs/Approximation
begin

context *akra-bazzi-params-nonzero*
begin

lemma *sum-alt*: $(\sum i<k.\ as!i * bs!i\ powr\ p') = (\sum i<k.\ as!i * exp\ (p' * ln\ (bs!i)))$
proof (*intro setsum.cong*)
fix i **assume** $i \in \{..<k\}$
with *b-bounds* **have** $bs!i > 0$ **by** *simp*
thus $as!i * bs!i\ powr\ p' = as!i * exp\ (p' * ln\ (bs!i))$ **by** (*simp add*: *powr-def*)
qed *simp*

lemma *akra-bazzi-p-rel-intros-aux*:

$1 < (\sum i<k.\ as!i * exp\ (p' * ln\ (bs!i))) \Longrightarrow p' < p$
$1 > (\sum i<k.\ as!i * exp\ (p' * ln\ (bs!i))) \Longrightarrow p' > p$
$1 \leq (\sum i<k.\ as!i * exp\ (p' * ln\ (bs!i))) \Longrightarrow p' \leq p$
$1 \geq (\sum i<k.\ as!i * exp\ (p' * ln\ (bs!i))) \Longrightarrow p' \geq p$
$(\sum i<k.\ as!i * exp\ (x * ln\ (bs!i))) \leq 1 \wedge (\sum i<k.\ as!i * exp\ (y * ln\ (bs!i))) \geq 1 \Longrightarrow p \in \{y..x\}$
$(\sum i<k.\ as!i * exp\ (x * ln\ (bs!i))) < 1 \wedge (\sum i<k.\ as!i * exp\ (y * ln\ (bs!i))) > 1 \Longrightarrow p \in \{y<..<x\}$
using *p-lessI p-greaterI p-leI p-geI p-boundsI p-boundsI'* **by** (*simp-all only*: *sum-alt*)
end

lemmas *akra-bazzi-p-rel-intros-exp* =
akra-bazzi-params-nonzero.akra-bazzi-p-rel-intros-aux[*rotated*, *OF* - *akra-bazzi-params-nonzeroI*]

lemma *eval-akra-bazzi-setsum*:
$(\sum i<0.\ as!i * exp\ (x * ln\ (bs!i))) = 0$
$(\sum i<Suc\ 0.\ (a\#as)!i * exp\ (x * ln\ ((b\#bs)!i))) = a * exp\ (x * ln\ b)$
$(\sum i<Suc\ k.\ (a\#as)!i * exp\ (x * ln\ ((b\#bs)!i))) = a * exp\ (x * ln\ b) + (\sum i<k.\ as!i * exp\ (x * ln\ (bs!i)))$
apply *simp*
apply *simp*
apply (*induction k arbitrary*: *a as b bs*)
apply *simp-all*
done

ML ⟪

```
signature AKRA-BAZZI-APPROXIMATION =
sig
  val akra-bazzi-approximate-tac : int -> Proof.context -> int -> tactic
end

structure Akra-Bazzi-Approximation: AKRA-BAZZI-APPROXIMATION =
struct

fun akra-bazzi-approximate-tac prec ctxt =
  let
    val simps = @{thms eval-length eval-akra-bazzi-setsum add-0-left add-0-right
                  mult-1-left mult-1-right}
  in
    SELECT-GOAL (
      resolve-tac ctxt @{thms akra-bazzi-p-rel-intros-exp} 1
      THEN ALLGOALS (fn i =>
        if i > 1 then
          SELECT-GOAL (
            Local-Defs.unfold-tac ctxt
           @{thms bex-set-simps ball-set-simps greaterThanLessThan-iff eval-length}
```

THEN TRY (*SOLVE* (*Eval-Numeral.eval-numeral-tac ctxt 1*))
) *i*
else
SELECT-GOAL (*Local-Defs.unfold-tac ctxt simps*) *i*
THEN Approximation.approximation-tac prec [] *NONE ctxt i*
)
)
end

end;
⟫

method-setup *akra-bazzi-approximate* = ⟪
Scan.lift Parse.nat >>
(*fn prec* => *fn ctxt* =>
SIMPLE-METHOD′ (*Akra-Bazzi-Approximation.akra-bazzi-approximate-tac prec ctxt*))
⟫ *approximate transcendental Akra−Bazzi parameters*

end

8 Examples

theory *Master-Theorem-Examples*
imports
Complex-Main
Akra-Bazzi-Method
Akra-Bazzi-Approximation
begin

8.1 Merge sort

function *merge-sort-cost* :: $(nat \Rightarrow real) \Rightarrow nat \Rightarrow real$ **where**
merge-sort-cost t 0 = 0
| *merge-sort-cost t 1 = 1*
| $n \geq 2 \Longrightarrow$ *merge-sort-cost t n* =
merge-sort-cost t (*nat* ⌊*real n* / *2*⌋) + *merge-sort-cost t* (*nat* ⌈*real n* / *2*⌉) + *t n*
by *force simp-all*
termination by *akra-bazzi-termination simp-all*

lemma *merge-sort-nonneg*[*simp*]: $(\bigwedge n.\ t\ n \geq 0) \Longrightarrow merge\text{-}sort\text{-}cost\ t\ x \geq 0$
by (*induction t x rule*: *merge-sort-cost.induct*) (*simp-all del*: *One-nat-def*)

lemma $t \in \Theta(\lambda n.\ real\ n) \Longrightarrow (\bigwedge n.\ t\ n \geq 0) \Longrightarrow merge\text{-}sort\text{-}cost\ t \in \Theta(\lambda n.\ real\ n * ln\ (real\ n))$
by (*master-theorem 2.3*) *simp-all*

8.2 Karatsuba multiplication

function *karatsuba-cost* :: *nat* ⇒ *real* **where**
  karatsuba-cost 0 = *0*
| *karatsuba-cost 1* = *1*
| $n \geq 2 \Longrightarrow$ *karatsuba-cost n* =
    3 ∗ *karatsuba-cost* (*nat* ⌈*real n* / *2*⌉) + *real n*
by *force simp-all*
termination by *akra-bazzi-termination simp-all*

lemma *karatsuba-cost-nonneg*[*simp*]: *karatsuba-cost n* ≥ *0*
  by (*induction n rule*: *karatsuba-cost.induct*) (*simp-all del*: *One-nat-def*)

lemma *karatsuba-cost* ∈ *O*(λ*n*. *real n powr log 2 3*)
  by (*master-theorem 1 p'*: *1*) (*simp-all add*: *powr-divide*)

lemma *karatsuba-cost-pos*: *n* ≥ *1* ⟹ *karatsuba-cost n* > *0*
  by (*induction n rule*: *karatsuba-cost.induct*) (*auto intro*!: *add-nonneg-pos simp del*: *One-nat-def*)

lemma *karatsuba-cost* ∈ Θ(λ*n*. *real n powr log 2 3*)
  using *karatsuba-cost-pos*
  by (*master-theorem 1 p'*: *1*) (*auto simp add*: *powr-divide eventually-at-top-linorder*)

8.3 Strassen matrix multiplication

function *strassen-cost* :: *nat* ⇒ *real* **where**
  strassen-cost 0 = *0*
| *strassen-cost 1* = *1*
| *n* ≥ *2* ⟹ *strassen-cost n* = *7* ∗ *strassen-cost* (*nat* ⌈*real n* / *2*⌉) + *real* (*n*ˆ*2*)
by *force simp-all*
termination by *akra-bazzi-termination simp-all*

lemma *strassen-cost-nonneg*[*simp*]: *strassen-cost n* ≥ *0*
  by (*induction n rule*: *strassen-cost.induct*) (*simp-all del*: *One-nat-def*)

lemma *strassen-cost* ∈ *O*(λ*n*. *real n powr log 2 7*)
  by (*master-theorem 1 p'*: *2*) (*auto simp*: *powr-divide eventually-at-top-linorder*)

lemma *strassen-cost-pos*: *n* ≥ *1* ⟹ *strassen-cost n* > *0*
  by (*cases n rule*: *strassen-cost.cases*) (*simp-all add*: *add-nonneg-pos del*: *One-nat-def*)

lemma *strassen-cost* ∈ Θ(λ*n*. *real n powr log 2 7*)
  using *strassen-cost-pos*
  by (*master-theorem 1 p'*: *2*) (*auto simp*: *powr-divide eventually-at-top-linorder*)

8.4 Deterministic select

function *select-cost* :: *nat* ⇒ *real* **where**
  n ≤ *20* ⟹ *select-cost n* = *0*

| $n > 20 \Longrightarrow$ *select-cost* n =
select-cost (*nat* $\lfloor$*real* n / $5\rfloor$) + *select-cost* (*nat* $\lfloor 7 *$ *real* n / $10\rfloor + 6) + 12$
$*$ *real* n / 5
by *force simp-all*
termination by *akra-bazzi-termination simp-all*

lemma *select-cost* $\in \Theta(\lambda n.$ *real* $n)$
by (*master-theorem 3*) *auto*

8.5 Decreasing function

function *dec-cost* :: *nat* $\Rightarrow$ *real* **where**
$n \leq 2 \Longrightarrow$ *dec-cost* $n = 1$
| $n > 2 \Longrightarrow$ *dec-cost* $n = 0.5*$*dec-cost* (*nat* $\lfloor$*real* n / $2\rfloor$) + 1 / *real* n
by *force simp-all*
termination by *akra-bazzi-termination simp-all*

lemma *dec-cost* $\in \Theta(\lambda x$::*nat*. *ln* x / $x)$
by (*master-theorem 2.3*) *simp-all*

8.6 Example taken from Drmota and Szpakowski

function *drmota1* :: *nat* $\Rightarrow$ *real* **where**
$n < 20 \Longrightarrow$ *drmota1* $n = 1$
| $n \geq 20 \Longrightarrow$ *drmota1* $n = 2 *$ *drmota1* (*nat* $\lfloor$*real* $n/2\rfloor$) + $8/9 *$ *drmota1* (*nat* $\lfloor 3*$*real* $n/4\rfloor$) + *real* nˆ2 / *ln* (*real* n)
by *force simp-all*
termination by *akra-bazzi-termination simp-all*

lemma *drmota1* $\in \Theta(\lambda n$::*real*. nˆ$2 *$ *ln* (*ln* n))
by (*master-theorem 2.2*) (*simp-all add*: *power-divide*)

function *drmota2* :: *nat* $\Rightarrow$ *real* **where**
$n < 20 \Longrightarrow$ *drmota2* $n = 1$
| $n \geq 20 \Longrightarrow$ *drmota2* $n = 1/3 *$ *drmota2* (*nat* $\lfloor$*real* $n/3 + 1/2\rfloor$) + $2/3 *$ *drmota2* (*nat* $\lfloor 2*$*real* $n/3 - 1/2\rfloor$) + 1
by *force simp-all*
termination by *akra-bazzi-termination simp-all*

lemma *drmota2* $\in \Theta(\lambda x.$ *ln* (*real* x))
by *master-theorem simp-all*

lemma *boncelet-phrase-length*:
fixes p δ :: *real* **assumes** p: $p > 0$ $p < 1$ **and** δ: $\delta > 0$ $\delta < 1$ $2*p + \delta < 2$
fixes d :: *nat* $\Rightarrow$ *real*
defines $q \equiv 1 - p$
assumes *d-nonneg*: $\bigwedge n.$ d $n \geq 0$

assumes *d-rec*: $\bigwedge n.\ n \geq 2 \Longrightarrow d\ n = 1 + p * d\ (nat\ \lfloor p * real\ n + \delta \rfloor) + q * d\ (nat\ \lfloor q * real\ n - \delta \rfloor)$
shows $d \in \Theta(\lambda x.\ ln\ x)$
using *assms* **by** (*master-theorem recursion*: *d-rec*, *simp-all*)

8.7 Transcendental exponents

function *foo-cost* :: *nat* ⇒ *real* **where**
$n < 200 \Longrightarrow foo\text{-}cost\ n = 0$
| $n \geq 200 \Longrightarrow foo\text{-}cost\ n =$
$foo\text{-}cost\ (nat\ \lfloor real\ n\ /\ 3 \rfloor) + foo\text{-}cost\ (nat\ \lfloor 3 * real\ n\ /\ 4 \rfloor + 42) + real\ n$
by *force simp-all*
termination by *akra-bazzi-termination simp-all*

lemma *foo-cost-nonneg* [*simp*]: $foo\text{-}cost\ n \geq 0$
by (*induction n rule*: *foo-cost.induct*) *simp-all*

lemma $foo\text{-}cost \in \Theta(\lambda n.\ real\ n\ powr\ akra\text{-}bazzi\text{-}exponent\ [1,1]\ [1/3,3/4])$
proof (*master-theorem 1 p'*: *1*)
have $\forall n \geq 200.\ foo\text{-}cost\ n > 0$ **by** (*simp add*: *add-nonneg-pos*)
thus *eventually* ($\lambda n.\ foo\text{-}cost\ n > 0$) *at-top* **unfolding** *eventually-at-top-linorder* **by** *blast*
qed *simp-all*

lemma $akra\text{-}bazzi\text{-}exponent\ [1,1]\ [1/3,3/4] \in \{1.1519623..1.1519624\}$
by (*akra-bazzi-approximate 29*)

8.8 Functions in locale contexts

locale *det-select* =
fixes *b* :: *real*
assumes *b*: $b > 0$ $b < 7/10$
begin

function *select-cost'* :: *nat* ⇒ *real* **where**
$n \leq 20 \Longrightarrow select\text{-}cost'\ n = 0$
| $n > 20 \Longrightarrow select\text{-}cost'\ n =$
$select\text{-}cost'\ (nat\ \lfloor real\ n\ /\ 5 \rfloor) + select\text{-}cost'\ (nat\ \lfloor b * real\ n \rfloor + 6) + 6 * real\ n + 5$
by *force simp-all*
termination using *b* **by** *akra-bazzi-termination simp-all*

lemma $a \geq 0 \Longrightarrow select\text{-}cost' \in \Theta(\lambda n.\ real\ n)$
using *b* **by** (*master-theorem 3*, *force+*)

end

8.9 Non-curried functions

function *baz-cost* :: *nat* × *nat* ⇒ *real* **where**

$n \leq 2 \Longrightarrow$ *baz-cost* $(a,\ n) = 0$
| $n > 2 \Longrightarrow$ *baz-cost* $(a,\ n) = 3 * $ *baz-cost* $(a,\ nat\ \lfloor real\ n\ /\ 2\rfloor) + real\ a$
by *force simp-all*
termination by *akra-bazzi-termination simp-all*

lemma *baz-cost-nonneg* [*simp*]: $a \geq 0 \Longrightarrow$ *baz-cost* $(a,\ n) \geq 0$
by (*induction a n rule*: *baz-cost.induct*[*split-format* (*complete*)]) *simp-all*

lemma
assumes $a > 0$
shows $(\lambda x.\ baz\text{-}cost\ (a,\ x)) \in \Theta(\lambda x.\ x\ powr\ log\ 2\ 3)$
proof (*master-theorem 1 p'*: *0*)
from *assms* **have** $\forall x{\geq}3.\ baz\text{-}cost\ (a,\ x) > 0$ **by** (*auto intro*: *add-nonneg-pos*)
thus *eventually* $(\lambda x.\ baz\text{-}cost\ (a, x) > 0)$ *at-top* **by** (*force simp*: *eventually-at-top-linorder*)
qed (*insert assms*, *simp-all add*: *powr-divide*)

function *bar-cost* :: $nat \times nat \Rightarrow real$ **where**
$n \leq 2 \Longrightarrow$ *bar-cost* $(a,\ n) = 0$
| $n > 2 \Longrightarrow$ *bar-cost* $(a,\ n) = 3 * $ *bar-cost* $(2 * a,\ nat\ \lfloor real\ n\ /\ 2\rfloor) + real\ a$
by *force simp-all*
termination by *akra-bazzi-termination simp-all*

8.10 Ham-sandwich trees

function *ham-sandwich-cost* :: $nat \Rightarrow real$ **where**
$n < 4 \Longrightarrow$ *ham-sandwich-cost* $n = 1$
| $n \geq 4 \Longrightarrow$ *ham-sandwich-cost* $n =$
ham-sandwich-cost $(nat\ \lfloor n/4\rfloor)$ + *ham-sandwich-cost* $(nat\ \lfloor n/2\rfloor) + 1$
by *force simp-all*
termination by *akra-bazzi-termination simp-all*

lemma *ham-sandwich-cost-pos* [*simp*]: *ham-sandwich-cost* $n > 0$
by (*induction n rule*: *ham-sandwich-cost.induct*) *simp-all*

The golden ratio

definition $\varphi = ((1 + sqrt\ 5)\ /\ 2 :: real)$

lemma *φ-pos* [*simp*]: $\varphi > 0$ **and** *φ-nonneg* [*simp*]: $\varphi \geq 0$ **and** *φ-nonzero* [*simp*]: $\varphi \neq 0$
proof−
show $\varphi > 0$ **unfolding** *φ-def* **by** (*simp add*: *add-pos-nonneg*)
thus $\varphi \geq 0$ $\varphi \neq 0$ **by** *simp-all*
qed

lemma *ham-sandwich-cost* $\in \Theta(\lambda n.\ n\ powr\ (log\ 2\ \varphi))$
proof (*master-theorem 1 p'*: *0*)
have $(1\ /\ 4)\ powr\ log\ 2\ \varphi + (1\ /\ 2)\ powr\ log\ 2\ \varphi =$

inverse $(2$ *powr log* $2\ \varphi)\hat{}2$ + *inverse* $(2$ *powr log* $2\ \varphi)$
by (*simp add*: *powr-divide field-simps powr-powr power2-eq-square powr-mult*[*symmetric*]
del: *powr-log-cancel*)
also have ... = *inverse* $(\varphi\hat{}2)$ + *inverse* φ **by** (*simp add*: *power2-eq-square*)
also have $\varphi + 1 = \varphi{*}\varphi$ **by** (*simp add*: φ*-def field-simps*)
hence *inverse* $(\varphi\hat{}2)$ + *inverse* $\varphi = 1$ **by** (*simp add*: *field-simps power2-eq-square*)
finally show $(1\ /\ 4)$ *powr log* $2\ \varphi + (1\ /\ 2)$ *powr log* $2\ \varphi = 1$ **by** *simp*
qed *simp-all*

end

References

[1] M. Akra and L. Bazzi. On the solution of linear recurrence equations. *Computational Optimization and Applications*, 10(2):195–210, 1998.

[2] T. Leighton. Notes on better Master theorems for divide-and-conquer recurrences. 1996.