

# The Akra–Bazzi theorem and the Master theorem

Manuel Eberl

October 27, 2022

## 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

<b>1</b>	<b>Auxiliary lemmas</b>	<b>2</b>
<b>2</b>	<b>Asymptotic bounds</b>	<b>8</b>
<b>3</b>	<b>The continuous Akra-Bazzi theorem</b>	<b>16</b>
<b>4</b>	<b>The discrete Akra-Bazzi theorem</b>	<b>45</b>
<b>5</b>	<b>The Master theorem</b>	<b>67</b>
<b>6</b>	<b>Evaluating expressions with rational numerals</b>	<b>77</b>
<b>7</b>	<b>The proof methods</b>	<b>81</b>
7.1	Master theorem and termination . . . . .	81
<b>8</b>	<b>Examples</b>	<b>92</b>
8.1	Merge sort . . . . .	92
8.2	Karatsuba multiplication . . . . .	92
8.3	Strassen matrix multiplication . . . . .	93
8.4	Deterministic select . . . . .	93
8.5	Decreasing function . . . . .	94

8.6	Example taken from Drmota and Szpakowski . . . . .	94
8.7	Transcendental exponents . . . . .	95
8.8	Functions in locale contexts . . . . .	95
8.9	Non-curried functions . . . . .	95
8.10	Ham-sandwich trees . . . . .	96

## 1 Auxiliary lemmas

**theory** *Akra-Bazzi-Library*

**imports**

*Complex-Main*

*Landau-Symbols.Landau-More*

*Pure-ex.Guess*

**begin**

**lemma** *ln-mono*:  $0 < x \implies 0 < y \implies x \leq y \implies \ln (x::real) \leq \ln y$   
**by** (*subst ln-le-cancel-iff*) *simp-all*

**lemma** *ln-mono-strict*:  $0 < x \implies 0 < y \implies x < y \implies \ln (x::real) < \ln y$   
**by** (*subst ln-less-cancel-iff*) *simp-all*

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

**lemma** *sum-pos'*:

**assumes** *finite I*

**assumes**  $\exists x \in I. f x > (0 :: - :: \text{linordered-ab-group-add})$

**assumes**  $\bigwedge x. x \in I \implies f x \geq 0$

**shows**  $\text{sum } f I > 0$

**proof** –

**from** *assms(2)* **guess** *x* **by** (*elim bexE*) **note**  $x = \text{this}$

**from** *x* **have**  $I = \text{insert } x I$  **by** *blast*

**also from** *assms(1)* **have**  $\text{sum } f \dots = f x + \text{sum } f (I - \{x\})$  **by** (*rule sum.insert-remove*)

**also from** *x assms* **have**  $\dots > 0$  **by** (*intro add-pos-nonneg sum-nonneg*) *simp-all*

**finally show** *?thesis* .

**qed**

**lemma** *min-mult-left*:

**assumes**  $(x::real) > 0$

**shows**  $x * \text{min } y z = \text{min } (x*y) (x*z)$

**using** *assms* **by** (*auto simp add: min-def algebra-simps*)

**lemma** *max-mult-left*:

**assumes**  $(x::real) > 0$

**shows**  $x * \text{max } y z = \text{max } (x*y) (x*z)$

**using** *assms* **by** (*auto simp add: max-def algebra-simps*)

**lemma** *DERIV-nonneg-imp-mono*:

**assumes**  $\bigwedge t. t \in \{x..y\} \implies (f \text{ has-field-derivative } f' t) (at\ t)$

**assumes**  $\bigwedge t. t \in \{x..y\} \implies f' t \geq 0$

**assumes**  $(x :: real) \leq y$

**shows**  $(f x :: real) \leq f y$

**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**  $z = \text{this}$

**from**  $z(1,2)$  *assms(2) xy* **have**  $0 \leq (y - x) * f' z$  **by** (*intro mult-nonneg-nonneg*) *simp-all*

**also note**  $z(3)$  [*symmetric*]

**finally show**  $f x \leq f y$  **by** *simp*

**qed** (*insert assms(3), simp-all*)

**lemma** *eventually-conjE*:  $\text{eventually } (\lambda x. P x \wedge Q x) F \implies (\text{eventually } P F \implies \text{eventually } Q F \implies R) \implies R$

**apply** (*frule eventually-rev-mp[of - - P], simp*)

**apply** (*drule eventually-rev-mp[of - - Q], simp*)

**apply** *assumption*

**done**

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

**lemma** *eventually-natfloor*:

**assumes**  $\text{eventually } P (at\text{-top} :: \text{nat filter})$

**shows**  $\text{eventually } (\lambda x. P (\text{nat } \lfloor x \rfloor)) (at\text{-top} :: \text{real filter})$

**proof** –

**from** *assms* **obtain**  $N$  **where**  $N: \bigwedge n. n \geq N \implies P n$  **using** *eventually-at-top-linorder* **by** *blast*

**have**  $\forall n \geq \text{real } N. P (\text{nat } \lfloor n \rfloor)$  **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*:  $f \in o(\lambda x. 1 :: real) \implies (f \longrightarrow 0) \text{ at-top}$

**by** (*drule smalloD-tendsto*) *simp*

**lemma** *smallo-1-tendsto-0*:  $(f \longrightarrow 0) \text{ at-top} \implies f \in o(\lambda x. 1 :: real)$

**by** (*rule smalloI-tendsto*) *simp-all*

**lemma** *filterlim-at-top-smallomega-1*:

**assumes**  $f \in \omega[F](\lambda x. 1 :: real) \text{ eventually } (\lambda x. f x > 0) F$

**shows** *filterlim f at-top F*

**proof** –

**from** *assms* **have** *filterlim*  $(\lambda x. \text{norm } (f x / 1)) \text{ at-top } F$

**by** (*intro smallomegaD-filterlim-at-top-norm*) (*auto elim: eventually-mono*)

**also have** *?this*  $\longleftrightarrow$  *?thesis*

using *assms* by (*intro filterlim-cong refl*) (*auto elim!*: *eventually-mono*)  
 finally show *?thesis* .  
 qed

**lemma** *smallo-imp-abs-less-real*:  
 assumes  $f \in o[F](g)$  eventually  $(\lambda x. g\ x > (0::real))\ F$   
 shows eventually  $(\lambda x. |f\ x| < g\ x)\ F$   
**proof** –  
 have  $1/2 > (0::real)$  by *simp*  
 from *landau-o.smallD[OF assms(1) this] assms(2)* show *?thesis*  
 by *eventually-elim auto*  
 qed

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

**lemma** *smallo-imp-le-real*:  
 assumes  $f \in o[F](g)$  eventually  $(\lambda x. g\ x \geq (0::real))\ F$   
 shows eventually  $(\lambda x. f\ x \leq g\ x)\ F$   
 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  $abs\ t < 1/2$   $(1 + x)^{powr\ p} =$   
 $1 + p * x + p * (p - 1) * (1 + t)^{powr\ (p - 2)} / 2 * x^2$   
**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  $?fs = (!) [?f, ?f', ?f'']$   
  
 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  $t : t \geq -0.5 \wedge t \leq 0.5$   
 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 = (∑ m<2. ?fs m 0 / (fact m) * (x - 0) ^ 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 ^ 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 powr-lower-bound: [(l::real) > 0; l ≤ x; x ≤ u] ⇒ min (l powr z) (u powr
z) ≤ x powr z
apply (cases z ≥ 0)
apply (rule order.trans[OF min.cobounded1 powr-mono2], simp-all) []
apply (rule order.trans[OF min.cobounded2 powr-mono2'], simp-all) []
done

```

```

lemma powr-upper-bound: [(l::real) > 0; l ≤ x; x ≤ u] ⇒ max (l powr z) (u
powr z) ≥ x powr z
apply (cases z ≥ 0)
apply (rule order.trans[OF powr-mono2 max.cobounded2], simp-all) []
apply (rule order.trans[OF powr-mono2' max.cobounded1], simp-all) []
done

```

**lemma one-plus-x-powr-Taylor2:**

```

  obtains k where ∧x. abs (x::real) ≤ 1/2 ⇒ abs ((1 + x) powr p - 1 - p*x)
≤ k*x^2
proof -
  define k where k = |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 ≤ 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) ≤ 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) ≤ k*x^2
      by (simp add: mult-left-mono mult-right-mono k-def)
  qed
qed

```

**lemma** *one-plus-x-powr-Taylor2-bigo*:  
**assumes**  $lim: (f \longrightarrow 0) F$   
**shows**  $(\lambda x. (1 + f x) \text{ powr } (p::real) - 1 - p * f x) \in O[F](\lambda x. f x \wedge 2)$   
**proof** –  
**from** *one-plus-x-powr-Taylor2*[of  $p$ ] **guess**  $k$  .  
**moreover from** *tendstoD*[OF  $lim$ , of  $1/2$ ]  
**have** *eventually*  $(\lambda x. abs (f x) < 1/2) F$  **by** (*simp add: dist-real-def*)  
**ultimately have** *eventually*  $(\lambda x. norm ((1 + f x) \text{ powr } p - 1 - p * f x) \leq k * norm (f x \wedge 2)) F$   
**by** (*auto elim!: eventually-mono*)  
**thus** *?thesis* **by** (*rule bigoI*)  
**qed**

**lemma** *one-plus-x-powr-Taylor1-bigo*:  
**assumes**  $lim: (f \longrightarrow 0) F$   
**shows**  $(\lambda x. (1 + f x) \text{ powr } (p::real) - 1) \in O[F](\lambda x. f x)$   
**proof** –  
**from** *assms* **have**  $(\lambda x. (1 + f x) \text{ powr } p - 1 - p * f x) \in O[F](\lambda x. (f x)^2)$   
**by** (*rule one-plus-x-powr-Taylor2-bigo*)  
**also from** *assms* **have**  $f \in O[F](\lambda \cdot. 1)$  **by** (*intro bigoI-tendsto simp-all*)  
**from** *landau-o.big.mult*[of  $f F f$ , OF - *this*] **have**  $(\lambda x. (f x) \wedge 2) \in O[F](\lambda x. f x)$   
**by** (*simp add: power2-eq-square*)  
**finally have**  $A: (\lambda x. (1 + f x) \text{ powr } p - 1 - p * f x) \in O[F](f)$  .  
**have**  $B: (\lambda x. p * f x) \in O[F](f)$  **by** *simp*  
**from** *sum-in-bigo*(1)[OF  $A B$ ] **show** *?thesis* **by** *simp*  
**qed**

**lemma** *x-times-x-minus-1-nonneg*:  $x \leq 0 \vee x \geq 1 \implies (x:::linordered-idom) * (x - 1) \geq 0$   
**proof** (*elim disjE*)  
**assume**  $x: x \leq 0$   
**also have**  $0 \leq x \wedge 2$  **by** *simp*  
**finally show**  $x * (x - 1) \geq 0$  **by** (*simp add: power2-eq-square algebra-simps*)  
**qed** *simp*

**lemma** *x-times-x-minus-1-nonpos*:  $x \geq 0 \implies x \leq 1 \implies (x:::linordered-idom) * (x - 1) \leq 0$   
**by** (*intro mult-nonneg-nonpos simp-all*)

**lemma** *powr-mono'*:  
**assumes**  $(x::real) > 0 \ x \leq 1 \ a \leq b$   
**shows**  $x \text{ powr } b \leq x \text{ powr } a$   
**proof** –  
**have** *inverse*  $x \text{ powr } a \leq \text{inverse } x \text{ powr } b$  **using** *assms*  
**by** (*intro powr-mono*) (*simp-all add: field-simps*)  
**hence** *inverse*  $(x \text{ powr } a) \leq \text{inverse } (x \text{ powr } b)$  **using** *assms* **by** *simp*  
**with** *assms* **show** *?thesis* **by** (*simp add: field-simps*)  
**qed**

**lemma** *powr-less-mono'*:

**assumes**  $(x::real) > 0 \ x < 1 \ a < b$

**shows**  $x \text{ powr } b < x \text{ powr } a$

**proof** –

**have**  $\text{inverse } x \text{ powr } a < \text{inverse } x \text{ powr } b$  **using** *assms*

**by** (*intro powr-less-mono*) (*simp-all add: field-simps*)

**hence**  $\text{inverse } (x \text{ powr } a) < \text{inverse } (x \text{ powr } b)$  **using** *assms* **by** *simp*

**with** *assms* **show** *?thesis* **by** (*simp add: field-simps*)

**qed**

**lemma** *real-powr-at-bot*:

**assumes**  $(a::real) > 1$

**shows**  $((\lambda x. a \text{ powr } x) \longrightarrow 0) \text{ at-bot}$

**proof** –

**from** *assms* **have**  $\text{filterlim } (\lambda x. \ln a * x) \text{ at-bot at-bot}$

**by** (*intro filterlim-tendsto-pos-mult-at-bot*[*OF tendsto-const - filterlim-ident*])

*auto*

**hence**  $((\lambda x. \exp (x * \ln a)) \longrightarrow 0) \text{ at-bot}$

**by** (*intro filterlim-compose*[*OF exp-at-bot*]) (*simp add: algebra-simps*)

**thus** *?thesis* **using** *assms* **unfolding** *powr-def* **by** *simp*

**qed**

**lemma** *real-powr-at-bot-neg*:

**assumes**  $(a::real) > 0 \ a < 1$

**shows**  $\text{filterlim } (\lambda x. a \text{ powr } x) \text{ at-top at-bot}$

**proof** –

**from** *assms* **have**  $\text{LIM } x \text{ at-bot. } \ln (\text{inverse } a) * -x \text{ :> at-top}$

**by** (*intro filterlim-tendsto-pos-mult-at-top*[*OF tendsto-const*] *filterlim-uminus-at-top-at-bot*)  
(*simp-all add: ln-inverse*)

**with** *assms* **have**  $\text{LIM } x \text{ at-bot. } x * \ln a \text{ :> at-top}$

**by** (*subst (asm) ln-inverse*) (*simp-all add: mult.commute*)

**hence**  $\text{LIM } x \text{ at-bot. } \exp (x * \ln a) \text{ :> at-top}$

**by** (*intro filterlim-compose*[*OF exp-at-top*]) *simp*

**thus** *?thesis* **using** *assms* **unfolding** *powr-def* **by** *simp*

**qed**

**lemma** *real-powr-at-top-neg*:

**assumes**  $(a::real) > 0 \ a < 1$

**shows**  $((\lambda x. a \text{ powr } x) \longrightarrow 0) \text{ at-top}$

**proof** –

**from** *assms* **have**  $\text{LIM } x \text{ at-top. } \ln (\text{inverse } a) * x \text{ :> at-top}$

**by** (*intro filterlim-tendsto-pos-mult-at-top*[*OF tendsto-const*])  
(*simp-all add: filterlim-ident field-simps*)

**with** *assms* **have**  $\text{LIM } x \text{ at-top. } \ln a * x \text{ :> at-bot}$

**by** (*subst filterlim-uminus-at-bot*) (*simp add: ln-inverse*)

**hence**  $((\lambda x. \exp (x * \ln a)) \longrightarrow 0) \text{ at-top}$

**by** (*intro filterlim-compose*[*OF exp-at-bot*]) (*simp-all add: mult.commute*)

**with** *assms* **show** *?thesis* **unfolding** *powr-def* **by** *simp*

qed

**lemma** *eventually-nat-real*:  
 **assumes** *eventually*  $P$  (*at-top* :: *real filter*)  
 **shows** *eventually*  $(\lambda x. P (\text{real } x))$  (*at-top* :: *nat filter*)  
 **using** *assms filterlim-real-sequentially*  
 **unfolding** *filterlim-def le-filter-def eventually-filtermap* **by** *auto*

end

## 2 Asymptotic bounds

**theory** *Akra-Bazzi-Asymptotics*

**imports**

*Complex-Main*

*Akra-Bazzi-Library*

*HOL-Library.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.  $\ln (b * x) \text{ powr } (e / 2 + 1)$ .

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

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

**private lemma** *negl-mult*:  $\text{negl } f \Longrightarrow \text{negl } g \Longrightarrow \text{negl } (\lambda x. f\ x * g\ x)$   
 **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*  $(\lambda x. \ln (b * x) \text{ powr } (-e / 2) - \ln x \text{ powr } (-e / 2) \geq g\ x) \text{ at-top}$

**proof** (*rule* *smallo-imp-le-real*)

**define** *h1* **where** [*abs-def*]:

$h1\ x = (1 + \ln b / \ln x) \text{ powr } (-e / 2) - 1 + e / 2 * (\ln b / \ln x)$  **for** *x*

**define** *h2* **where** [*abs-def*]:

$h2\ x = \ln x \text{ powr } (-e / 2) * ((1 + \ln b / \ln x) \text{ powr } (-e / 2) - 1)$  **for** *x*

**from** *bep* **have**  $(\lambda x. \ln b / \ln x \longrightarrow 0) \text{ at-top}$

**by** (*simp add: tendsto-0-smallo-1*)



**note** *one-plus-x-powr-Taylor2-bigo*[*OF this, of  $-e/2$* ]  
**also have**  $(\lambda x. (1 + \ln b / \ln x) \text{ powr } (-e/2) - 1 - -e/2 * (\ln b / \ln x))$   
 $= h1$   
**by** (*simp add: h1-def*)  
**finally have**  $h1 \in o(\lambda x. 1 / \ln x)$   
**by** (*rule landau-o.big-small-trans*) (*insert bep, simp add: power2-eq-square*)  
**with bep have**  $(\lambda x. h1 x - e/2 * (\ln b / \ln x)) \in \Theta(\lambda x. 1 / \ln x)$  **by** *simp*  
**also have**  $(\lambda x. h1 x - e/2 * (\ln b / \ln x)) = (\lambda x. (1 + \ln b / \ln x) \text{ powr } (-e/2)$   
 $- 1)$   
**by** (*rule ext*) (*simp add: h1-def*)  
**finally have**  $h2 \in \Theta(\lambda x. \ln x \text{ powr } (-e/2) * (1 / \ln x))$  **unfolding** *h2-def*  
**by** (*intro landau-theta.mult*) *simp-all*  
**also have**  $(\lambda x. \ln x \text{ powr } (-e/2) * (1 / \ln x)) \in \Theta(\lambda x. \ln x \text{ powr } (-(e/2+1)))$   
**by** *simp*  
**also from g bep have**  $(\lambda x. \ln x \text{ powr } (-(e/2+1))) \in \omega(g)$  **by** (*simp add: smal-*  
*lomega-iff-smallo*)  
**finally have**  $g \in o(h2)$  **by** (*simp add: smallomega-iff-smallo*)  
**also have** *eventually*  $(\lambda x. h2 x = \ln (b*x) \text{ powr } (-e/2) - \ln x \text{ powr } (-e/2))$   
*at-top*  
**using** *eventually-gt-at-top*[*of  $1::\text{real}$* ] *eventually-gt-at-top*[*of  $1/b$* ]  
**by** *eventually-elim* (*insert bep, simp add: field-simps powr-diff* [*symmetric*]  
*h2-def*

$$\ln\text{-mult$$
 [*symmetric*] *powr-divide del: ln-mult*)

**hence**  $h2 \in \Theta(\lambda x. \ln (b*x) \text{ powr } (-e/2) - \ln x \text{ powr } (-e/2))$  **by** (*rule bigh-*  
*etaI-cong*)

**finally show**  $g \in o(\lambda x. \ln (b * x) \text{ powr } (-e/2) - \ln x \text{ powr } (-e/2))$  .

**next**

**show** *eventually*  $(\lambda x. \ln (b*x) \text{ powr } (-e/2) - \ln x \text{ powr } (-e/2) \geq 0)$  *at-top*

**using** *eventually-gt-at-top*[*of  $1/b$* ] *eventually-gt-at-top*[*of  $1::\text{real}$* ]

**by** *eventually-elim* (*insert bep, auto intro!: powr-mono2' simp: field-simps simp*  
*del: ln-mult*)

**qed**

**private lemma** *ev1*:

*negl*  $(\lambda x. (1 + c * \text{inverse } b * \ln x \text{ powr } (-(1+e))) \text{ powr } p - 1)$

**proof**–

**from bep have**  $((\lambda x. c * \text{inverse } b * \ln x \text{ powr } (-(1+e))) \longrightarrow 0)$  *at-top*

**by** (*simp add: tendsto-0-smallo-1*)

**have**  $(\lambda x. (1 + c * \text{inverse } b * \ln x \text{ powr } (-(1+e))) \text{ powr } p - 1)$

$\in O(\lambda x. c * \text{inverse } b * \ln x \text{ powr } -(1 + e))$

**using bep by** (*intro one-plus-x-powr-Taylor1-bigo*) (*simp add: tendsto-0-smallo-1*)

**also from bep have** *negl*  $(\lambda x. c * \text{inverse } b * \ln x \text{ powr } -(1 + e))$  **by** *simp*

**finally show** *?thesis* .

**qed**

**private lemma** *ev2-aux*:

**defines**  $f \equiv \lambda x. (1 + 1/\ln (b*x) * \ln (1 + hb / b * \ln x \text{ powr } (-1-e))) \text{ powr } (-e/2)$

**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  $\lambda x. f x - 1$ ])  
**define**  $g$  **where** [abs-def]:  $g x = 1 / \ln (b * x) * \ln (1 + hb / b * \ln x \text{ powr } (-1 - e))$   
**for**  $x$   
**have**  $lim$ :  $((\lambda x. \ln (1 + hb / b * \ln x \text{ powr } (-1 - e))) \longrightarrow 0)$  *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**  $(\lambda x. (1 + g x) \text{ powr } (-e/2) - 1 - -e/2 * g x) \in O(\lambda x. (g x)^2)$  .  
**also from**  $lim'$  **have**  $(\lambda x. g x \wedge 2) \in o(\lambda x. g x * 1)$  **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**  $(\lambda x. (1 + g x) \text{ powr } (-e/2) - 1 - -e/2 * g x) = (\lambda x. f x - 1 +$   
 $e/2 * g x)$   
**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**  $f \equiv \lambda x. \ln (b * x + hb * x / \ln x \text{ powr } (1 + e)) \text{ powr } (-e/2)$   
**obtains**  $h$  **where**  
*negl*  $h$   
*eventually*  $(\lambda x. f x \geq \ln (b * x) \text{ powr } (-e/2) + h x)$  *at-top*  
*eventually*  $(\lambda x. |\ln (b * x) \text{ powr } (-e/2) + h x| < 1)$  *at-top*  
**proof** –  
**define**  $f'$   
**where**  $f' x = (1 + 1 / \ln (b * x) * \ln (1 + hb / b * \ln x \text{ powr } (-1 - e))) \text{ powr } (-e/2)$  **for**  $x$   
**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 .  
**define**  $h$  **where** [abs-def]:  $h x = \ln (b * x) \text{ powr } (-e/2) * g x$  **for**  $x$   
**show** ?thesis  
**proof** (rule that[of  $h$ ])  
**from** bep  $g$  **show** *negl*  $h$  **unfolding**  $h$ -def  
**by** (auto simp: powr-diff 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*  $(\lambda x. |\ln (b * x) \text{ powr } (-e/2) * (1 + g x)| < 1)$  *at-top*  
**by** (intro smallo-imp-abs-less-real) simp-all

**thus eventually**  $(\lambda x. |\ln (b*x) \text{ powr } (-e/2) + h x| < 1)$  *at-top*  
**by** (*simp add: algebra-simps h-def*)  
**next**  
**from** *eventually-gt-at-top[of 1/b]* **and**  $g(1)$   
**show eventually**  $(\lambda x. f x \geq \ln (b*x) \text{ powr } (-e/2) + h x)$  *at-top*  
**proof** *eventually-elim*  
**case** (*elim x*)  
**from** *bep* **have**  $b * x + hb * x / \ln x \text{ powr } (1 + e) = b*x * (1 + hb / b * \ln x \text{ powr } (-1 - e))$   
**by** (*simp add: field-simps powr-diff powr-add powr-minus*)  
**also from** *elim(1) bep*  
**have**  $\ln \dots = \ln (b*x) * (1 + 1/\ln (b*x) * \ln (1 + hb / b * \ln x \text{ powr } (-1 - e)))$   
**by** (*subst ln-mult*) (*simp-all add: add-pos-nonneg field-simps*)  
**also from** *elim(1) bep* **have**  $\dots \text{ powr } (-e/2) = \ln (b*x) \text{ powr } (-e/2) * f' x$   
**by** (*subst powr-mult*) (*simp-all add: field-simps f'-def*)  
**also from** *elim* **have**  $\dots \geq \ln (b*x) \text{ powr } (-e/2) * (1 + g x)$   
**by** (*intro mult-left-mono*) *simp-all*  
**finally show**  $f x \geq \ln (b*x) \text{ powr } (-e/2) + h x$   
**by** (*simp add: f-def h-def algebra-simps*)  
**qed**  
**qed**  
**qed**

**private lemma** *ev21*:

**obtains**  $g$  **where**

*negl g*

*eventually*  $(\lambda x. 1 + \ln (b * x + hb * x / \ln x \text{ powr } (1 + e)) \text{ powr } (-e/2) \geq 1 + \ln (b * x) \text{ powr } (-e/2) + g x)$  *at-top*

*eventually*  $(\lambda x. 1 + \ln (b * x) \text{ powr } (-e/2) + g x > 0)$  *at-top*

**proof**–

**from** *ev2* **guess**  $g$  . **note**  $g = \text{this}$

**from**  $g(3)$  **have** *eventually*  $(\lambda x. 1 + \ln (b * x) \text{ powr } (-e/2) + g x > 0)$  *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*  $(\lambda x. 1 - \ln (b * x + hb * x / \ln x \text{ powr } (1 + e)) \text{ powr } (-e/2) \leq 1 - \ln (b * x) \text{ powr } (-e/2) - g x)$  *at-top*

*eventually*  $(\lambda x. 1 - \ln (b * x) \text{ powr } (-e/2) - g x > 0)$  *at-top*

**proof**–

**from** *ev2* **guess**  $g$  . **note**  $g = \text{this}$

**from**  $g(2)$  **have** *eventually*  $(\lambda x. 1 - \ln (b * x + hb * x / \ln x \text{ powr } (1 + e)) \text{ powr } (-e/2) \leq$

$1 - \ln (b * x) \text{ powr } (-e/2) - g x)$  *at-top*

**by** *eventually-elim simp*

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

**lemma asymptotics1:**

**shows eventually**  $(\lambda x.$   
 $(1 + c * \text{inverse } b * \ln x \text{ powr } -(1+e)) \text{ powr } p *$   
 $(1 + \ln (b * x + hb * x / \ln x \text{ powr } (1 + e)) \text{ powr } (- e / 2)) \geq$   
 $1 + (\ln x \text{ powr } (-e/2)))$  *at-top*

**proof –**

**let**  $?f = \lambda x. (1 + c * \text{inverse } b * \ln x \text{ powr } -(1+e)) \text{ powr } p$   
**let**  $?g = \lambda x. 1 + \ln (b * x + hb * x / \ln x \text{ powr } (1 + e)) \text{ powr } (- e / 2)$   
**define  $f$  where [abs-def]:  $f x = 1 - ?f x$  for  $x$**   
**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  $(\lambda x. f x \leq \ln (b*x) \text{ powr } -(e/2+1))$  at-top**  
**by eventually-elim (simp add:  $f$ -def)**

**from ev21 guess  $g$  . note  $g = \text{this}$**

**define  $h$  where [abs-def]:  $h x = -g x + f x + f x * \ln (b*x) \text{ powr } (-e/2) + f x * g x$  for  $x$**

**have  $A$ : eventually  $(\lambda x. ?f x * ?g x \geq 1 + \ln (b*x) \text{ powr } (-e/2) - h x)$  at-top**  
**using  $g(2,3)$   $f$**

**proof eventually-elim**

**case (elim  $x$ )**

**let**  $?t = \ln (b*x) \text{ powr } (-e/2)$

**have**  $1 + ?t - h x = (1 - f x) * (1 + \ln (b*x) \text{ powr } (-e/2) + g x)$

**by (simp add: algebra-simps  $h$ -def)**

**also from elim have  $?f x * ?g x \geq (1 - f x) * (1 + \ln (b*x) \text{ powr } (-e/2) + g x)$**

**by (intro mult-mono[OF - elim(1)]) (simp-all add: algebra-simps  $f$ -def)**

**finally show  $?f x * ?g x \geq 1 + \ln (b*x) \text{ powr } (-e/2) - h x$  .**

**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-diff**  
*intro: landau-o.small-trans)+*

**from ev4[OF this]  $A$  show ?thesis by eventually-elim simp**

**qed**

**lemma asymptotics2:**

**shows eventually**  $(\lambda x.$   
 $(1 + c * \text{inverse } b * \ln x \text{ powr } -(1+e)) \text{ powr } p *$   
 $(1 - \ln (b * x + hb * x / \ln x \text{ powr } (1 + e)) \text{ powr } (- e / 2)) \leq$   
 $1 - (\ln x \text{ powr } (-e/2)))$  *at-top*

**proof**–

**let**  $?f = \lambda x. (1 + c * \text{inverse } b * \ln x \text{ powr } -(1+e)) \text{ powr } p$   
**let**  $?g = \lambda x. 1 - \ln (b * x + hb * x / \ln x \text{ powr } (1 + e)) \text{ powr } (- e / 2)$

**define**  $f$  **where**  $[abs-def]: f x = 1 - ?f x$  **for**  $x$   
**from**  $ev1[of c]$  **have**  $negl f$  **unfolding**  $f-def$   
**by**  $(subst \text{landau-o.small.uminus-in-iff } [symmetric]) \text{ simp}$   
**from**  $\text{landau-o.smallD}[OF \text{ this zero-less-one}]$   
**have**  $f: \text{eventually } (\lambda x. f x \leq \ln (b*x) \text{ powr } -(e/2+1)) \text{ at-top}$   
**by**  $\text{eventually-elim } (simp \text{ add: } f-def)$

**from**  $ev22$  **guess**  $g$  . **note**  $g = \text{this}$   
**define**  $h$  **where**  $[abs-def]: h x = -g x - f x + f x * \ln (b*x) \text{ powr } (-e/2) + f$   
 $x * g x$  **for**  $x$   
**have**  $((\lambda x. \ln (b * x + hb * x / \ln x \text{ powr } (1 + e)) \text{ powr } - (e / 2)) \longrightarrow 0)$   
 $\text{at-top}$   
**apply**  $(\text{insert } bep, \text{ intro } \text{tendsto-neg-powr}, \text{ simp})$   
**apply**  $(\text{rule } \text{filterlim-compose}[OF \text{ ln-at-top}])$   
**apply**  $(\text{rule } \text{filterlim-at-top-smallomega-1}, \text{ simp})$   
**using**  $\text{eventually-gt-at-top}[of \text{ max } 1 (1/b)]$   
**apply**  $(\text{auto } elim!: \text{eventually-mono } intro!: \text{add-pos-nonneg } simp: \text{field-simps})$   
**apply**  $(\text{smt } (z3) \text{ divide-nonneg-nonneg mult-neg-pos mult-nonneg-nonneg powr-non-neg})$   
**done**  
**hence**  $ev-g: \text{eventually } (\lambda x. |1 - ?g x| < 1) \text{ at-top}$   
**by**  $(\text{intro } \text{smallo-imp-abs-less-real } \text{smalloI-tendsto}) \text{ simp-all}$

**have**  $A: \text{eventually } (\lambda x. ?f x * ?g x \leq 1 - \ln (b*x) \text{ powr } (-e/2) + h x) \text{ at-top}$   
**using**  $g(2,3) \text{ ev-g } f$

**proof**  $\text{eventually-elim}$

**case**  $(elim x)$

**let**  $?t = \ln (b*x) \text{ powr } (-e/2)$

**from**  $elim$  **have**  $?f x * ?g x \leq (1 - f x) * (1 - \ln (b*x) \text{ powr } (-e/2) - g x)$

**by**  $(\text{intro } \text{mult-mono}) (simp-all \text{ add: } f-def)$

**also** **have**  $\dots = 1 - ?t + h x$  **by**  $(simp \text{ add: algebra-simps } h-def)$

**finally** **show**  $?f x * ?g x \leq 1 - \ln (b*x) \text{ powr } (-e/2) + h x$  .

**qed**

**from**  $bep \langle \text{negl } f \rangle g(1)$  **have**  $negl h$  **unfolding**  $h-def$

**by**  $(\text{fastforce } intro!: \text{sum-in-smallo } \text{landau-o.small.mult } simp: \text{powr-diff}$   
 $\text{intro: } \text{landau-o.small-trans})+$

**from**  $ev4[OF \text{ this}] A$  **show**  $?thesis$  **by**  $\text{eventually-elim } simp$

**qed**

**lemma**  $\text{asymptotics3}: \text{eventually } (\lambda x. (1 + (\ln x \text{ powr } (-e/2))) / 2 \leq 1) \text{ at-top}$   
 $(\text{is } \text{eventually } (\lambda x. ?f x \leq 1) -)$

**proof**  $(\text{rule } \text{eventually-mp}[OF \text{ always-eventually}], \text{ clarify})$

**from**  $bep$  **have**  $(?f \longrightarrow 1/2) \text{ at-top}$

**by**  $(\text{force } intro: \text{tendsto-eq-intros } \text{tendsto-neg-powr } \text{ln-at-top})$

**hence**  $\wedge e. e > 0 \implies \text{eventually } (\lambda x. |?f x - 0.5| < e) \text{ at-top}$

**by**  $(subst (asm) \text{tendsto-iff}) (simp \text{ add: } \text{dist-real-def})$

**from** *this*[of 0.5] **show** *eventually* ( $\lambda x. |?f x - 0.5| < 0.5$ ) *at-top* **by** *simp*  
**fix**  $x$  **assume**  $|?f x - 0.5| < 0.5$   
**thus**  $?f x \leq 1$  **by** *simp*  
**qed**

**lemma** *asymptotics4*: *eventually* ( $\lambda x. (1 - (\ln x \text{ powr } (-e/2))) * 2 \geq 1$ ) *at-top*  
(is *eventually* ( $\lambda x. ?f x \geq 1$ ) -)  
**proof** (*rule eventually-mp[OF always-eventually]*, *clarify*)  
**from** *bep* **have** ( $?f \longrightarrow 2$ ) *at-top*  
**by** (*force intro: tendsto-eq-intros tendsto-neg-powr ln-at-top*)  
**hence**  $\bigwedge e. e > 0 \implies \text{eventually } (\lambda x. |?f x - 2| < e)$  *at-top*  
**by** (*subst (asm) tendsto-iff*) (*simp add: dist-real-def*)  
**from** *this*[of 1] **show** *eventually* ( $\lambda x. |?f x - 2| < 1$ ) *at-top* **by** *simp*  
**fix**  $x$  **assume**  $|?f x - 2| < 1$   
**thus**  $?f x \geq 1$  **by** *simp*  
**qed**

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

**lemma** *asymptotics6*: *eventually* ( $\lambda x. hb / \ln x \text{ powr } (1 + e) < b/2$ ) *at-top*  
**and** *asymptotics7*: *eventually* ( $\lambda x. hb / \ln x \text{ powr } (1 + e) < (1 - b) / 2$ ) *at-top*  
**and** *asymptotics8*: *eventually* ( $\lambda x. x*(1 - b - hb / \ln x \text{ powr } (1 + e)) > 1$ )  
*at-top*  
**proof** -  
**from** *bep* **have**  $A: (\lambda x. hb / \ln x \text{ powr } (1 + e)) \in o(\lambda-. 1)$  **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* ( $\lambda x. hb / \ln x \text{ powr } (1+e) < b/2$ ) *at-top*  
**by** *eventually-elim* (*insert bep, simp*)  
**from** *landau-o.smallD[OF A C]* **show** *eventually* ( $\lambda x. hb / \ln x \text{ powr } (1 + e) <$

$(1 - b)/2$  at-top

by eventually-elim (insert bep, simp)

from bep have  $(\lambda x. hb / \ln x \text{ powr } (1 + e)) \in o(\lambda-. 1) (1 - b) / 2 > 0$  by simp-all

from landau-o.smallD[OF this] eventually-gt-at-top[of 1::real]

have A: eventually  $(\lambda x. 1 - b - hb / \ln x \text{ powr } (1 + e) > 0)$  at-top

by eventually-elim (insert bep, simp add: field-simps)

from bep have  $(\lambda x. x * (1 - b - hb / \ln x \text{ powr } (1+e))) \in \omega(\lambda-. 1) (0::real) < 2$  by simp-all

from landau-omega.smallD[OF this] A eventually-gt-at-top[of 0::real]

show eventually  $(\lambda x. x*(1 - b - hb / \ln x \text{ powr } (1 + e)) > 1)$  at-top

by eventually-elim (simp-all add: abs-mult)

qed

end

end

**definition** akra-bazzi-asymptotic1 b hb e p x  $\longleftrightarrow$

$(1 - hb * \text{inverse } b * \ln x \text{ powr } -(1+e)) \text{ powr } p * (1 + \ln (b*x + hb*x/\ln x \text{ powr } (1+e)) \text{ powr } (-e/2))$

$\geq 1 + (\ln x \text{ powr } (-e/2) :: \text{real})$

**definition** akra-bazzi-asymptotic1' b hb e p x  $\longleftrightarrow$

$(1 + hb * \text{inverse } b * \ln x \text{ powr } -(1+e)) \text{ powr } p * (1 + \ln (b*x + hb*x/\ln x \text{ powr } (1+e)) \text{ powr } (-e/2))$

$\geq 1 + (\ln x \text{ powr } (-e/2) :: \text{real})$

**definition** akra-bazzi-asymptotic2 b hb e p x  $\longleftrightarrow$

$(1 + hb * \text{inverse } b * \ln x \text{ powr } -(1+e)) \text{ powr } p * (1 - \ln (b*x + hb*x/\ln x \text{ powr } (1+e)) \text{ powr } (-e/2))$

$\leq 1 - \ln x \text{ powr } (-e/2 :: \text{real})$

**definition** akra-bazzi-asymptotic2' b hb e p x  $\longleftrightarrow$

$(1 - hb * \text{inverse } b * \ln x \text{ powr } -(1+e)) \text{ powr } p * (1 - \ln (b*x + hb*x/\ln x \text{ powr } (1+e)) \text{ powr } (-e/2))$

$\leq 1 - \ln x \text{ powr } (-e/2 :: \text{real})$

**definition** akra-bazzi-asymptotic3 e x  $\longleftrightarrow (1 + (\ln x \text{ powr } (-e/2))) / 2 \leq (1::\text{real})$

**definition** akra-bazzi-asymptotic4 e x  $\longleftrightarrow (1 - (\ln x \text{ powr } (-e/2))) * 2 \geq (1::\text{real})$

**definition** akra-bazzi-asymptotic5 b hb e x  $\longleftrightarrow$

$\ln (b*x - hb*x*\ln x \text{ powr } -(1+e)) \text{ powr } (-e/2::\text{real}) < 1$

**definition** akra-bazzi-asymptotic6 b hb e x  $\longleftrightarrow hb / \ln x \text{ powr } (1 + e :: \text{real}) < b/2$

**definition** akra-bazzi-asymptotic7 b hb e x  $\longleftrightarrow hb / \ln x \text{ powr } (1 + e :: \text{real}) < (1 - b) / 2$

**definition** akra-bazzi-asymptotic8 b hb e x  $\longleftrightarrow x*(1 - b - hb / \ln x \text{ powr } (1 + e :: \text{real})) > 1$

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

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

*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*  $:: (\text{real} \Rightarrow \text{real}) \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{bool}$   
**and** *integral*  $:: (\text{real} \Rightarrow \text{real}) \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{real}$   
**assumes** *integrable-const*:  $c \geq 0 \implies \text{integrable } (\lambda-. c) \text{ a } b$   
**and** *integral-const*:  $c \geq 0 \implies a \leq b \implies \text{integral } (\lambda-. c) \text{ a } b = (b - a) * c$



```

and integrable-subinterval:
  integrable f a b  $\implies a \leq a' \implies b' \leq b \implies \text{integrable } f a' b'$ 
and integral-le:
  integrable f a b  $\implies \text{integrable } g a b \implies (\bigwedge x. x \in \{a..b\} \implies f x \leq g x)$ 
 $\implies$ 
  integral f a b  $\leq$  integral g a b
and integral-combine:
   $a \leq c \implies c \leq b \implies \text{integrable } f a b \implies$ 
  integral f a c + integral f c b = integral f a b
begin
lemma integral-nonneg:
   $a \leq b \implies \text{integrable } f a b \implies (\bigwedge x. x \in \{a..b\} \implies f x \geq 0) \implies \text{integral } f a b \geq$ 
  0
  using integral-le[OF integrable-const[of 0], of f a b] by (simp add: integral-const)
end

declare sum.cong[fundef-cong]

lemma strict-mono-imp-ex1-real:
  fixes f :: real  $\Rightarrow$  real
  assumes lim-neg-inf: LIM x at-bot. f x  $\rightarrow$  at-top
  assumes lim-inf: (f  $\longrightarrow$  z) at-top
  assumes mono:  $\bigwedge a b. a < b \implies f b < f a$ 
  assumes cont:  $\bigwedge x. \text{isCont } f x$ 
  assumes y-greater-z:  $z < y$ 
  shows  $\exists! x. f x = y$ 
proof (rule ex-ex1I)
  fix a b assume  $f a = y \wedge f b = y$ 
  thus  $a = b$  by (cases rule: linorder-cases[of a b]) (auto dest: mono)
next
  from lim-neg-inf have eventually ( $\lambda x. y \leq f x$ ) at-bot by (subst (asm) filter-
lim-at-top) simp
  then obtain l where  $\bigwedge x. x \leq l \implies y \leq f x$  by (subst (asm) eventu-
ally-at-bot-linorder) auto

  from order-tendstoD(2)[OF lim-inf y-greater-z]
  obtain u where  $\bigwedge x. x \geq u \implies f x < y$  by (subst (asm) eventually-at-top-linorder)
  auto
  define a where  $a = \min l u$ 
  define b where  $b = \max l u$ 
  have  $a: f a \geq y$  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:  $a \leq b$  by (simp add: a-def b-def)
  ultimately have  $\exists x \geq a. x \leq b \wedge f x = y$  using cont by (intro IVT2) auto
  thus  $\exists x. f x = y$  by blast
qed

```

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

**definition** *akra-bazzi-exponent* :: real list  $\Rightarrow$  real list  $\Rightarrow$  real **where**  
*akra-bazzi-exponent* as bs  $\equiv$  (THE p.  $(\sum i < \text{length } as. as!i * bs!i \text{ powr } p) = 1$ )

**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  $\implies$  a  $\geq$  0  
**and** *b-bounds*: b  $\in$  set bs  $\implies$  b  $\in$  {0 < .. < 1}  
**begin**

**abbreviation** p :: real **where** p  $\equiv$  *akra-bazzi-exponent* as bs

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

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

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

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

**lemma** *b-in-bs*[*intro, simp*]: i < k  $\implies$  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  $\implies$  a  $\geq$  0  
**and** *ex-a-pos*:  $\exists a \in \text{set } as. a > 0$   
**and** *b-bounds*: b  $\in$  set bs  $\implies$  b  $\in$  {0 < .. < 1}  
**begin**

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

**lemma** *akra-bazzi-p-strict-mono*:  
**assumes** x < y  
**shows**  $(\sum i < k. as!i * bs!i \text{ powr } y) < (\sum i < k. as!i * bs!i \text{ powr } x)$   
**proof** (*intro sum-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  $\langle x < y \rangle$  **have**  $as!i * bs!i \text{ powr } y < as!i * bs!i \text{ powr } x$   
**by** (*intro mult-strict-left-mono powr-less-mono'*) *auto*  
**with**  $\langle i < k \rangle$  **show**  $\exists i \in \{..<k\}. as!i * bs!i \text{ powr } y < as!i * bs!i \text{ powr } x$  **by** *blast*  
**next**  
**fix**  $i$  **assume**  $i \in \{..<k\}$   
**with**  $a$ -ge-0  $b$ -bounds[*of*  $bs!i$ ]  $\langle x < y \rangle$  **show**  $as!i * bs!i \text{ powr } y \leq as!i * bs!i \text{ powr } x$   
**by** (*intro mult-left-mono powr-mono'*) *simp-all*  
**qed** *simp-all*

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

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

**hence** *LIM*  $p$  *at-bot*.  $as!i * bs!i \text{ powr } p \text{ :> } \text{at-top}$  **using**  $b$ -bounds  $i$   
**by** (*intro filterlim-tendsto-pos-mult-at-top[OF tendsto-const] real-powr-at-bot-neg*)  
*simp-all*

**moreover have**  $\forall p. as!i * bs!i \text{ powr } p \leq (\sum i \in \{..<k\}. as!i * bs!i \text{ powr } p)$   
**proof**  
**fix**  $p :: \text{real}$   
**from**  $a$ -ge-0  $b$ -bounds **have**  $(\sum i \in \{..<k\} - \{i\}. as!i * bs!i \text{ powr } p) \geq 0$   
**by** (*intro sum-nonneg mult-nonneg-nonneg*) *simp-all*  
**also have**  $as!i * bs!i \text{ powr } p + \dots = (\sum i \in \text{insert } i \{..<k\}. as!i * bs!i \text{ powr } p)$   
**by** (*simp add: sum.insert-remove*)  
**also from**  $i$  **have**  $\text{insert } i \{..<k\} = \{..<k\}$  **by** *blast*  
**finally show**  $as!i * bs!i \text{ powr } p \leq (\sum i \in \{..<k\}. as!i * bs!i \text{ powr } p)$  **by** *simp*  
**qed**  
**ultimately show** *LIM*  $p$  *at-bot*.  $\sum i < k. as!i * bs!i \text{ powr } p \text{ :> } \text{at-top}$   
**by** (*rule filterlim-at-top-mono[OF - always-eventually]*)

**next**  
**from**  $b$ -bounds **show**  $((\lambda x. \sum i < k. as!i * bs!i \text{ powr } x) \longrightarrow (\sum i < k. 0))$   
*at-top*  
**by** (*intro tendsto-sum tendsto-mult-right-zero real-powr-at-top-neg*) *simp-all*  
**next**  
**fix**  $x$

**from** *b-bounds* **have**  $A: \bigwedge i. i < k \implies bs ! i > 0$  **by** *simp*  
**show** *isCont*  $(\lambda x. \sum i < k. as ! i * bs ! i \text{ powr } x) x$   
**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*:  $(\sum i < k. as ! i * bs ! i \text{ powr } p) = 1$   
**and** *p-unique*:  $(\sum i < k. as ! i * bs ! i \text{ powr } p') = 1 \implies p = p'$   
**proof** –  
**from** *theI'*[*OF akra-bazzi-p-unique*] *the1-equality*[*OF akra-bazzi-p-unique*]  
**show**  $(\sum i < k. as ! i * bs ! i \text{ powr } p) = 1 \ (\sum i < k. as ! i * bs ! i \text{ powr } p') = 1 \implies p = p'$   
**unfolding** *p-def* **by** – *blast+*  
**qed**

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

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

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

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

**lemma** *p-nonneg*: *sum-list as*  $\geq 1 \implies p \geq 0$

**proof** (*rule p-geI*)  
**assume** *sum-list as*  $\geq 1$   
**also have**  $\dots = (\sum i < k. as ! i)$  **by** (*simp add: sum-list-sum-nth length-as atLeast0LessThan*)  
**also** {  
**fix** *i* **assume**  $i < k$   
**with** *b-bounds* **have**  $bs ! i > 0$  **by** *simp*  
**hence**  $as ! i * bs ! i \text{ powr } 0 = as ! i$  **by** *simp*  
**}**  
**hence**  $(\sum i < k. as ! i) = (\sum i < k. as ! i * bs ! i \text{ powr } 0)$  **by** (*intro sum.cong*) *simp-all*  
**finally show**  $1 \leq (\sum i < k. as ! i * bs ! i \text{ powr } 0)$  .

qed

end

**locale** *akra-bazzi-real-recursion* =

**fixes** *as bs* :: *real list* **and** *hs* :: (*real*  $\Rightarrow$  *real*) *list* **and** *k* :: *nat* **and** *x<sub>0</sub> x<sub>1</sub> 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  $\implies$  a  $\geq$  0*

**and** *b-bounds*: *b  $\in$  set bs  $\implies$  b  $\in$  {0 <.. $<$ 1}*

**and** *x0-ge-1*: *x<sub>0</sub>  $\geq$  1*

**and** *x0-le-x1*: *x<sub>0</sub>  $\leq$  x<sub>1</sub>*

**and** *x1-ge*: *b  $\in$  set bs  $\implies$  x<sub>1</sub>  $\geq$  2 \* x<sub>0</sub> \* inverse b*

**and** *e-pos*: *e > 0*

**and** *h-bounds*: *x  $\geq$  x<sub>1</sub>  $\implies$  h  $\in$  set hs  $\implies$  |h x|  $\leq$  hb \* x / ln x powr (1 + e)*

**and** *asymptotics*: *x  $\geq$  x<sub>0</sub>  $\implies$  b  $\in$  set bs  $\implies$  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  $\implies$  hs ! i  $\in$  set hs*

**by** (*rule nth-mem*) (*simp add: length-hs*)

**lemma** *x1-gt-1*: *x<sub>1</sub> > 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<sub>0</sub> \* inverse b*

**by** (*simp add: field-simps*)

**also from** *x1-ge* **and**  $\langle$ *b  $\in$  set bs $\rangle$  **have**  $\dots \leq x_1$  **by** *simp**

**finally show** *?thesis* .

qed

**lemma** *x1-ge-1*: *x<sub>1</sub>  $\geq$  1* **using** *x1-gt-1* **by** *simp*

**lemma** *x1-pos*: *x<sub>1</sub> > 0* **using** *x1-ge-1* **by** *simp*

**lemma** *bx-le-x*: *x  $\geq$  0  $\implies$  b  $\in$  set bs  $\implies$  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**  $x \geq x_0$   $b \in \text{set } bs$

**shows** *x0-hb-bound0*:  $hb / \ln x \text{ powr } (1 + e) < b/2$

**and** *x0-hb-bound1*:  $hb / \ln x \text{ powr } (1 + e) < (1 - b) / 2$

**and** *x0-hb-bound2*:  $x*(1 - b - hb / \ln x \text{ powr } (1 + e)) > 1$

**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**  $|(hs ! i) x| \leq hb * x / \ln x \text{ powr } (1 + e)$  **by** (*simp add: h-bounds*)

**also from** *assms x0-le-x1* **have**  $x*(1 - bs ! i - hb / \ln x \text{ powr } (1 + e)) > 1$

**by** (*simp add: x0-hb-bound2*)

**hence**  $bs ! i * x + hb * x / \ln x \text{ powr } (1 + e) + 1 < x$  **by** (*simp add: algebra-simps*)

**finally show** *?thesis* **by** *simp*

**qed**

**lemma** *step-le-x*:  $i < k \implies x \geq x_1 \implies bs ! i * x + (hs ! i) x \leq x$

**by** (*drule (1) step-diff*) *simp*

**lemma** *x0-hb-bound0'*:  $\bigwedge x b. x \geq x_0 \implies b \in \text{set } bs \implies hb / \ln x \text{ powr } (1 + e) < b$

**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**  $hb / \ln x \text{ powr } (1 + e) < bs ! i$  **by** (*simp add: x0-hb-bound0'*)

**with** *assms x0-pos x0-le-x1* **have**  $x * 0 < x * (bs ! i - hb / \ln x \text{ powr } (1 + e))$  **by** *simp*

**also have**  $\dots = bs ! i * x - hb * x / \ln x \text{ powr } (1 + e)$

**by** (*simp add: algebra-simps*)

**also from** *assms* **have**  $-hb * x / \ln x \text{ powr } (1 + e) \leq -|(hs ! i) x|$  **by** (*simp add: h-bounds*)

**hence**  $bs ! i * x - hb * x / \ln x \text{ powr } (1 + e) \leq bs ! i * x + -|(hs ! i) x|$  **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*:  $i < k \implies x \geq x_1 \implies bs ! i * x + (hs ! i) x \geq 0$

by (drule (1) step-pos) simp

**lemma** *step-nonneg'*:  $i < k \implies x \geq x_1 \implies bs ! i + (hs ! i) x / x \geq 0$   
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  $h: h \in \text{set } hs$  by (cases *hs*) auto

have  $0 \leq |h x_1|$  by simp

also from *h* have  $|h x_1| \leq hb * x_1 / \ln x_1 \text{ powr } (1+e)$  by (intro *h-bounds*)

*simp-all*

finally have  $0 \leq hb * x_1 / \ln x_1 \text{ powr } (1 + e)$ .

hence  $0 \leq \dots * (\ln x_1 \text{ powr } (1 + e) / x_1)$

by (rule *mult-nonneg-nonneg*) (intro *divide-nonneg-nonneg*, insert *x1-pos*, *simp-all*)

also have  $\dots = hb$  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  $\dots \leq hb * x / \ln x \text{ powr } (1 + e)$  by (simp add: *h-bounds*)

also have  $\dots = x * (hb / \ln x \text{ powr } (1 + e))$  by simp

also from *assms* *x0-pos* *x0-le-x1* have  $\dots < x * bs ! i$

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  $hb / \ln x \text{ powr } (1 + e) < bs ! i / 2$  by (intro *x0-hb-bound0*) *simp-all*

with *assms* *x0-pos* *x0-le-x1* have  $(-bs ! i / 2) * x < (-hb / \ln x \text{ powr } (1 + e)) * x$

by (intro *mult-strict-right-mono*) *simp-all*

also from *assms* *x0-pos* have  $\dots \leq -|(hs ! i) x|$  using *h-bounds* by simp

also have  $\dots \leq (hs ! i) x$  by simp

finally show *?thesis* using *assms* *x1-pos* by (simp add: *field-simps*)

qed

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

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**  $\dots \leq hb * x / \ln x \text{ powr } (1 + e)$  **by** (*simp add: h-bounds*)  
**also have**  $\dots = x * (hb / \ln x \text{ powr } (1 + e))$  **by** *simp*  
**also from** *assms*  $x0\text{-pos}$   $x0\text{-le-}x1$  **have**  $\dots < x * (bs!i / 2)$   
**by** (*intro mult-strict-left-mono x0-hb-bound0*) *simp-all*  
**finally show** *?thesis* **using** *assms*  $x1\text{-pos}$  **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*  $x0\text{-le-}x1$  **have**  $hb / \ln x \text{ powr } (1 + e) < (1 - bs!i) / 2$  **using** *x0-hb-bound1* **by** *simp*  
**with** *assms*  $x1\text{-pos}$  **have**  $x * ((1 - bs!i) / 2) \leq x * (1 - (bs!i + hb / \ln x \text{ powr } (1 + e)))$   
**by** (*intro mult-left-mono*) (*simp-all add: field-simps*)  
**also have**  $\dots = x - bs!i * x + -hb * x / \ln x \text{ powr } (1 + e)$  **by** (*simp add: algebra-simps*)  
**also from** *h-bounds* *assms* **have**  $-hb * x / \ln x \text{ powr } (1 + e) \leq -|(hs!i) x|$   
**by** (*simp add: length-hs*)  
**also have**  $\dots \leq -(hs!i) x$  **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*  $x0\text{-le-}x1$  **have**  $x' : x \geq x_0$  **by** *simp*  
**from**  $x1\text{-ge}$  *assms* **have**  $2 * x_0 * \text{inverse } (bs!i) \leq x_1$  **by** *simp*  
**with** *assms*  $b\text{-pos}$  **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**  $x_1 * \dots \leq x * \dots$  **by** (*intro mult-right-mono*) (*simp-all*)  
**also from** *assms*  $x1\text{-pos}$  **have**  $x * (bs!i + (hs!i) x / x) = bs!i * x + (hs!i) x$   
**by** (*simp add: field-simps*)  
**finally show** *?thesis* **using**  $x1\text{-pos}$  **by** *simp*  
**qed**

**lemma** *x0-hb-bound7'*:  $x \geq x_1 \implies i < k \implies bs!i * x + (hs!i) x > 1$   
**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**  $bs!i * x - hb * x / \ln x \text{ powr } (1+e) > x_0$



**proof** –

**from** *assms* **have**  $2 * x_0 * \text{inverse } (bs!i) \leq x_1$  **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**  $\dots \leq x * (bs!i/2)$  **by** *simp*  
**also from** *assms x0-le-x1* **have**  $hb / \ln x \text{ powr } (1+e) < bs!i/2$  **by** (*intro x0-hb-bound0*) *simp-all*  
**with** *assms* **have**  $bs!i/2 < bs!i - hb / \ln x \text{ powr } (1+e)$  **by** (*simp add: field-simps*)  
**also have**  $x * \dots = bs!i*x - hb * x / \ln x \text{ powr } (1+e)$  **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**  $bs!i*x + hb * x / \ln x \text{ powr } (1+e) > x_0$

**proof** –

**from** *assms* **have**  $x_0 < bs!i*x - hb * x / \ln x \text{ powr } (1+e)$  **by** (*rule x0-hb-bound8*)  
**also from** *assms hb-nonneg x1-pos* **have**  $hb * x / \ln x \text{ powr } (1+e) \geq 0$   
**by** (*intro mult-nonneg-nonneg divide-nonneg-nonneg*) *simp-all*  
**hence**  $bs!i*x - hb * x / \ln x \text{ powr } (1+e) \leq bs!i*x + hb * x / \ln x \text{ powr } (1+e)$   
**by** *simp*  
**finally show** *?thesis* .  
**qed**

**lemma**

**assumes**  $x \geq x_0$   
**shows** *asymptotics1*:  $i < k \implies 1 + \ln x \text{ powr } (-e/2) \leq (1 - hb * \text{inverse } (bs!i) * \ln x \text{ powr } -(1+e)) \text{ powr } p * (1 + \ln (bs!i*x + hb*x/\ln x \text{ powr } (1+e)) \text{ powr } (-e/2))$   
**and** *asymptotics2*:  $i < k \implies 1 - \ln x \text{ powr } (-e/2) \geq (1 + hb * \text{inverse } (bs!i) * \ln x \text{ powr } -(1+e)) \text{ powr } p * (1 - \ln (bs!i*x + hb*x/\ln x \text{ powr } (1+e)) \text{ powr } (-e/2))$   
**and** *asymptotics1'*:  $i < k \implies 1 + \ln x \text{ powr } (-e/2) \leq (1 + hb * \text{inverse } (bs!i) * \ln x \text{ powr } -(1+e)) \text{ powr } p * (1 + \ln (bs!i*x + hb*x/\ln x \text{ powr } (1+e)) \text{ powr } (-e/2))$   
**and** *asymptotics2'*:  $i < k \implies 1 - \ln x \text{ powr } (-e/2) \geq (1 - hb * \text{inverse } (bs!i) * \ln x \text{ powr } -(1+e)) \text{ powr } p * (1 - \ln (bs!i*x + hb*x/\ln x \text{ powr } (1+e)) \text{ powr } (-e/2))$   
**and** *asymptotics3*:  $(1 + \ln x \text{ powr } (-e/2)) / 2 \leq 1$   
**and** *asymptotics4*:  $(1 - \ln x \text{ powr } (-e/2)) * 2 \geq 1$   
**and** *asymptotics5*:  $i < k \implies \ln (bs!i*x - hb*x*\ln x \text{ powr } -(1+e)) \text{ powr } (-e/2) < 1$   
**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**  $\ln (bs!i*x + (hs!i) x) \text{ powr } -(e/2) < 1$

**proof** –

**from** *b-pos assms* **have**  $0 < bs!i/2$  **by** *simp*

**also from** *assms x0-le-x1* **have**  $\dots < bs!i + (hs!i) x / x$  **by** (*intro x0-hb-bound4*)

*simp-all*

**also from** *assms x1-pos* **have**  $x * \dots = bs!i*x + (hs!i) x$  **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'*:  $bs!i*x - hb * x / \ln x \text{ powr } (1+e) > 1$  **by** *simp*

**from** *assms* **have**  $-(hb * x / \ln x \text{ powr } (1+e)) \leq -(hs!i) x$

**by** (*intro le-imp-neg-le h-bounds*) *simp-all*

**also have**  $\dots \leq (hs!i) x$  **by** *simp*

**finally have**  $\ln (bs!i*x - hb * x / \ln x \text{ powr } (1+e)) \leq \ln (bs!i*x + (hs!i) x)$

**using** *assms b-pos x0-pos pos'* **by** (*intro ln-mono mult-pos-pos pos*) *simp-all*

**hence**  $\ln (bs!i*x + (hs!i) x) \text{ powr } -(e/2) \leq \ln (bs!i*x - hb * x / \ln x \text{ powr } (1+e)) \text{ powr } -(e/2)$

**using** *assms e-pos asymptotics5* [*of x*] *pos'* **by** (*intro powr-mono2' ln-gt-zero*) *simp-all*

**also have**  $\dots < 1$  **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*  $\Rightarrow$  *nat* **where**

*akra-bazzi-measure*  $x = \text{nat } \lceil x \rceil$

**lemma** *akra-bazzi-measure-decreases*:

**assumes**  $x \geq x_1$   $i < k$

**shows** *akra-bazzi-measure*  $(bs!i*x + (hs!i) x) < \text{akra-bazzi-measure } x$

**proof** –

**from** *step-diff assms* **have**  $(bs!i * x + (hs!i) x) + 1 < x$  **by** (*simp add: algebra-simps*)

**hence**  $\lceil (bs!i * x + (hs!i) x) + 1 \rceil \leq \lceil x \rceil$  **by** (*intro ceiling-mono*) *simp*

**hence**  $\lceil (bs!i * x + (hs!i) x) \rceil < \lceil x \rceil$  **by** *simp*

**with** *assms x1-pos* **have**  $\text{nat } \lceil (bs!i * x + (hs!i) x) \rceil < \text{nat } \lceil x \rceil$  **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:  $\bigwedge x. x \geq x_0 \implies x \leq x_1 \implies P x$ 
assumes rec:  $\bigwedge x. x > x_1 \implies (\bigwedge i. i < k \implies P (bs!i*x + (hs!i) x)) \implies P x$ 
shows  $P x$ 
proof (insert  $\langle x \geq x_0 \rangle$ , 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  $\langle x \geq x_0 \rangle$  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  $f :: real \Rightarrow real$ 
  and  $g :: real \Rightarrow real$ 
  and  $C :: real$ 
  assumes p-props:  $(\sum i < k. as!i * bs!i \text{ powr } p) = 1$ 
  and f-base:  $x \geq x_0 \implies x \leq x_1 \implies f x \geq 0$ 
  and f-rec:  $x > x_1 \implies f x = g x + (\sum i < k. as!i * f (bs!i * x + (hs!i) x))$ 
  and g-nonneg:  $x \geq x_0 \implies g x \geq 0$ 
  and C-bound:  $b \in \text{set } bs \implies x \geq x_1 \implies C*x \leq b*x - hb*x/\ln x \text{ powr } (1+e)$ 
  and g-integrable:  $x \geq x_0 \implies \text{integrable } (\lambda u. g u / u \text{ powr } (p + 1)) x_0 x$ 
begin

```

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

**lemma** akra-bazzi-integrable:

```

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

```

**definition** g-approx ::  $nat \Rightarrow real \Rightarrow real$  **where**

```

g-approx  $i x = x \text{ powr } p * \text{integral } (\lambda u. g u / u \text{ powr } (p + 1)) (bs!i * x + (hs!i)$ 
```

$x$ )  $x$

**lemma**  $f$ -nonneg:  $x \geq x_0 \implies f x \geq 0$

**proof** (induction  $x$  rule: akra-bazzi-induct)

case (base  $x$ )

with  $f$ -base[*of*  $x$ ] show ?case by simp

next

case (rec  $x$ )

with  $x_0$ -le- $x_1$  have  $g x \geq 0$  by (intro  $g$ -nonneg) simp-all

moreover {

fix  $i$  assume  $i: i < k$

with rec.IH have  $f (bs!i*x + (hs!i) x) \geq 0$  by simp

with  $i$  have  $as!i * f (bs!i*x + (hs!i) x) \geq 0$

by (intro mult-nonneg-nonneg[OF  $a$ -ge-0]) simp-all

}

hence  $(\sum_{i < k} as!i * f (bs!i*x + (hs!i) x)) \geq 0$  by (intro sum-nonneg) blast

ultimately show  $f x \geq 0$  using rec.hyps by (subst  $f$ -rec) simp-all

qed

**definition**  $f$ -approx ::  $real \Rightarrow real$  where

$f$ -approx  $x = x$  powr  $p * (1 + \text{integral } (\lambda u. g u / u \text{ powr } (p + 1)) x_0 x)$

**lemma**  $f$ -approx-aux:

assumes  $x \geq x_0$

shows  $1 + \text{integral } (\lambda u. g u / u \text{ powr } (p + 1)) x_0 x \geq 1$

**proof** –

from assms have  $\text{integral } (\lambda u. g u / u \text{ powr } (p + 1)) x_0 x \geq 0$

by (intro integral-nonneg ballI  $g$ -nonneg divide-nonneg-nonneg  $g$ -integrable)

simp-all

thus ?thesis by simp

qed

**lemma**  $f$ -approx-pos:  $x \geq x_0 \implies f$ -approx  $x > 0$

unfolding  $f$ -approx-def by (intro mult-pos-pos, insert  $x_0$ -pos, simp, drule  $f$ -approx-aux, simp)

**lemma**  $f$ -approx-nonneg:  $x \geq x_0 \implies f$ -approx  $x \geq 0$

using  $f$ -approx-pos[*of*  $x$ ] by simp

**lemma**  $f$ -approx-bounded-below:

obtains  $c$  where  $\bigwedge x. x \geq x_0 \implies x \leq x_1 \implies f$ -approx  $x \geq c$   $c > 0$

**proof** –

{

fix  $x$  assume  $x: x \geq x_0$   $x \leq x_1$

with  $x_0$ -pos have  $x$  powr  $p \geq \min (x_0 \text{ powr } p) (x_1 \text{ powr } p)$

by (intro powr-lower-bound) simp-all

with  $x$  have  $f$ -approx  $x \geq \min (x_0 \text{ powr } p) (x_1 \text{ powr } p) * 1$

**unfolding**  $f$ -approx-def **by** (intro mult-mono  $f$ -approx-aux) simp-all  
**}**  
**from** this  $x_0$ -pos  $x_1$ -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  $x_1$ -gt-1 **have**  $\ln x$ -pos:  $\ln x > 0$  **by** simp  
**from** assms  $x_1$ -pos **have**  $x$ -pos:  $x > 0$  **by** simp  
**from** assms  $x_0$ -le- $x_1$  **have** \*:  $hb / \ln x$  powr  $(1+e) < bs!i/2$  **by** (intro  $x_0$ -hb-bound0) simp-all

**with**  $hb$ -nonneg  $\ln x$ -pos **have** ( $bs!i - hb * \ln x$  powr  $-(1+e)$ )  $> 0$   
**by** (subst powr-minus) (simp-all add: field-simps)

**with** \* **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**  $\dots \leq (hs!i) x$  **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**  $\dots \leq (hb*x*\ln x$  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  $x_0$ -le- $x_1$  **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**  $b\text{-less-1}$  [of  $bs!i$ ] **assms** **have**  $b\text{-less-1}$ :  $bs!i < 1$  **by** *simp*  
**from**  $x1\text{-gt-1}$  **assms** **have**  $ln\text{-x-pos}$ :  $ln\ x > 0$  **by** *simp*  
**have**  $mono$ :  $\bigwedge a\ b. a \leq b \implies (bs!i*x)\ \text{powr}\ p * a \leq (bs!i*x)\ \text{powr}\ p * b$   
**by** (*rule mult-left-mono*) *simp-all*

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

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

```

(-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  $(bs!i*x + (hs!i) x) \text{ powr } p * (1 - \ln (bs!i*x + (hs!i) x) \text{ powr } (-e/2))$ 
   $\leq$ 
     $(bs!i*x) \text{ powr } p * (1 - \ln x \text{ powr } (-e/2))$ 
proof -
  define  $s :: \text{real}$  where  $s = (\text{if } p \geq 0 \text{ then } 1 \text{ else } -1)$ 
  from  $\text{assms } x0\text{-le-}x1$  have  $x: x \geq x_0$  by simp
  from  $\text{assms } x1\text{-gt-}1$  have  $\ln\text{-}x\text{-pos}: \ln x > 0$  by simp
  from  $b\text{-pos}[of\ bs!i]$   $\text{assms}$  have  $b\text{-pos}: bs!i > 0$   $bs!i \neq 0$  by simp-all
  from  $b\text{-pos } hb\text{-nonneg}$  have  $\text{pos}: 1 + s * hb * (\text{inverse } (bs!i) * \ln x \text{ powr } -(1+e))$ 
   $> 0$ 
  using  $x0\text{-hb-bound}0'[OF\ x, of\ bs!i]$   $b\text{-pos}$   $\text{assms } \ln\text{-}x\text{-pos}$ 
  by (subst  $\text{powr}\text{-minus}$ ) (simp add:  $\text{field}\text{-simps } s\text{-def}$ )
  have  $\text{mono}: \bigwedge a\ b. a \leq b \implies (bs!i*x) \text{ powr } p * a \leq (bs!i*x) \text{ powr } p * b$ 
  by (rule  $\text{mult}\text{-left}\text{-mono}$ ) simp-all

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

  from  $\text{assms } x$  have  $(bs!i*x + (hs!i) x) \text{ powr } p \leq (bs!i*x + s*hb*x*\ln x \text{ powr } -(1+e)) \text{ powr } p$ 
  by ( $\text{intro } \text{asymptotics}\text{-aux}(2)$ ) (simp-all add:  $s\text{-def}$ )
  moreover from  $x0\text{-hb-bound}9[OF\ \text{assms}(1,2)]$  have  $?B \geq 0$  by ( $\text{simp add: field}\text{-simps}$ )
  ultimately have  $(bs!i*x + (hs!i) x) \text{ powr } p * ?B \leq$ 
     $(bs!i*x + s*hb*x*\ln x \text{ powr } -(1+e)) \text{ powr } p * ?B$  by ( $\text{rule mult}\text{-right}\text{-mono}$ )
  also from  $\text{assms } e\text{-pos } \text{pos}$  have  $?B \leq ?B'$ 
proof -
  from  $x0\text{-hb-bound}8'[OF\ \text{assms}(1,2)]$   $x0\text{-hb-bound}8[OF\ \text{assms}(1,2)]$   $x0\text{-ge-}1$ 
  have  $*$ :  $bs!i * x + s*hb * x / \ln x \text{ powr } (1 + e) > 1$  by ( $\text{simp add: } s\text{-def}$ )
  moreover from  $*$  have  $\dots > 0$  by simp
  moreover from  $x0\text{-hb-bound}7[OF\ \text{assms}(1,2)]$   $x0\text{-ge-}1$  have  $bs!i * x + (hs!i) x > 1$  by simp
  moreover {
    have  $(hs!i) x \leq |(hs!i) x|$  by simp
    also from  $\text{assms } x0\text{-le-}x1$  have  $\dots \leq hb*x/\ln x \text{ powr } (1+e)$  by ( $\text{intro } h\text{-bounds}$ )
  }
  finally have  $bs!i*x + (hs!i) x \leq bs!i*x + hb*x/\ln x \text{ powr } (1+e)$  by simp
  ultimately show  $?B \leq ?B'$  using  $\text{assms } e\text{-pos } x \text{ step}\text{-pos}$ 

```

by (intro diff-left-mono powr-mono2' ln-mono ln-gt-zero) simp-all  
 qed  
 hence  $(bs!i*x + s*hb*x*ln\ x\ powr\ -(1+e))\ powr\ p * ?B \leq$   
 $(bs!i*x + s*hb*x*ln\ x\ powr\ -(1+e))\ powr\ p * ?B'$  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  $?A * ?B' \leq 1 - ln\ x\ powr\ (-e/2)$  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  $i < k\ x \geq x_1$   
 shows  $C*x \leq bs!i*x + (hs!i)\ x$   
 proof -  
 from assms have  $C*x \leq bs!i*x - hb*x/ln\ x\ powr\ (1+e)$  by (intro C-bound)  
 simp-all  
 also from assms have  $-(hb*x/ln\ x\ powr\ (1+e)) \leq -|(hs!i)\ x|$   
 by (subst neg-le-iff-le, intro h-bounds) simp-all  
 hence  $bs!i*x - hb*x/ln\ x\ powr\ (1+e) \leq bs!i*x + -|(hs!i)\ x|$  by simp  
 also have  $-|(hs!i)\ x| \leq (hs!i)\ x$  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:  $real\ x \geq x_0 \implies real\ x \leq x_1 \implies f\ x \geq 0$   
 and f-rec:  $real\ x > x_1 \implies$   
 $f\ x = g\ (real\ x) + (\sum\ i < k. as!i * f\ (nat\ [bs!i * x + (hs!i)$   
 $(real\ x)]))$   
 and x0-int:  $real\ (nat\ [x_0]) = x_0$   
 begin

function  $f' :: real \Rightarrow real$  where

$x \leq x_1 \implies f'\ x = f\ (nat\ [x])$   
 $| x > x_1 \implies f'\ x = g\ x + (\sum\ i < k. as!i * f'\ (bs!i * x + (hs!i)\ x))$   
 by (force, simp-all)

termination by (relation Wellfounded.measure akra-bazzi-measure)  
 (simp-all add: akra-bazzi-measure-decreases)



```

lemma f'-base:  $x \geq x_0 \implies x \leq x_1 \implies f' x \geq 0$ 
  apply (subst f'.simps(1), assumption)
  apply (rule f-base)
  apply (rule order.trans[of - real (nat [x0])], 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:  $x \geq x_0 \implies x \leq x_1 \implies f x \geq fb2$ 
  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:  $x \geq x_0 \implies x \leq x_1 \implies g x \leq gb2$ 
begin

```

```

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

```

```

lemma gb2-nonneg:  $gb2 \geq 0$  using g-bounded[of x0] x0-le-x1 x0-pos g-nonneg[of x0] 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  $\bigwedge x i. x \geq x_1 \implies i < k \implies g\text{-approx } i x \leq c4 * g x$   $c4 > 0$ 
proof -
  define c4
  where  $c4 = \text{Max } \{c2 / \text{min } 1 (\text{min } ((b/2) \text{powr } (p+1)) ((b*3/2) \text{powr } (p+1)))$ 
   $| b. b \in \text{set } bs\}$ 

```

```

{
  from bs-nonempty obtain b where  $b: b \in \text{set } bs$  by (cases bs) auto
  let  $?m = \text{min } 1 (\text{min } ((b/2) \text{powr } (p+1)) ((b*3/2) \text{powr } (p+1)))$ 
  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**  $c_4 > 0$  **unfolding**  $c_4$ -def **by** (*subst Max-gr-iff*) (*simp, simp, blast*)  
**}**

**{**  
**fix**  $x$   $i$  **assume**  $i: i < k$  **and**  $x: x \geq x_1$   
**have** *powr-negD*:  $a \text{ powr } b \leq 0 \implies a = 0$   
**for**  $a$   $b :: \text{real}$  **unfolding** *powr-def* **by** (*simp split: if-split-asm*)  
**let**  $?m = \min 1 (\min ((bs!i/2) \text{ powr } (p+1)) ((bs!i*3/2) \text{ powr } (p+1)))$   
**have**  $\min 1 ((bs!i + (hs!i) x / x) \text{ powr } (p+1)) \geq \min 1 (\min ((bs!i/2) \text{ powr } (p+1)) ((bs!i*3/2) \text{ powr } (p+1)))$   
**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**  $c_2 / \min 1 ((bs!i + (hs!i) x / x) \text{ powr } (p+1)) \leq c_2 / ?m$  **using** *c2-pos*  
**unfolding** *min-def* **by** (*intro divide-left-mono*) (*auto intro!: mult-pos-pos dest!: powr-negD*)

**also from**  $i$   $x$  **have**  $\dots \leq c_4$  **unfolding**  $c_4$ -def **by** (*intro Max.coboundedI*) *auto*  
**finally have**  $c_2 / \min 1 ((bs!i + (hs!i) x / x) \text{ powr } (p+1)) \leq c_4$  .  
**} note**  $c_4 = \text{this}$

**{**  
**fix**  $x :: \text{real}$  **and**  $i :: \text{nat}$   
**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**  $c_2: c_2 > 0 \forall u \in \{?x'..x\}. g u \leq c_2 * g x$  **by** *auto*

**from**  $x_0$ -le- $x_1$   $x$   $i$  **have**  $x'$ -le- $x$ :  $?x' \leq x$  **by** (*intro step-le-x*) *simp-all*  
**let**  $?m = \min (?x' \text{ powr } (p + 1)) (x \text{ powr } (p + 1))$   
**define**  $m'$  **where**  $m' = \min 1 (?x'' \text{ powr } (p + 1))$   
**have** [*simp*]:  $bs!i > 0$  **by** (*intro b-pos nth-mem*) (*simp add: i length-bs*)  
**from**  $x_0$ -le- $x_1$   $x$   $i$  **have** [*simp*]:  $?x' > 0$  **by** (*intro step-pos*) *simp-all*

**{**  
**fix**  $u$  **assume**  $u: u \geq ?x' u \leq x$   
**have**  $?m \leq u \text{ powr } (p + 1)$  **using**  $x$   $u$  **by** (*intro powr-lower-bound mult-pos-pos*)  
*simp-all*  
**moreover from**  $c_2$  **and**  $u$  **have**  $g u \leq c_2 * g x$  **by** *simp*  
**ultimately have**  $g u * ?m \leq c_2 * g x * u \text{ powr } (p + 1)$  **using**  $c_2$   $x$  *x1-pos*

```

x0-le-x1
  by (intro mult-mono mult-nonneg-nonneg g-nonneg) auto
}
hence integral (λu. g u / u powr (p+1)) ?x' x ≤ integral (λu. 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 ≥ 0
  by (intro mult-nonneg-nonneg g-nonneg) simp-all
with x i x0-le-x1 have (x - ?x') * (c2 * g x / ?m) ≤ 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 ... ≤ 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 ≤ 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  $\bigwedge x. x \geq x_0 \implies x \leq x_1 \implies f\text{-approx } x \leq c \ c > 0$

proof –

```

let ?m1 = max (x0 powr p) (x1 powr p)
let ?m2 = max (x0 powr (-(p+1))) (x1 powr (-(p+1)))
let ?m3 = gb2 * ?m2
let ?m4 = 1 + (x1 - x0) * ?m3
let ?int = λx. integral (λu. g u / u powr (p + 1)) x0 x
{

```

```

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

```

**lemma**  $f$ -bounded-below:

```

assumes  $c': c' > 0$ 
obtains  $c$  where  $\bigwedge x. x \geq x_0 \implies x \leq x_1 \implies 2 * (c * f$ -approx  $x) \leq f \ x \ c \leq c'$ 
 $c > 0$ 
proof -
obtain  $c$  where  $c: \bigwedge x. x_0 \leq x \implies x \leq x_1 \implies f$ -approx  $x \leq c \ c > 0$ 
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  $x_0$ -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**  $\text{inverse } c * \text{fb2} * \text{f-approx } x \leq f x$  **by** (*simp add: field-simps*)  
**moreover have**  $\min c' (\text{inverse } c * \text{fb2}) * \text{f-approx } x \leq \text{inverse } c * \text{fb2} * \text{f-approx } x$   
**using** *f-approx-nonneg x c*  
**by** (*intro mult-right-mono f-approx-nonneg*) (*simp-all add: field-simps*)  
**ultimately have**  $2 * (\min c' (\text{inverse } c * \text{fb2}) / 2 * \text{f-approx } x) \leq f x$  **by** *simp*  
**}**  
**moreover from**  $c'$  **have**  $\min c' (\text{inverse } c * \text{fb2}) / 2 \leq c'$  **by** *simp*  
**moreover have**  $\min c' (\text{inverse } c * \text{fb2}) / 2 > 0$   
**using** *c fb2-pos c'* **by** *simp*  
**ultimately show** *?thesis* **by** (*rule that*)  
**qed**

**lemma** *akra-bazzi-lower*:

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

**{**  
**fix**  $x :: \text{real}$  **assume**  $x: x \geq x_0$   
**from**  $c5$   $x$  **have**  $c5 * 1 * \text{f-approx } x \leq c5 * (1 + \ln x \text{ powr } (-e/2)) * \text{f-approx } x$   
**by** (*intro mult-right-mono mult-left-mono f-approx-nonneg*) *simp-all*  
**also from**  $x$  **have**  $c5 * (1 + \ln x \text{ powr } (-e/2)) * \text{f-approx } x \leq f x$   
**proof** (*induction x rule: akra-bazzi-induct*)  
**case** (*base x*)  
**have**  $1 + \ln x \text{ powr } (-e/2) \leq 2$  **using** *asymptotics3 base* **by** *simp*  
**hence**  $(1 + \ln x \text{ powr } (-e/2)) * (c5 * \text{f-approx } x) \leq 2 * (c5 * \text{f-approx } x)$   
**using**  $c5$  *f-approx-nonneg base x0-ge-1* **by** (*intro mult-right-mono mult-nonneg-nonneg*)  
*simp-all*  
**also from** *base* **have**  $2 * (c5 * \text{f-approx } x) \leq f x$  **by** (*intro c5*) *simp-all*  
**finally show** *?case* **by** (*simp add: algebra-simps*)  
**next**  
**case** (*rec x*)  
**let**  $?a = \lambda i. \text{as!}i$  **and**  $?b = \lambda i. \text{bs!}i$  **and**  $?h = \lambda i. \text{hs!}i$   
**let**  $?int = \text{integral } (\lambda u. g u / u \text{ powr } (p+1)) x_0 x$   
**let**  $?int1 = \lambda i. \text{integral } (\lambda u. g u / u \text{ powr } (p+1)) x_0 (?b i*x + ?h i x)$   
**let**  $?int2 = \lambda i. \text{integral } (\lambda u. g u / u \text{ powr } (p+1)) (?b i*x + ?h i x) x$   
**let**  $?l = \ln x \text{ powr } (-e/2)$  **and**  $?l' = \lambda i. \ln (?b i*x + ?h i x) \text{ powr } (-e/2)$   
  
**from** *rec* **and**  $x_0\text{-le-}x_1$   $x_0\text{-ge-}1$  **have**  $x: x \geq x_0$  **and**  $x\text{-gt-}1: x > 1$  **by** *simp-all*  
**with**  $x_0\text{-pos}$  **have**  $x\text{-pos}: x > 0$  **and**  $x\text{-nonneg}: x \geq 0$  **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**  $0 \leq g\ x * \dots$  **by** (*intro mult-nonneg-nonneg*) *simp-all*  
**hence**  $c5 * (1 + ?l) * f\text{-approx}\ x \leq c5 * (1 + ?l) * f\text{-approx}\ x + g\ x -$   
 $c5*c4*(1 + ?l) * g\ x$   
**by** (*simp add: algebra-simps*)  
**also from** *x-gt-1* **have**  $\dots = c5 * x\ \text{powr}\ p * (1 + ?l) * (1 + ?int - c4*g\ x/x$   
 $\text{powr}\ p) + g\ x$   
**by** (*simp add: field-simps f-approx-def powr-minus*)  
**also have**  $c5 * x\ \text{powr}\ p * (1 + ?l) * (1 + ?int - c4*g\ x/x\ \text{powr}\ p) =$   
 $(\sum\ i < k. (?a\ i * ?b\ i\ \text{powr}\ p) * (c5 * x\ \text{powr}\ p * (1 + ?l) * (1 + ?int$   
 $- c4*g\ x/x\ \text{powr}\ p)))$   
**by** (*subst sum-distrib-right[symmetric]*) (*simp add: p-props*)  
**also have**  $\dots \leq (\sum\ i < k. ?a\ i * f\ (?b\ i*x + ?h\ i\ x))$   
**proof** (*intro sum-mono, clarify*)  
**fix**  $i$  **assume**  $i < k$   
**let**  $?f = c5 * ?a\ i * (?b\ i * x)\ \text{powr}\ p$   
**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 ≥ 0* **by** *simp*  
  
**have**  $(?a\ i * ?b\ i\ \text{powr}\ p) * (c5 * x\ \text{powr}\ p * (1 + ?l) * (1 + ?int - c4*g$   
 $x/x\ \text{powr}\ p)) =$   
 $?f * (1 + ?l) * (1 + ?int - c4*g\ x/x\ \text{powr}\ p)$  (**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**  $g\text{-approx}\ i\ x \leq c4 * g\ x$  **by** (*intro c4*) *simp-all*  
**hence**  $c4*g\ x/x\ \text{powr}\ p \geq ?int2\ i$  **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**  $?A * (1 + ?int1\ i) = (c5*?a\ i*(1 + ?int1\ i)) * ((?b\ i*x)\ \text{powr}\ p$   
 $* (1 + ?l))$   
**by** (*simp add: algebra-simps*)  
**also have**  $\dots \leq (c5*?a\ i*(1 + ?int1\ i)) * ((?b\ i*x + ?h\ i\ x)\ \text{powr}\ p * (1 +$   
 $?l'\ i))$   
**using** *rec.hyps i c5 a-ge-0 int-nonneg*  
**by** (*intro mult-left-mono asymptotics1' mult-nonneg-nonneg*) *simp-all*  
**also have**  $\dots = ?a\ i*(c5*(1 + ?l'\ i)*f\text{-approx}\ (?b\ i*x + ?h\ i\ x))$   
**by** (*simp add: algebra-simps f-approx-def*)

**also from  $i$  have**  $\dots \leq ?a \ i * f \ ( ?b \ i*x + ?h \ i \ x)$   
**by** *(intro mult-left-mono a-ge-0 rec.IH) simp-all*  
**finally show**  $?expr \leq ?a \ i * f \ ( ?b \ i*x + ?h \ i \ x) .$   
**qed**  
**also have**  $\dots + g \ x = f \ x$  **using** *f-rec[of x] rec.hyps x0-le-x1* **by** *simp*  
**finally show**  $?case$  **by** *simp*  
**qed**  
**finally have**  $c5 * f\text{-approx } x \leq f \ x$  **by** *simp*  
**}**  
**from this and  $c5(3)$  show**  $?thesis$  **by** *(rule that)*  
**qed**

**lemma** *akra-bazzi-bigomega:*

$f \in \Omega(\lambda x. x \text{ powr } p * (1 + \text{integral } (\lambda u. g \ u / u \text{ powr } (p + 1)) \ x_0 \ x))$   
**apply** *(fold f-approx-def, rule akra-bazzi-lower, erule landau-omega.bigI)*  
**apply** *(subst eventually-at-top-linorder, rule exI[of - x<sub>0</sub>])*  
**apply** *(simp add: f-nonneg f-approx-nonneg)*  
**done**

**end**

**locale** *akra-bazzi-real-upper = akra-bazzi-real +*

**fixes**  $fb1 \ c1 :: \text{real}$

**assumes**  $f\text{-base1}: \ x \geq x_0 \implies x \leq x_1 \implies f \ x \leq fb1$

**and**  $g\text{-growth1}: \ \forall x \geq x_1. \ \forall u \in \{C*x..x\}. \ c1 * g \ x \leq g \ u$

**and**  $c1\text{-pos}: \ c1 > 0$

**begin**

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

**lemma**  $g\text{-growth1}'$ :

**assumes**  $x \geq x_1 \ i < k \ u \in \{bs!i*x+(hs!i) \ x..x\}$

**shows**  $c1 * g \ x \leq g \ u$

**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-growth1* **show**  $?thesis$  **by** *simp*

**qed**

**lemma**  $g\text{-bounds1}$ :

**obtains**  $c3$  **where**

$\bigwedge x \ i. \ x \geq x_1 \implies i < k \implies c3 * g \ x \leq g\text{-approx } i \ x \ c3 > 0$

**proof**–

**define**  $c3$  **where**  $c3 =$

$\text{Min } \{c1*((1-b)/2) / \max 1 \ (\max ((b/2) \text{ powr } (p+1)) \ ((b*3/2) \text{ powr } (p+1)))\}$   
 $| b. \ b \in \text{set } bs\}$

**{**

```

fix  $b$  assume  $b: b \in \text{set } bs$ 
let  $?x = \max 1 (\max ((b/2) \text{ powr } (p+1)) ((b*3/2) \text{ powr } (p+1)))$ 
have  $?x \geq 1$  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  $x: x \geq x_1$ 
  with b-less-1 have b-less-1':  $bs ! i < 1$  by simp
  let  $?m = \max 1 (\max ((bs!i/2) \text{ powr } (p+1)) ((bs!i*3/2) \text{ powr } (p+1)))$ 
  from  $i$   $x$  have  $c3 \leq c1*((1-bs!i)/2) / ?m$  unfolding c3-def by (intro
Min.coboundedI) auto
  also have  $\max 1 ((bs!i + (hs ! i) x / x) \text{ powr } (p+1)) \leq \max 1 (\max ((bs!i/2) \text{ powr } (p+1)) ((bs!i*3/2) \text{ powr } (p+1)))$ 
  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  $c1*((1-bs!i)/2) / ?m \leq$ 
 $c1*((1-bs!i)/2) / \max 1 ((bs!i + (hs ! i) x / x) \text{ powr } (p+1))$ 
  by (intro divide-left-mono mult-nonneg-nonneg) (simp-all add: algebra-simps)
  finally have  $c3 \leq c1*((1-bs!i)/2) / \max 1 ((bs!i + (hs ! i) x / x) \text{ powr } (p+1))$  .
} note  $c3 = \text{this}$ 

{
  fix  $x :: \text{real}$  and  $i :: \text{nat}$ 
  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 x0-le-x1  $i$  c1-pos g-growth1'
  have  $c1: c1 > 0 \forall u \in \{?x'..x\}. g u \geq c1 * g x$  by auto
  define  $b'$  where  $b' = (1 - bs!i)/2$ 

  from  $x$  x0-le-x1  $i$  have x'-le-x:  $?x' \leq x$  by (intro step-le-x) simp-all
  let  $?m = \max (?x' \text{ powr } (p + 1)) (x \text{ powr } (p + 1))$ 
  define  $m'$  where  $m' = \max 1 (?x'' \text{ powr } (p + 1))$ 
  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:  $c1 * g x / ?m \geq 0$ 
  by (intro mult-nonneg-nonneg divide-nonneg-pos g-nonneg) simp-all

```



```

from  $x$   $i$   $g$ -nonneg  $x0$ -le- $x1$  have  $c\beta * (g\ x / x\ \text{powr}\ p) \leq (c1 * b' / m') * (g\ x /$ 
 $x\ \text{powr}\ p)$ 
unfolding  $m'$ -def  $b'$ -def by (intro mult-right-mono  $c\beta$ ) (simp-all add: field-simps)
also from  $x$ -pos have  $\dots = (x * b') * (c1 * g\ x / (x\ \text{powr}\ (p + 1) * m'))$ 
by (simp add: field-simps powr-add)
also from  $x$   $i$   $c1$ -pos  $x1$ -pos  $x0$ -le- $x1$ 
have  $\dots \leq (x - ?x') * (c1 * g\ x / (x\ \text{powr}\ (p + 1) * m'))$ 
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  $x\ \text{powr}\ (p + 1) * m' =$ 
 $\max (x\ \text{powr}\ (p + 1) * (bs ! i + (hs ! i) x / x)\ \text{powr}\ (p + 1)) (x$ 
 $\text{powr}\ (p + 1) * 1)$ 
unfolding  $m'$ -def using  $x$ -pos by (subst max.commute, intro max-mult-left)
simp
also have  $(x\ \text{powr}\ (p + 1) * (bs ! i + (hs ! i) x / x)\ \text{powr}\ (p + 1)) =$ 
 $(bs ! i + (hs ! i) x / x)\ \text{powr}\ (p + 1) * x\ \text{powr}\ (p + 1)$  by simp
also have  $\dots = (bs ! i * x + (hs ! i) x)\ \text{powr}\ (p + 1)$ 
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  $x\ \text{powr}\ (p + 1) * 1 = x\ \text{powr}\ (p + 1)$  by simp
also have  $(x - ?x') * (c1 * g\ x / ?m) = \text{integral} (\lambda \cdot. c1 * g\ x / ?m) ?x' x$ 
using  $x'$ -le- $x$  by (subst integral-const[OF  $c1$ - $g$ - $m$ -nonneg]) auto
also {
fix  $u$  assume  $u: u \geq ?x' u \leq x$ 
have  $u\ \text{powr}\ (p + 1) \leq ?m$  using  $x$   $u$   $x'$ -pos by (intro powr-upper-bound
mult-pos-pos) simp-all
moreover from  $x'$ -pos  $u$  have  $u \geq 0$  by simp
moreover from  $c1$  and  $u$  have  $c1 * g\ x \leq g\ u$  by simp
ultimately have  $c1 * g\ x * u\ \text{powr}\ (p + 1) \leq g\ u * ?m$  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  $c1 * g\ x / ?m \leq g\ u / u\ \text{powr}\ (p + 1)$  by (simp add: field-simps)
}
hence  $\text{integral} (\lambda \cdot. c1 * g\ x / ?m) ?x' x \leq \text{integral} (\lambda u. g\ u / u\ \text{powr}\ (p + 1))$ 
 $?x' x$ 
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  $c\beta * g\ x \leq g$ -approx  $i$   $x$  using  $x$ -pos
unfolding  $g$ -approx-def by (simp add: field-simps)
}
thus  $?thesis$  using that  $\langle c\beta > 0 \rangle$  by blast
qed

```

lemma  $f$ -bounded-above:

**assumes**  $c'$ :  $c' > 0$   
**obtains**  $c$  **where**  $\bigwedge x. x \geq x_0 \implies x \leq x_1 \implies f x \leq (1/2) * (c * f\text{-approx } x)$   $c \geq c'$   $c > 0$   
**proof** –  
**obtain**  $c$  **where**  $c$ :  $\bigwedge x. x_0 \leq x \implies x \leq x_1 \implies f\text{-approx } x \geq c$   $c > 0$   
**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**  $\dots \leq (fb1 + 1) * f\text{-approx } x$  **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**  $\dots \leq \max((fb1+1) / c) c' * f\text{-approx } x$   
**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**  $\bigwedge x. x \geq x_0 \implies f x \leq c6 * f\text{-approx } x$   $c6 > 0$   
**proof** –  
**obtain**  $c3$  **where**  $c3$ :  $\bigwedge x i. x \geq x_1 \implies i < k \implies c3 * g x \leq g\text{-approx } i x$   $c3 > 0$   
**by** (*rule g-bounds1*) *blast*  
**hence**  $2 / c3 > 0$  **by** *simp*  
**then obtain**  $c6$  **where**  $c6$ :  $\bigwedge x. x \geq x_0 \implies x \leq x_1 \implies f x \leq 1/2 * (c6 * f\text{-approx } x)$   

$$c6 \geq 2 / c3$$
  $c6 > 0$   
**by** (*rule f-bounded-above*) *blast*

{  
**fix**  $x :: \text{real}$  **assume**  $x$ :  $x \geq x_0$   
**hence**  $f x \leq c6 * (1 - \ln x \text{ powr } (-e/2)) * f\text{-approx } x$   
**proof** (*induction*  $x$  *rule*: *akra-bazzi-induct*)  
**case** (*base*  $x$ )  
**from** *base* **have**  $f x \leq 1/2 * (c6 * f\text{-approx } x)$  **by** (*intro*  $c6$ ) *simp-all*  
**also have**  $1 - \ln x \text{ powr } (-e/2) \geq 1/2$  **using** *asymptotics4* *base* **by** *simp*  
**hence**  $(1 - \ln x \text{ powr } (-e/2)) * (c6 * f\text{-approx } x) \geq 1/2 * (c6 * f\text{-approx } x)$

**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**  $?a = \lambda i. as!i$  **and**  $?b = \lambda i. bs!i$  **and**  $?h = \lambda i. hs!i$   
**let**  $?int = \text{integral } (\lambda u. g u / u \text{ powr } (p+1)) x_0 x$   
**let**  $?int1 = \lambda i. \text{integral } (\lambda u. g u / u \text{ powr } (p+1)) x_0 (?b i*x + ?h i x)$   
**let**  $?int2 = \lambda i. \text{integral } (\lambda u. g u / u \text{ powr } (p+1)) (?b i*x + ?h i x) x$   
**let**  $?l = \ln x \text{ powr } (-e/2)$  **and**  $?l' = \lambda i. \ln (?b i*x + ?h i x) \text{ powr } (-e/2)$   
  
**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**  $?sum \leq (\sum i < k. (?a i * ?b i \text{ powr } p) * (c6 * x \text{ powr } p * (1 - ?l) * (1 + ?int - c3 * g x / x \text{ powr } p)))$  (**is**  $- \leq ?sum'$ )  
**proof** (*rule sum-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**  $l-le-1: \ln x \text{ powr } -(e/2) \leq 1$  **using** *asymptotics3[OF x]* **by** (*simp add: field-simps*)  
  
**from**  $i$  **have**  $f (?b i*x + ?h i x) \leq c6 * (1 - ?l' i) * f\text{-approx } (?b i*x + ?h i x)$   
**by** (*rule rec.IH*)  
**hence**  $?a i * f (?b i*x + ?h i x) \leq ?a i * \dots$  **using** *a-ge-0 i*  
**by** (*intro mult-left-mono*) *simp-all*  
**also have**  $\dots = (c6 * ?a i * (1 + ?int1 i)) * ((?b i*x + ?h i x) \text{ powr } p * (1 - ?l' i))$   
**unfolding** *f-approx-def* **by** (*simp add: algebra-simps*)  
**also from** *rec.hyps c6 a-ge-0*  
**have**  $\dots \leq (c6 * ?a i * (1 + ?int1 i)) * ((?b i*x) \text{ powr } p * (1 - ?l))$   
**by** (*intro mult-left-mono asymptotics2' mult-nonneg-nonneg int-nonneg*)  
*simp-all*  
**also have**  $\dots = (1 + ?int1 i) * (c6 * ?a i * (?b i*x) \text{ powr } p * (1 - ?l))$   
**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**  $c3 * g x \leq g\text{-approx } i x$  **by** (*intro c3*) *simp-all*

**hence**  $?int2\ i \geq c3 * g\ x/x\ powr\ p$  **unfolding**  $g\text{-approx-def}$  **using**  $x\text{-pos}$   
**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**  $i\ c6\ a\text{-ge-0}\ l\text{-le-1}$   
**by** (*intro mult-right-mono mult-nonneg-nonneg*) (*simp-all add: field-simps*)  
**also have**  $\dots = (?a\ i * ?b\ i\ powr\ p) * (c6 * x\ powr\ p * (1 - ?l) * (1 + ?int -$   
 $c3 * g\ x/x\ powr\ p))$   
**using**  $b\text{-pos}[of\ bs!i]\ x\ x0\text{-pos}\ i$  **by** (*subst powr-mult*) (*simp-all add: alge-*  
*bra-simps*)  
**finally show**  $?a\ i * f\ (?b\ i * x + ?h\ i\ x) \leq \dots$  .  
**qed**

**hence**  $?sum + g\ x \leq ?sum' + g\ x$  **by** *simp*  
**also have**  $\dots = c6 * x\ powr\ p * (1 - ?l) * (1 + ?int - c3 * g\ x/x\ powr\ p) + g$   
 $x$   
**by** (*simp add: sum-distrib-right[symmetric] p-props*)  
**also have**  $\dots = c6 * (1 - ?l) * f\text{-approx}\ x - (c6 * c3 * (1 - ?l) - 1) * g\ x$   
**unfolding**  $f\text{-approx-def}$  **using**  $x\text{-pos}$  **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**  $x\ x\text{-pos}$  **have**  $(c6 * c3 * (1 - ?l) - 1) * g\ x \geq 0$   
**by** (*intro mult-nonneg-nonneg g-nonneg*) *simp-all*  
**hence**  $c6 * (1 - ?l) * f\text{-approx}\ x - (c6 * c3 * (1 - ?l) - 1) * g\ x \leq$   
 $c6 * (1 - ?l) * f\text{-approx}\ x$  **by** (*simp add: algebra-simps*)  
**}**  
**finally show**  $?case$  .  
**qed**

**also from**  $x\ c6$  **have**  $\dots \leq c6 * 1 * f\text{-approx}\ x$   
**by** (*intro mult-left-mono mult-right-mono f-approx-nonneg*) *simp-all*  
**finally have**  $f\ x \leq c6 * f\text{-approx}\ x$  **by** *simp*  
**}**  
**from this and**  $c6(\beta)$  **show**  $?thesis$  **by** (*rule that*)  
**qed**

**lemma** *akra-bazzi-bigo*:  
 $f \in O(\lambda x. x\ powr\ p * (1 + \text{integral} (\lambda u. g\ u / u\ powr\ (p + 1))\ x_0\ x))$   
**apply** (*fold f-approx-def, rule akra-bazzi-upper, erule landau-o.bigI*)  
**apply** (*subst eventually-at-top-linorder, rule exI[of - x0]*)  
**apply** (*simp add: f-nonneg f-approx-nonneg*)  
**done**

**end**

**end**

## 4 The discrete Akra-Bazzi theorem

**theory** *Akra-Bazzi*

**imports**

*Complex-Main*

*HOL-Library.Landau-Symbols*

*Akra-Bazzi-Real*

**begin**

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

**lemma** *x-over-ln-mono*:

**assumes**  $(e::real) > 0$

**assumes**  $x > \exp e$

**assumes**  $x \leq y$

**shows**  $x / \ln x \text{ powr } e \leq y / \ln y \text{ powr } e$

**proof** (*rule DERIV-nonneg-imp-mono*[of - -  $\lambda x. x / \ln x \text{ powr } e$ ])

**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**  $\langle x > \exp e \rangle$  **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**  $((\lambda x. x / \ln x \text{ powr } e)$  *has-real-derivative*

$(\ln t - e) / \ln t \text{ powr } (e+1))$  (at  $t$ ) **using** *assms*  $t' t''$

**by** (*force intro!*: *derivative-eq-intros simp: powr-diff field-simps powr-add*)

**from**  $t''$  **show**  $(\ln t - e) / \ln t \text{ powr } (e + 1) \geq 0$  **by** (*intro divide-nonneg-nonneg*) *simp-all*

**qed** (*simp-all add: assms*)

**definition** *akra-bazzi-term* ::  $\text{nat} \Rightarrow \text{nat} \Rightarrow \text{real} \Rightarrow (\text{nat} \Rightarrow \text{nat}) \Rightarrow \text{bool}$  **where**

*akra-bazzi-term*  $x_0 x_1 b t =$

$(\exists e h. e > 0 \wedge (\lambda x. h x) \in O(\lambda x. \text{real } x / \ln (\text{real } x) \text{ powr } (1+e)) \wedge$   
 $(\forall x \geq x_1. t x \geq x_0 \wedge t x < x \wedge b * x + h x = \text{real } (t x)))$

**lemma** *akra-bazzi-termI* [*intro?*]:

**assumes**  $e > 0$   $(\lambda x. h x) \in O(\lambda x. \text{real } x / \ln (\text{real } x) \text{ powr } (1+e))$

$\bigwedge x. x \geq x_1 \implies t x \geq x_0$   $\bigwedge x. x \geq x_1 \implies t x < x$

$\bigwedge x. x \geq x_1 \implies b * x + h x = \text{real } (t x)$

**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**  $x_0$   $x_1$   $k :: \text{nat}$  **and**  $as$   $bs :: \text{real list}$  **and**  $ts :: (\text{nat} \Rightarrow \text{nat}) \text{ list}$  **and**  $f :: \text{nat} \Rightarrow \text{real}$   
**assumes** *k-not-0*:  $k \neq 0$   
**and** *length-as*:  $\text{length } as = k$   
**and** *length-bs*:  $\text{length } bs = k$   
**and** *length-ts*:  $\text{length } ts = k$   
**and** *a-ge-0*:  $a \in \text{set } as \Rightarrow a \geq 0$   
**and** *b-bounds*:  $b \in \text{set } bs \Rightarrow b \in \{0 < .. < 1\}$   
**and** *ts*:  $i < \text{length } bs \Rightarrow \text{akra-bazzi-term } x_0$   $x_1$   $(bs!i)$   $(ts!i)$   
**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* ::  $\text{real} \times (\text{nat} \Rightarrow \text{real}) \text{ list}$  **where**  
 $e\text{-hs} = (\text{SOME } (e,hs)).$   
 $e > 0 \wedge \text{length } hs = k \wedge (\forall h \in \text{set } hs. (\lambda x. h\ x) \in O(\lambda x. \text{real } x / \ln(\text{real } x)$   
 $\text{powr } (1+e))) \wedge$   
 $(\forall t \in \text{set } ts. \forall x \geq x_1. t\ x \geq x_0 \wedge t\ x < x) \wedge$   
 $(\forall i < k. \forall x \geq x_1. (bs!i)*x + (hs!i)\ x = \text{real } ((ts!i)\ x))$   
 $)$

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

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

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

**proof** –  
**have**  $\exists x (\lambda(e,hs). e > 0 \wedge \text{length } hs = k \wedge$   
 $(\forall h \in \text{set } hs. (\lambda x. h\ x) \in O(\lambda x. \text{real } x / \ln(\text{real } x)\ \text{powr } (1+e))) \wedge$

$(\forall t \in \text{set } ts. \forall x \geq x_1. t x \geq x_0 \wedge t x < x) \wedge$   
 $(\forall i < k. \forall x \geq x_1. (bs!i)*x + (hs!i) x = \text{real } ((ts!i) x))$

**proof** –

**from**  $ts$  **have**  $A: \forall i \in \{..<k\}. \text{akra-bazzi-term } x_0 x_1 (bs!i) (ts!i)$  **by**  $(\text{auto simp: length-bs})$

**hence**  $\exists e h. (\forall i < k. e i > 0 \wedge$   
 $(\lambda x. h i x) \in O(\lambda x. \text{real } x / \ln (\text{real } x) \text{ powr } (1+e i)) \wedge$   
 $(\forall x \geq x_1. (ts!i) x \geq x_0 \wedge (ts!i) x < x) \wedge$   
 $(\forall i < k. \forall x \geq x_1. (bs!i)*\text{real } x + h i x = \text{real } ((ts!i) x)))$

**unfolding**  $\text{akra-bazzi-term-def}$

**by**  $(\text{subst } (asm) \text{ bchoice-iff}, \text{subst } (asm) \text{ bchoice-iff})$  **blast**

**then guess**  $ee :: - \Rightarrow \text{real}$  **and**  $hh :: - \Rightarrow \text{nat} \Rightarrow \text{real}$

**by**  $(\text{elim } exE \text{ conjE})$  **note**  $eh = \text{this}$

**define**  $e$  **where**  $e = \text{Min } \{ee i \mid i. i < k\}$

**define**  $hs$  **where**  $hs = \text{map } hh (\text{upt } 0 k)$

**have**  $e\text{-pos}: e > 0$  **unfolding**  $e\text{-def}$  **using**  $eh \text{ k-not-0}$  **by**  $(\text{subst } \text{Min-gr-iff})$

$\text{auto}$

**moreover have**  $\text{length } hs = k$  **unfolding**  $hs\text{-def}$  **by**  $(\text{simp-all add: length-ts})$

**moreover have**  $hs\text{-growth}: \forall h \in \text{set } hs. (\lambda x. h x) \in O(\lambda x. \text{real } x / \ln (\text{real } x) \text{ powr } (1+e))$

**proof**

**fix**  $h$  **assume**  $h \in \text{set } hs$

**then obtain**  $i$  **where**  $t: i < k \ h = hh \ i$  **unfolding**  $hs\text{-def}$  **by**  $\text{force}$

**hence**  $(\lambda x. h x) \in O(\lambda x. \text{real } x / \ln (\text{real } x) \text{ powr } (1+ee \ i))$  **using**  $eh$  **by**

$\text{blast}$

**also from**  $t \text{ k-not-0}$  **have**  $e \leq ee \ i$  **unfolding**  $e\text{-def}$  **by**  $(\text{subst } \text{Min-le-iff})$

$\text{auto}$

**hence**  $(\lambda x::\text{nat}. \text{real } x / \ln (\text{real } x) \text{ powr } (1+ee \ i)) \in O(\lambda x. \text{real } x / \ln (\text{real } x) \text{ powr } (1+e))$

**by**  $(\text{intro } \text{bigo-real-nat-transfer})$   $\text{auto}$

**finally show**  $(\lambda x. h x) \in O(\lambda x. \text{real } x / \ln (\text{real } x) \text{ powr } (1+e))$  .

**qed**

**moreover have**  $\forall t \in \text{set } ts. (\forall x \geq x_1. t x \geq x_0 \wedge t x < x)$

**proof**  $(\text{rule } \text{ballI})$

**fix**  $t$  **assume**  $t \in \text{set } ts$

**then obtain**  $i$  **where**  $i < k \ t = ts!i$  **using**  $\text{length-ts}$  **by**  $(\text{subst } (asm) \text{ in-set-conv-nth})$   $\text{auto}$

**with**  $eh$  **show**  $\forall x \geq x_1. t x \geq x_0 \wedge t x < x$  **unfolding**  $hs\text{-def}$  **by**  $\text{force}$

**qed**

**moreover have**  $\forall i < k. \forall x \geq x_1. (bs!i)*x + (hs!i) x = \text{real } ((ts!i) x)$

**proof**  $(\text{rule } \text{allI}, \text{rule } \text{impI})$

**fix**  $i$  **assume**  $i: i < k$

**with**  $eh$  **show**  $\forall x \geq x_1. (bs!i)*x + (hs!i) x = \text{real } ((ts!i) x)$

**using**  $\text{length-ts}$  **unfolding**  $hs\text{-def}$  **by**  $\text{fastforce}$

**qed**

**ultimately show**  $?thesis$  **by**  $\text{blast}$

**qed**

**from**  $\text{someI-ex}[OF \ \text{this}, \text{folded } e\text{-hs-def}]$  **show**  $?thesis$

**unfolding**  $e\text{-def}$   $hs\text{-def}$  **by**  $(\text{intro } \text{conjI})$   $\text{fastforce}+$

qed

lemma

*e-pos*:  $e > 0$  **and** *length-hs*:  $\text{length } hs = k$  **and**  
*hs-growth*:  $\bigwedge h. h \in \text{set } hs \implies (\lambda x. h x) \in O(\lambda x. \text{real } x / \ln (\text{real } x) \text{ powr } (1 + e))$  **and**  
*step-ge-x0*:  $\bigwedge t x. t \in \text{set } ts \implies x \geq x_1 \implies x_0 \leq t x$  **and**  
*step-less*:  $\bigwedge t x. t \in \text{set } ts \implies x \geq x_1 \implies t x < x$  **and**  
*decomp*:  $\bigwedge i x. i < k \implies x \geq x_1 \implies (bs!i)*x + (hs!i) x = \text{real } ((ts!i) x)$   
**by** (*insert e-hs-aux*) *simp-all*

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

lemma *t-in-ts* [*intro, simp*]:  $i < k \implies ts ! i \in \text{set } ts$   
**by** (*rule nth-mem*) (*simp add: length-ts*)

lemma *x0-less-x1*:  $x_0 < x_1$  **and** *x0-le-x1*:  $x_0 \leq x_1$   
**proof** -

**from** *ts-nonempty* **have**  $x_0 \leq \text{hd } ts \ x_1$  **using** *step-ge-x0* [*of hd ts x1*] **by** *simp*  
**also from** *ts-nonempty* **have**  $\dots < x_1$  **by** (*intro step-less*) *simp-all*  
**finally show**  $x_0 < x_1$  **by** *simp*  
**thus**  $x_0 \leq x_1$  **by** *simp*

qed

lemma *akra-bazzi-induct* [*consumes 1, case-names base rec*]:

**assumes**  $x \geq x_0$   
**assumes** *base*:  $\bigwedge x. x \geq x_0 \implies x < x_1 \implies P x$   
**assumes** *rec*:  $\bigwedge x. x \geq x_1 \implies (\bigwedge t. t \in \text{set } ts \implies P (t x)) \implies P x$   
**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**  $g :: \text{nat} \Rightarrow \text{real}$

**assumes** *f-nonneg-base*:  $x \geq x_0 \implies x < x_1 \implies f x \geq 0$

**and** *f-rec*:  $x \geq x_1 \implies f x = g x + (\sum i < k. as!i * f ((ts!i) x))$

**and** *g-nonneg*:  $x \geq x_1 \implies g x \geq 0$

**and** *ex-pos-a*:  $\exists a \in \text{set } as. a > 0$

**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*  $\Rightarrow$  *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*:  $x \geq x_0 \implies f x \geq 0$

**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* *sum-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*:  $i < k \implies (hs'!i)$  (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**  $\forall h \in set\ hs. \exists c > 0. \text{eventually } (\lambda x. |h x| \leq c * real\ x / \ln (real\ x)\ powr (1 + e))$  *at-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**  $\exists c > 0. \text{eventually } (\lambda x. |h x| \leq c * x / \ln x\ powr (1 + e))$  *at-top*

**unfolding** *bigO-def* **by** *auto*

**qed**

**from** *bchoice*[*OF* *this*] **obtain** *hb* **where** *hb*:

$\forall h \in set\ hs. hb\ h > 0 \wedge \text{eventually } (\lambda x. |h x| \leq hb\ h * real\ x / \ln (real\ x)\ powr (1 + e))$  *at-top* **by** *blast*

**define** *hb'* **where**  $hb' = \max 1 (Max \{hb\ h \mid h. h \in set\ hs\})$

**have**  $hb' > 0$  **unfolding** *hb'-def* **by** *simp*

**moreover** **have**  $\forall h \in set\ hs. \text{eventually } (\lambda x. |h (nat \lfloor x \rfloor)| \leq hb' * x / \ln x\ powr (1 + e))$  *at-top*

**proof** (*intro* *ballI*, *rule* *eventually-mp*[*OF* *always-eventually eventually-conj*], *clarify*)

**fix**  $h$  **assume**  $h: h \in \text{set } hs$   
**with**  $hb$  **have**  $hb\text{-pos}: hb\ h > 0$  **by** *auto*

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

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

**lemma** *C-bound*:  
**assumes**  $\bigwedge b. b \in \text{set } bs \implies C < b\ hb > 0$   
**shows** *eventually*  $(\lambda x :: \text{real}. \forall b \in \text{set } bs. C * x \leq b * x - hb * x / \ln x \text{ powr } (1 + e))$   
*at-top*  
**proof** -  
**from**  $e\text{-pos}$  **have**  $(\lambda x. hb * \ln x \text{ powr } -(1 + e)) \longrightarrow 0$  **at-top**  
**by** (*intro tendsto-mult-right-zero tendsto-neg-powr ln-at-top*) *simp-all*  
**with**  $assms$  **have**  $\forall b \in \text{set } bs. \text{eventually } (\lambda x. |hb * \ln x \text{ powr } -(1 + e)| < b - C)$

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

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

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

**lemma** *gc2*:  $gc2 > 0$  *eventually*  $(\lambda x. g \ x \geq gc2 * g' \ (\text{real } x))$  *at-top*

**proof**–

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

**definition**  $gx0 \equiv \text{max } x_1 \ (\text{SOME } gx0. \forall x \geq gx0. g \ x \geq gc2 * g' \ (\text{real } x) \wedge f \ x >$

$0 \wedge g'(\text{real } x) \geq 0$ )

**definition**  $gx1 \equiv \max gx0 \text{ (SOME } gx1. \forall x \geq gx1. \forall i < k. (ts!i) x \geq gx0)$

**lemma**  $gx0$ :

**assumes**  $x \geq gx0$

**shows**  $g x \geq gc2 * g'(\text{real } x) \wedge f x > 0 \wedge g'(\text{real } x) \geq 0$

**proof** –

**from** *eventually-conj*[*OF*  $gc2(2)$ ] *eventually-conj*[*OF* *f-pos*] *eventually-nat-real*[*OF* *g'-nonneg*]]

**have**  $\exists gx0. \forall x \geq gx0. g x \geq gc2 * g'(\text{real } x) \wedge f x > 0 \wedge g'(\text{real } x) \geq 0$

**by** (*simp* *add: eventually-at-top-linorder*)

**note** *someI-ex*[*OF* *this*]

**moreover** **have**  $x \geq (\text{SOME } gx0. \forall x \geq gx0. g x \geq gc2 * g'(\text{real } x) \wedge f x > 0 \wedge g'(\text{real } x) \geq 0)$

**using** *assms* **unfolding**  $gx0\text{-def}$  **by** *simp*

**ultimately show**  $g x \geq gc2 * g'(\text{real } x) \wedge f x > 0 \wedge g'(\text{real } x) \geq 0$  **unfolding**  $gx0\text{-def}$  **by** *blast+*

**qed**

**lemma**  $gx1$ :

**assumes**  $x \geq gx1 \wedge i < k$

**shows**  $(ts!i) x \geq gx0$

**proof** –

**define**  $mb$  **where**  $mb = \text{Min}(\text{set } bs)/2$

**from** *b-bounds* *bs-nonempty* **have**  $mb\text{-pos}: mb > 0$  **unfolding**  $mb\text{-def}$  **by** *simp*

**from** *h-bound* **guess**  $hb$  . **note**  $hb = \text{this}$

**from** *e-pos* **have**  $((\lambda x. hb * \ln x \text{ powr } -(1+e)) \longrightarrow 0)$  *at-top*

**by** (*intro* *tendsto-mult-right-zero* *tendsto-neg-powr* *ln-at-top*) *simp-all*

**moreover** **note**  $mb\text{-pos}$

**ultimately** **have** *eventually*  $(\lambda x. hb * \ln x \text{ powr } -(1+e) < mb)$  *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**  $i x :: \text{nat}$  **assume**  $A: hb * \ln(\text{real } x) \text{ powr } -(1+e) < mb$  **and**  $i: i < k$

**assume**  $B: \forall h \in \text{set } hs'. |h(\text{real } x)| \leq hb * \text{real } x / \ln(\text{real } x) \text{ powr } (1+e)$

**with**  $i$  **have**  $B': |(hs!i)(\text{real } x)| \leq hb * \text{real } x / \ln(\text{real } x) \text{ powr } (1+e)$

**using** *length-hs'*[*symmetric*] **by** *auto*

**assume**  $C: x \geq \text{nat } \lceil gx0/mb \rceil$

**hence**  $C': \text{real } gx0/mb \leq \text{real } x$  **by** *linarith*

**assume**  $D: x \geq x_1$

**from**  $mb\text{-pos}$  **have**  $\text{real } gx0 = mb * (\text{real } gx0/mb)$  **by** *simp*

**also** **from**  $i$  *bs-nonempty* **have**  $mb \leq bs!i/2$  **unfolding**  $mb\text{-def}$  **by** *simp*

**hence**  $mb * (\text{real } gx0/mb) \leq bs!i/2 * x$

**using**  $C'$   $i$  *b-bounds*[*of*  $bs!i$ ]  $mb\text{-pos}$  **by** (*intro* *mult-mono*) *simp-all*

**also have**  $\dots = bs!i*x + -bs!i/2 * x$  **by** *simp*  
**also** {  
**have**  $-(hs!i) x \leq |(hs!i) x|$  **by** *simp*  
**also from**  $i$  *B'* *length-hs* **have**  $|(hs!i) x| \leq hb * real\ x / \ln\ (real\ x)\ powr\ (1+e)$   
**by** (*simp add: hs'-def*)  
**also from**  $A$  **have**  $hb / \ln\ x\ powr\ (1+e) \leq mb$   
**by** (*subst (asm) powr-minus (simp add: field-simps)*)  
**hence**  $hb / \ln\ x\ powr\ (1+e) * x \leq mb * x$  **by** (*intro mult-right-mono (simp-all)*)  
**hence**  $hb * x / \ln\ x\ powr\ (1+e) \leq mb * x$  **by** *simp*  
**also from**  $i$  **have**  $\dots \leq (bs!i/2) * x$  **unfolding** *mb-def* **by** (*intro mult-right-mono (simp-all)*)  
**finally have**  $-bs!i/2 * x \leq (hs!i) x$  **by** *simp*  
**}**  
**also have**  $bs!i*real\ x + (hs!i) x = real\ ((ts!i) x)$  **using**  $i$  *D decomp* **by** *simp*  
**finally show**  $(ts!i) x \geq gx0$  **by** *simp*  
**qed**  
**hence**  $\exists gx1. \forall x \geq gx1. \forall i < k. gx0 \leq (ts!i) x$  (**is**  $Ex\ ?P$ )  
**by** (*simp add: eventually-at-top-linorder*)  
**from** *someI-ex[OF this]* **have**  $?P\ (SOME\ x.\ ?P\ x)$ .  
**moreover have**  $\bigwedge x. x \geq gx1 \implies x \geq (SOME\ x.\ ?P\ x)$  **unfolding** *gx1-def* **by** *simp*  
**ultimately have**  $?P\ gx1$  **by** *blast*  
**with** *assms* **show**  $?thesis$  **by** *blast*  
**qed**

**lemma** *gx0-ge-x1*:  $gx0 \geq x_1$  **unfolding** *gx0-def* **by** *simp*  
**lemma** *gx0-le-gx1*:  $gx0 \leq gx1$  **unfolding** *gx1-def* **by** *simp*

**function**  $f2' :: nat \Rightarrow real$  **where**  
 $x < gx1 \implies f2' x = \max\ 0\ (f\ x / gc2)$   
 $| x \geq gx1 \implies f2' x = g'\ (real\ x) + (\sum\ i < k. as!i * f2'\ ((ts!i) x))$   
**using** *le-less-linear* **by** (*blast, simp-all*)  
**termination by** (*relation Wellfounded.measure*  $(\lambda x. x)$ )  
*(insert gx0-le-gx1 gx0-ge-x1, simp-all add: step-less)*

**lemma** *f2'-nonneg*:  $x \geq gx0 \implies f2' x \geq 0$   
**by** (*induction x rule: f2'.induct*)  
*(auto intro!: add-nonneg-nonneg sum-nonneg gx0 gx1 mult-nonneg-nonneg[OF a-ge-0])*

**lemma** *f2'-le-f*:  $x \geq x_0 \implies gc2 * f2' x \leq f x$

**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**  $\bigwedge i. i < k \implies x_0 \leq (ts!i) x$  **by** *simp*  
**hence**  $\bigwedge i. i < k \implies as!i * (gc2 * f2' ((ts!i) x)) \leq as!i * f ((ts!i) x)$   
**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 sum-distrib-left, intro sum-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 sum-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**  $i \in \{..<k\}$   
**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**  $a \geq A \forall a' \geq a. a' \in \mathbb{N} \implies$   
 $f \in \Omega(\lambda x. x \text{ powr } p * (1 + \text{integral } (\lambda u. g' u / u \text{ powr } (p + 1)) a' x))$   
**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*  $(\lambda x. \forall b \in \text{set } bs. C * x \leq b * x - hb * x / \ln x \text{ powr } (1 + e))$  *at-top*  
**using** *hb(1) bs-nonempty* **by** (*intro C-bound*) *simp-all*  
**moreover from** *b-bounds hb(1) e-pos*  
**have** *eventually*  $(\lambda x. \forall b \in \text{set } bs. akra-bazzi-asymptotics b hb e p x)$  *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*  $(\lambda x. (\forall h \in \text{set } hs'. |h x| \leq hb*x/\ln x \text{ powr } (1+e)) \wedge$   
 $(\forall b \in \text{set } bs. C*x \leq b*x - hb*x/\ln x \text{ powr } (1+e)) \wedge$   
 $(\forall b \in \text{set } bs. \text{akra-bazzi-asymptotics } b \text{ hb } e \text{ p } x) \wedge$   
 $(\forall b > x. \exists c. \forall x \in \{x..b\}. g' x \leq c) \wedge f2' (\text{nat } \lfloor x \rfloor) > 0 \wedge$   
 $(\forall u \in \{C * x..x\}. g' u \leq c2 * g' x) \wedge$   
 $g' x \geq 0)$  *at-top*

**by** (*intro eventually-conj*) (*force elim!: eventually-conjE*)+  
**then have**  $\exists X. (\forall x \geq X. (\forall h \in \text{set } hs'. |h x| \leq hb*x/\ln x \text{ powr } (1+e)) \wedge$   
 $(\forall b \in \text{set } bs. C*x \leq b*x - hb*x/\ln x \text{ powr } (1+e)) \wedge$   
 $(\forall b \in \text{set } bs. \text{akra-bazzi-asymptotics } b \text{ hb } e \text{ p } x) \wedge$   
 $(\forall b > x. \exists c. \forall x \in \{x..b\}. g' x \leq c) \wedge$   
 $(\forall u \in \{C * x..x\}. g' u \leq c2 * g' x) \wedge$   
 $f2' (\text{nat } \lfloor x \rfloor) > 0 \wedge g' x \geq 0)$

**by** (*subst (asm) eventually-at-top-linorder*) (*erule ex-mono, blast*)  
**then guess**  $X$  **by** (*elim exE conjE*) **note**  $X = \text{this}$

**define**  $x_0'$ -*min* **where**  $x_0'$ -*min* =  $\max A (\max X (\max a0 (\max gx1 (\max 1 (\text{real } x_1 + 1))))))$

**{**  
**fix**  $x_0' :: \text{real}$  **assume**  $x_0'$ -*props*:  $x_0' \geq x_0'$ -*min*  $x_0' \in \mathbb{N}$   
**hence**  $x_0'$ -*ge-x1*:  $x_0' \geq \text{real } (x_1 + 1)$  **and**  $x_0'$ -*ge-1*:  $x_0' \geq 1$  **and**  $x_0'$ -*ge-X*:  $x_0' \geq X$   
**}**

**unfolding**  $x_0'$ -*min-def* **by** *linarith+*

**hence**  $x_0'$ -*pos*:  $x_0' > 0$  **and**  $x_0'$ -*nonneg*:  $x_0' \geq 0$  **by** *simp-all*  
**have**  $x_0'$ :  $\forall x \geq x_0'. (\forall h \in \text{set } hs'. |h x| \leq hb*x/\ln x \text{ powr } (1+e))$   
 $\forall x \geq x_0'. (\forall b \in \text{set } bs. C*x \leq b*x - hb*x/\ln x \text{ powr } (1+e))$   
 $\forall x \geq x_0'. (\forall b \in \text{set } bs. \text{akra-bazzi-asymptotics } b \text{ hb } e \text{ p } x)$   
 $\forall a \geq x_0'. \forall b > a. \exists c. \forall x \in \{a..b\}. g' x \leq c$   
 $\forall x \geq x_0'. \forall u \in \{C * x..x\}. g' u \leq c2 * g' x$   
 $\forall x \geq x_0'. f2' (\text{nat } \lfloor x \rfloor) > 0 \forall x \geq x_0'. g' x \geq 0$

**using**  $X$   $x_0'$ -*ge-X* **by** *auto*

**from**  $x_0'$ -*props*(2) **have**  $x_0'$ -*int*:  $\text{real } (\text{nat } \lfloor x_0' \rfloor) = x_0'$  **by** (*rule real-natfloor-nat*)

**from**  $x_0'$ -*props* **have**  $x_0'$ -*ge-gx1*:  $x_0' \geq gx1$  **and**  $x_0'$ -*ge-a0*:  $x_0' \geq a0$

**unfolding**  $x_0'$ -*min-def* **by** *simp-all*

**with**  $gx0$ -*le-gx1* **have**  $f2'$ -*nonneg*:  $\bigwedge x. x \geq x_0' \implies f2' x \geq 0$  **by** (*force intro!:*  
 $f2'$ -*nonneg*)

**define**  $bm$  **where**  $bm = \text{Min } (\text{set } bs)$

**define**  $x_1'$  **where**  $x_1' = 2 * x_0' * \text{inverse } bm$

**define**  $fb2$  **where**  $fb2 = \text{Min } \{f2' x \mid x. x \in \{x_0'..x_1'\}\}$

**define**  $gb2$  **where**  $gb2 = (\text{SOME } c. \forall x \in \{x_0'..x_1'\}. g' x \leq c)$

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

**hence**  $1 < 2 * \text{inverse } bm$  **by** (*simp add: field-simps*)

**from** *mult-strict-left-mono*[*OF this*  $x_0'$ -*pos*]

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

**from**  $x_0$ -*le-x1*  $x_0'$ -*ge-x1* **have**  $ge$ - $x_0'D$ :  $\bigwedge x. x_0' \leq \text{real } x \implies x_0 \leq x$  **by** *simp*

**from**  $x0'-ge-x1$   $x0'-le-x1'$  **have**  $gt-x1'D$ :  $\bigwedge x. x_1' < real\ x \implies x_1 \leq x$  **by** *simp*

**have**  $x0'-x1'$ :  $\forall b \in set\ bs. 2 * x_0' * inverse\ b \leq x_1'$

**proof**

**fix**  $b$  **assume**  $b$ :  $b \in set\ bs$

**hence**  $bm \leq b$  **by** (*simp add: bm-def*)

**moreover from**  $b$  *bs-nonempty* *b-bounds* **have**  $bm > 0$   $b > 0$  **unfolding** *bm-def*

**by** *auto*

**ultimately have**  $inverse\ b \leq inverse\ bm$  **by** *simp*

**with**  $x0'-nonneg$  **show**  $2 * x_0' * inverse\ b \leq x_1'$

**unfolding**  $x_1'-def$  **by** (*intro mult-left-mono*) *simp-all*

**qed**

**note**  $f-nonneg' = f-nonneg$

**have**  $\bigwedge x. real\ x \geq x_0' \implies x \geq nat\ \lfloor x_0' \rfloor \bigwedge x. real\ x \leq x_1' \implies x \leq nat\ \lceil x_1 \rceil$  **by**

*linarith+*

**hence**  $\{x \mid x. real\ x \in \{x_0'..x_1'\}\} \subseteq \{x \mid x. x \in \{nat\ \lfloor x_0' \rfloor .. nat\ \lceil x_1 \rceil\}\}$  **by** *auto*

**hence** *finite*  $\{x \mid x::nat. real\ x \in \{x_0'..x_1'\}\}$  **by** (*rule finite-subset*) *auto*

**hence** *fin*: *finite*  $\{f2'\ x \mid x::nat. real\ x \in \{x_0'..x_1'\}\}$  **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* *x0'* *x1'* *hb* *e* *p*  $f2'$   $g'$

**by** (*unfold-locales*) (*auto simp: facts simp del: f2'.simps intro!: f2'.simps(2)*)

**have**  $f'-nat$ :  $\bigwedge x::nat. abr.f'\ (real\ x) = f2'\ x$

**proof**–

**fix**  $x :: nat$  **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!: sum.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 sum.cong, simp, clarify*)

**fix**  $i$  **assume**  $i$ :  $i < k$

**from**  $i\ x\ x0'-le-x1'\ x0'-ge-gx1$  **have**  $*$ :  $bs!i * real\ x + (hs!i)\ x = real\ ((ts!i)$

$x)$

**by** (*intro decomp*) *simp-all*

**also from**  $i$  **have**  $abr.f'\ \dots = f2'\ ((ts!i)\ x)$

**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**  $g'\ x + \dots = f2'\ x$  **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 x0' x1' hb e p
  integrable integral abr.f' g' C fb2 gb2 c2
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 f2' (nat ⌊x⌋) ∈ {f2' x | x. real x ∈ {x0'..x1'}} by force
  from fin and this have f2' (nat ⌊x⌋) ≥ fb2 unfolding fb2-def by (rule Min-le)
  with x show abr.f' x ≥ fb2 by simp
next
  from x0'-int x0'-le-x1' have ∃ x::nat. real x ≥ x0' ∧ real x ≤ x1'
    by (intro exI[of - nat ⌊x0'⌋]) simp-all
  moreover {
    fix x :: nat assume real x ≥ x0' ∧ real x ≤ x1'
    with x0'(6) have f2' (nat ⌊real x⌋) > 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: x0' ≤ x ≤ x1'
  with x0'(4) x0'-lt-x1' have ∃ c. ∀ x ∈ {x0'..x1'}. g' x ≤ c by force
  from someI-ex[OF this] x show g' x ≤ 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 (λx. |f x| ≥ gc2 * c5 * |f-approx (real x)|) at-top
proof (unfold eventually-at-top-linorder, intro exI allI impI)
  fix x :: nat assume x ≥ nat ⌊x0'⌋
  hence x: real x ≥ x0' by linarith
  note c5(1)[OF x]
  also have abr.f' (real x) = f2' x by (rule f'-nat)
  also have gc2 * ... ≤ 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 * |f-approx (real x)| ≤ |f x|
    using gc2 f-approx-nonneg[OF x] by (simp add: algebra-simps)

```

**qed**  
**hence**  $f \in \Omega(\lambda x. f\text{-approx } (\text{real } x))$  **using**  $gc2(1)$   $f\text{-nonneg}'$   $f\text{-approx-nonneg}$   
**by**  $(\text{intro } \text{landau-omega.bigI}[\text{of } gc2 * c5] \text{ eventually-conj}$   
 $\text{mult-pos-pos } c5 \text{ eventually-nat-real})$   $(\text{auto simp: eventually-at-top-linorder})$   
**note**  $\text{this}[\text{unfolded } f\text{-approx-def}]$   
**}**  
**moreover have**  $x_0'\text{-min} \geq A$  **unfolding**  $x_0'\text{-min-def } gx0\text{-ge-}x1$  **by**  $\text{simp}$   
**ultimately show**  $?thesis$  **by**  $(\text{intro } \text{that})$   $\text{auto}$   
**qed**

**lemma**  $\text{bigomega-f}$ :

**obtains**  $a$  **where**  $a \geq A$   $f \in \Omega(\lambda x. x \text{ powr } p * (1 + \text{integral } (\lambda u. g' u / u \text{ powr } (p+1)) a x))$

**proof** –

**from**  $\text{bigomega-f-aux}[\text{of } A]$  **guess**  $a$  . **note**  $a = \text{this}$

**define**  $a'$  **where**  $a' = \text{real } (\text{max } (\text{nat } \lceil a \rceil) 0) + 1$

**note**  $a$

**moreover have**  $a' \in \mathbb{N}$  **by**  $(\text{auto simp: max-def } a'\text{-def})$

**moreover have**  $*$ :  $a' \geq a + 1$  **unfolding**  $a'\text{-def}$  **by**  $\text{linarith}$

**moreover from**  $*$  **and**  $a$  **have**  $a' \geq A$  **by**  $\text{simp}$

**ultimately show**  $?thesis$  **by**  $(\text{intro } \text{that}[\text{of } a'])$   $\text{auto}$

**qed**

**end**

**locale**  $\text{akra-bazzi-upper} = \text{akra-bazzi-function} +$

**fixes**  $g' :: \text{real} \Rightarrow \text{real}$

**assumes**  $g'\text{-integrable}$ :  $\exists a. \forall b \geq a. \text{integrable } (\lambda u. g' u / u \text{ powr } (p + 1)) a b$

**and**  $g'\text{-growth1}$ :  $\exists C c1. c1 > 0 \wedge C < \text{Min } (\text{set } bs) \wedge$

$\text{eventually } (\lambda x. \forall u \in \{C * x..x\}. g' u \geq c1 * g' x)$   $\text{at-top}$

**and**  $g'\text{-bigo}$ :  $g \in O(g')$

**and**  $g'\text{-nonneg}$ :  $\text{eventually } (\lambda x. g' x \geq 0)$   $\text{at-top}$

**begin**

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

**lemma**  $gc1$ :  $gc1 > 0$   $\text{eventually } (\lambda x. g x \leq gc1 * g' (\text{real } x))$   $\text{at-top}$

**proof** –

**from**  $g'\text{-bigo}$  **guess**  $c$  **by**  $(\text{elim } \text{landau-o.bigE})$  **note**  $c = \text{this}$

**from**  $g'\text{-nonneg}$  **have**  $\text{eventually } (\lambda x :: \text{nat}. g' (\text{real } x) \geq 0)$   $\text{at-top}$  **by**  $(\text{rule } \text{eventually-nat-real})$

**with**  $c(2)$  **have**  $\text{eventually } (\lambda x. g x \leq c * g' (\text{real } x))$   $\text{at-top}$

**using**  $\text{eventually-ge-at-top}[\text{of } x_1]$  **by**  $\text{eventually-elim } (\text{insert } g\text{-nonneg, simp-all})$

**with**  $c(1)$  **have**  $\exists gc1. gc1 > 0 \wedge \text{eventually } (\lambda x. g x \leq gc1 * g' (\text{real } x))$   $\text{at-top}$   
**by**  $\text{blast}$

**from** *someI-ex*[*OF this*] **show**  $gc1 > 0$  eventually  $(\lambda x. g x \leq gc1 * g' (real x))$   
*at-top*  
**unfolding** *gc1-def* **by** *blast+*  
**qed**

**definition**  $gx3 \equiv \max x_1 (SOME gx0. \forall x \geq gx0. g x \leq gc1 * g' (real x))$

**lemma** *gx3*:

**assumes**  $x \geq gx3$

**shows**  $g x \leq gc1 * g' (real x)$

**proof** –

**from** *gc1*(2) **have**  $\exists gx3. \forall x \geq gx3. g x \leq gc1 * g' (real x)$  **by** (*simp add: eventually-at-top-linorder*)

**note** *someI-ex*[*OF this*]

**moreover have**  $x \geq (SOME gx0. \forall x \geq gx0. g x \leq gc1 * g' (real x))$

**using** *assms unfolding gx3-def* **by** *simp*

**ultimately show**  $g x \leq gc1 * g' (real x)$  **unfolding** *gx3-def* **by** *blast*

**qed**

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

**function**  $f' :: nat \Rightarrow real$  **where**

$x < gx3 \implies f' x = \max 0 (f x / gc1)$

$| x \geq gx3 \implies f' x = g' (real x) + (\sum i < k. as!i * f' ((ts!i) x))$

**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*:  $x \geq x_0 \implies gc1 * f' x \geq f x$

**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**  $gc1 * g' (real x) \geq g x$  **by** *force*

**moreover from** *step-ge-x0 prems(1) gx3-ge-x1*

**have**  $\bigwedge i. i < k \implies x_0 \leq \text{nat } \lfloor (ts!i) x \rfloor$  **by** (*intro le-nat-floor*) *simp*

**hence**  $\bigwedge i. i < k \implies as!i * (gc1 * f' ((ts!i) x)) \geq as!i * f ((ts!i) x)$

**using** *prems(1)* **by** (*intro mult-left-mono a-ge-0 prems(2)*) *auto*

**hence**  $gc1 * (\sum i < k. as!i * f' ((ts!i) x)) \geq (\sum i < k. as!i * f ((ts!i) x))$

**by** (*subst sum-distrib-left, intro sum-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**  $a \geq A \forall a' \geq a. a' \in \mathbf{N} \implies$

$f \in O(\lambda x. x \text{ powr } p * (1 + \text{integral } (\lambda u. g' u / u \text{ powr } (p + 1)) a' x))$

**proof** –

**from**  $g'$ -integrable **guess**  $a0$  **by** (*elim exE*) **note**  $a0 = \text{this}$   
**from**  $h$ -bound **guess**  $hb$  . **note**  $hb = \text{this}$   
**moreover from**  $g$ -growth1 **guess**  $C$   $c1$  **by** (*elim conjE exE*) **note**  $C = \text{this}$   
**hence eventually**  $(\lambda x. \forall b \in \text{set } bs. C * x \leq b * x - hb * x / \ln x \text{ powr } (1 + e))$  *at-top*  
**using**  $hb(1)$  *bs-nonempty* **by** (*intro C-bound*) *simp-all*  
**moreover from**  $b$ -bounds  $hb(1)$  *e-pos*  
**have eventually**  $(\lambda x. \forall b \in \text{set } bs. \text{akra-bazzi-asymptotics } b \text{ } hb \text{ } e \text{ } p \text{ } x)$  *at-top*  
**by** (*rule akra-bazzi-asymptotics*)  
**moreover note**  $gc1(2)$   $C(3)$   $g'$ -nonneg  
**ultimately have eventually**  $(\lambda x. (\forall h \in \text{set } hs'. |h \ x| \leq hb * x / \ln x \text{ powr } (1 + e)) \wedge$   
 $(\forall b \in \text{set } bs. C * x \leq b * x - hb * x / \ln x \text{ powr } (1 + e)) \wedge$   
 $(\forall b \in \text{set } bs. \text{akra-bazzi-asymptotics } b \text{ } hb \text{ } e \text{ } p \text{ } x) \wedge$   
 $(\forall u \in \{C * x .. x\}. g' \ u \geq c1 * g' \ x) \wedge g' \ x \geq 0)$  *at-top*  
**by** (*intro eventually-conj*) (*force elim!: eventually-conjE*) +  
**then have**  $\exists X. (\forall x \geq X. (\forall h \in \text{set } hs'. |h \ x| \leq hb * x / \ln x \text{ powr } (1 + e)) \wedge$   
 $(\forall b \in \text{set } bs. C * x \leq b * x - hb * x / \ln x \text{ powr } (1 + e)) \wedge$   
 $(\forall b \in \text{set } bs. \text{akra-bazzi-asymptotics } b \text{ } hb \text{ } e \text{ } p \text{ } x) \wedge$   
 $(\forall u \in \{C * x .. x\}. g' \ u \geq c1 * g' \ x) \wedge g' \ x \geq 0)$   
**by** (*subst (asm) eventually-at-top-linorder*) *fast*  
**then guess**  $X$  **by** (*elim exE conjE*) **note**  $X = \text{this}$

**define**  $x_0'$ -min **where**  $x_0'$ -min =  $\max A (\max X (\max 1 (\max a0 (\max gx3 (\text{real } x_1 + 1))))))$   
 $\{$   
**fix**  $x_0' :: \text{real}$  **assume**  $x_0'$ -props:  $x_0' \geq x_0'$ -min  $x_0' \in \mathbb{N}$   
**hence**  $x_0'$ -ge- $x_1$ :  $x_0' \geq \text{real } (x_1 + 1)$  **and**  $x_0'$ -ge-1:  $x_0' \geq 1$  **and**  $x_0'$ -ge- $X$ :  $x_0' \geq X$   
 $\}$   
**unfolding**  $x_0'$ -min-def **by** *linarith+*  
**hence**  $x_0'$ -pos:  $x_0' > 0$  **and**  $x_0'$ -nonneg:  $x_0' \geq 0$  **by** *simp-all*  
**have**  $x_0'$ :  $\forall x \geq x_0'. (\forall h \in \text{set } hs'. |h \ x| \leq hb * x / \ln x \text{ powr } (1 + e))$   
 $\forall x \geq x_0'. (\forall b \in \text{set } bs. C * x \leq b * x - hb * x / \ln x \text{ powr } (1 + e))$   
 $\forall x \geq x_0'. (\forall b \in \text{set } bs. \text{akra-bazzi-asymptotics } b \text{ } hb \text{ } e \text{ } p \text{ } x)$   
 $\forall x \geq x_0'. \forall u \in \{C * x .. x\}. g' \ u \geq c1 * g' \ x \ \forall x \geq x_0'. g' \ x \geq 0$   
**using**  $X$   $x_0'$ -ge- $X$  **by** *auto*  
**from**  $x_0'$ -props(2) **have**  $x_0'$ -int:  $\text{real } (\text{nat } \lfloor x_0' \rfloor) = x_0'$  **by** (*rule real-natfloor-nat*)  
**from**  $x_0'$ -props **have**  $x_0'$ -ge- $gx0$ :  $x_0' \geq gx3$  **and**  $x_0'$ -ge- $a0$ :  $x_0' \geq a0$   
**unfolding**  $x_0'$ -min-def **by** *simp-all*  
**hence**  $f'$ -nonneg:  $\bigwedge x. x \geq x_0' \implies f' \ x \geq 0$   
**using** *order.trans[OF f-nonneg f'-ge-f]*  $gc1(1)$   $x_0'$ -ge- $x_1$   $x_0'$ -le- $x_1$   
**by** (*simp add: zero-le-mult-iff del: f'.simps*)

**define**  $bm$  **where**  $bm = \text{Min } (\text{set } bs)$   
**define**  $x_1'$  **where**  $x_1' = 2 * x_0' * \text{inverse } bm$   
**define**  $fb1$  **where**  $fb1 = \text{Max } \{f' \ x \mid x. x \in \{x_0' .. x_1'\}\}$

**from**  $b$ -bounds *bs-nonempty* **have**  $bm > 0$   $bm < 1$  **unfolding**  $bm$ -def **by** *auto*  
**hence**  $1 < 2 * \text{inverse } bm$  **by** (*simp add: field-simps*)  
**from** *mult-strict-left-mono[OF this x\_0'-pos]*  
**have**  $x_0'$ -lt- $x_1'$ :  $x_0' < x_1'$  **and**  $x_0'$ -le- $x_1'$ :  $x_0' \leq x_1'$  **unfolding**  $x_1'$ -def **by**

*simp-all*

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

**have**  $x_0'\text{-}x_1': \forall b \in \text{set } bs. 2 * x_0' * \text{inverse } b \leq x_1'$

**proof**

**fix**  $b$  **assume**  $b: b \in \text{set } bs$

**hence**  $bm \leq b$  **by** (*simp add: bm-def*)

**moreover from**  $b$   $b\text{-bounds}$   $bs\text{-nonempty}$  **have**  $bm > 0$   $b > 0$  **unfolding**  $bm\text{-def}$

**by** *auto*

**ultimately have**  $\text{inverse } b \leq \text{inverse } bm$  **by** *simp*

**with**  $x_0'\text{-nonneg}$  **show**  $2 * x_0' * \text{inverse } b \leq x_1'$

**unfolding**  $x_1'\text{-def}$  **by** (*intro mult-left-mono*) *simp-all*

**qed**

**note**  $f\text{-nonneg}' = f\text{-nonneg}$

**have**  $\bigwedge x. \text{real } x \geq x_0' \implies x \geq \text{nat } \lfloor x_0' \rfloor \bigwedge x. \text{real } x \leq x_1' \implies x \leq \text{nat } \lceil x_1' \rceil$  **by**  
*linarith+*

**hence**  $\{x \mid x. \text{real } x \in \{x_0'..x_1'\}\} \subseteq \{x \mid x. x \in \{\text{nat } \lfloor x_0' \rfloor .. \text{nat } \lceil x_1' \rceil\}\}$  **by** *auto*

**hence**  $\text{finite } \{x \mid x::\text{nat}. \text{real } x \in \{x_0'..x_1'\}\}$  **by** (*rule finite-subset*) *auto*

**hence**  $\text{fin}: \text{finite } \{f' x \mid x::\text{nat}. \text{real } x \in \{x_0'..x_1'\}\}$  **by** *force*

**note**  $\text{facts} = \text{hs}'\text{-real}$   $e\text{-pos}$   $\text{length-}\text{hs}'$   $\text{length-as}$   $\text{length-bs}$   $k\text{-not-}0$   $a\text{-ge-}0$   $p\text{-props}$   
 $x_0'\text{-ge-}1$

$f'\text{-nonneg}$   $f\text{-rec}[OF\ gt\text{-}x_1'D]$   $x_0'$   $x_0'\text{-int}$   $x_0'\text{-}x_1'$   $gc1(1)$  *decomp*

**from**  $b\text{-bounds}$   $x_0'\text{-le-}x_1'$   $x_0'\text{-ge-}x_0$   $x_0'\text{-ge-}x_1$

**interpret**  $\text{abr}: \text{akra-bazzi-nat-to-real}$   $\text{as } bs$   $\text{hs}'$   $k$   $x_0'$   $x_1'$   $hb$   $e$   $p$   $f'$   $g'$

**by** (*unfold-locales*) (*auto simp add: facts simp del: f'.simps intro!: f'.simps(2)*)

**have**  $f'\text{-nat}: \bigwedge x::\text{nat}. \text{abr}.f'(\text{real } x) = f' x$

**proof**–

**fix**  $x :: \text{nat}$  **show**  $\text{abr}.f'(\text{real } (x::\text{nat})) = f' x$

**proof** (*induction real x arbitrary: x rule: abr.f'.induct*)

**case**  $(2\ x)$

**note**  $x = \text{this}(1)$  **and**  $IH = \text{this}(2)$

**from**  $x$  **have**  $\text{abr}.f'(\text{real } x) = g'(\text{real } x) + (\sum i < k. \text{as}!i * \text{abr}.f'(bs!i * \text{real } x + (hs!i) x))$

**by** (*auto simp: gt-x1'D hs'-real intro!: sum.cong*)

**also have**  $(\sum i < k. \text{as}!i * \text{abr}.f'(bs!i * \text{real } x + (hs!i) x)) = (\sum i < k. \text{as}!i * f'((ts!i) x))$

**proof** (*rule sum.cong, simp, clarify*)

**fix**  $i$  **assume**  $i: i < k$

**from**  $i$   $x$   $x_0'\text{-le-}x_1'$   $x_0'\text{-ge-}x_1$  **have**  $*$ :  $bs!i * \text{real } x + (hs!i) x = \text{real } ((ts!i) x)$

**by** (*intro decomp*) *simp-all*

**also from**  $i$  **have**  $\text{abr}.f' \dots = f'((ts!i) x)$

**by** (*subst IH[of i]*) (*simp-all add: hs'-real*)

**finally show**  $\text{as}!i * \text{abr}.f'(bs!i * \text{real } x + (hs!i) x) = \text{as}!i * f'((ts!i) x)$  **by** *simp*

```

qed
also from x have  $g' x + \dots = f' x$  using  $x_0'-le-x_1'$   $x_0'-ge-gx_0$  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  $x_0'$   $x_1'$  hb e p integrable integral abr.f'
g' C fb1 c1
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  $x \geq x_0'$ 
  show integrable ( $\lambda x. g' x / x$  powr (p + 1))  $x_0'$  x
  by (rule integrable-subinterval[of - a0 x]) (insert a0  $x_0'-ge-a0$  x, auto)
next
  fix x assume  $x \geq x_0'$   $x \leq x_1'$ 
  have  $x_0' = \text{real } (\text{nat } \lfloor x_0' \rfloor)$  by (simp add:  $x_0'-int$ )
  also from x have  $\dots \leq \text{real } (\text{nat } \lfloor x \rfloor)$  by (auto intro!: nat-mono floor-mono)
  finally have  $x_0' \leq \text{real } (\text{nat } \lfloor x \rfloor)$  .
  moreover have  $\text{real } (\text{nat } \lfloor x \rfloor) \leq x_1'$  using x  $x_0'-ge-1$  by linarith
  ultimately have  $f' (\text{nat } \lfloor x \rfloor) \in \{f' x \mid x. \text{real } x \in \{x_0'..x_1'\}\}$  by force
  from fin and this have  $f' (\text{nat } \lfloor x \rfloor) \leq fb1$  unfolding fb1-def by (rule Max-ge)
  with x show abr.f'  $x \leq fb1$  by simp
qed (insert  $x_0'(2)$   $x_0'-le-x_1'$   $x_0'-ge-x_1$  C, simp-all add: facts)

from akra-bazzi-upper guess c6 . note c6 = this
{
  fix x :: nat assume  $x \geq \text{nat } \lceil x_0' \rceil$ 
  hence  $x: \text{real } x \geq x_0'$  by linarith
  have  $f x \leq gc1 * f' x$  using  $x_0'-ge-x_1$   $x_0'-le-x_1$  by (intro f'-ge-f) simp-all
  also have  $f' x = \text{abr.f}' (\text{real } x)$  by (simp add: f'-nat)
  also note c6(1)[OF x]
  also from f-nonneg' x  $x_0'-ge-x_1$   $x_0'-le-x_1$  have  $f x = |f x|$  by simp
  also from f-approx-nonneg x have  $f\text{-approx } (\text{real } x) = |f\text{-approx } (\text{real } x)|$  by
simp
  finally have  $gc1 * c6 * |f\text{-approx } (\text{real } x)| \geq |f x|$  using gc1 by (simp add:
algebra-simps)
}
hence eventually ( $\lambda x. |f x| \leq gc1 * c6 * |f\text{-approx } (\text{real } x)|$ ) at-top
using eventually-ge-at-top[of nat  $\lceil x_0' \rceil$ ] by (auto elim!: eventually-mono)
hence  $f \in O(\lambda x. f\text{-approx } (\text{real } x))$  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  $x_0'-min-def$   $gx_3-ge-x_1$  by simp
ultimately show ?thesis by (intro that) auto

```

qed

lemma *big-o-f*:

obtains *a* where  $a > A$   $f \in O(\lambda x. x \text{ powr } p * (1 + \text{integral } (\lambda u. g' u / u \text{ powr } (p + 1)) a x))$

proof –

from *big-o-f-aux*[of *A*] guess *a* . note *a* = this

define *a'* where  $a' = \text{real } (\max (\text{nat } \lceil a \rceil) 0) + 1$

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  $*$  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*:  $\exists a. \forall b \geq a. \text{integrable } (\lambda u. g' u / u \text{ powr } (p + 1)) a b$

and *g-growth1*:  $\exists C c1. c1 > 0 \wedge C < \text{Min } (\text{set } bs) \wedge$   
eventually  $(\lambda x. \forall u \in \{C * x..x\}. g' u \geq c1 * g' x)$  at-top

and *g-growth2*:  $\exists C c2. c2 > 0 \wedge C < \text{Min } (\text{set } bs) \wedge$   
eventually  $(\lambda x. \forall u \in \{C * x..x\}. g' u \leq c2 * g' x)$  at-top

and *g-bounded*: eventually  $(\lambda a :: \text{real}. (\forall b > a. \exists c. \forall x \in \{a..b\}. g' x \leq c))$  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  $a > A$   $f \in \Theta(\lambda x. x \text{ powr } p * (1 + \text{integral } (\lambda u. g' u / u \text{ powr } (p + 1)) a x))$

proof –

from *big-o-f-aux*[of *A*] guess *a* . note *a* = this

moreover from *bigomega-f-aux*[of *A*] guess *b* . note *b* = this

let *?a* = real  $(\max (\max (\text{nat } \lceil a \rceil) (\text{nat } \lceil b \rceil)) 0) + 1$

have *?a*  $\in \mathbb{N}$  by (auto simp: max-def)

moreover have *?a*  $\geq a$  *?a*  $\geq b$  by linarith+

ultimately have  $f \in \Theta(\lambda x. x \text{ powr } p * (1 + \text{integral } (\lambda u. g' u / u \text{ powr } (p + 1)) ?a x))$

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**  $(b::\text{real}) > 0$   $b < 1$   $\text{real } x_0 \leq b * \text{real } x_1 + c$   $c < (1 - b) * \text{real } x_1$   $x_1 > 0$

**shows** *akra-bazzi-term*  $x_0$   $x_1$   $b$   $(\lambda x. \text{nat } \lfloor b * \text{real } x + c \rfloor)$

**proof** (*rule akra-bazzi-termI[OF zero-less-one]*)

**fix**  $x$  **assume**  $x: x \geq x_1$

**from** *assms*  $x$  **have**  $\text{real } x_0 \leq b * \text{real } x_1 + c$  **by** *simp*

**also from**  $x$  *assms* **have**  $\dots \leq b * \text{real } x + c$  **by** *auto*

**finally have** *step-ge-x0*:  $b * \text{real } x + c \geq \text{real } x_0$  **by** *simp*

**thus**  $\text{nat } \lfloor b * \text{real } x + c \rfloor \geq x_0$  **by** (*subst le-nat-iff*) (*simp-all add: le-floor-iff*)

**from** *assms*  $x$  **have**  $c < (1 - b) * \text{real } x_1$  **by** *simp*

**also from** *assms*  $x$  **have**  $\dots \leq (1 - b) * \text{real } x$  **by** (*intro mult-left-mono*) *simp-all*

**finally show**  $\text{nat } \lfloor b * \text{real } x + c \rfloor < x$  **using** *assms step-ge-x0*

**by** (*subst nat-less-iff*) (*simp-all add: floor-less-iff algebra-simps*)

**from** *step-ge-x0* **have**  $\text{real-of-int } \lfloor c + b * \text{real } x \rfloor = \text{real-of-int } (\text{nat } \lfloor c + b * \text{real } x \rfloor)$  **by** *linarith*

**thus**  $(b * \text{real } x) + (\lfloor b * \text{real } x + c \rfloor - (b * \text{real } x)) =$   
 $\text{real } (\text{nat } \lfloor b * \text{real } x + c \rfloor)$  **by** *linarith*

**next**

**have**  $(\lambda x::\text{nat}. \text{real-of-int } \lfloor b * \text{real } x + c \rfloor - b * \text{real } x) \in O(\lambda-. |c| + 1)$

**by** (*intro landau-o.big-mono always-eventually allI, unfold real-norm-def*) *linarith*

**also have**  $(\lambda::\text{nat}. |c| + 1) \in O(\lambda x. \text{real } x / \ln (\text{real } x) \text{ powr } (1 + 1))$  **by** *force*

**finally show**  $(\lambda x::\text{nat}. \text{real-of-int } \lfloor b * \text{real } x + c \rfloor - b * \text{real } x) \in$   
 $O(\lambda x. \text{real } x / \ln (\text{real } x) \text{ powr } (1+1))$  .

qed

**lemma** *akra-bazzi-term-floor-add'* [*akra-bazzi-term-intros*]:

**assumes**  $(b::\text{real}) > 0$   $b < 1$   $\text{real } x_0 \leq b * \text{real } x_1 + \text{real } c$   $\text{real } c < (1 - b) * \text{real } x_1$   $x_1 > 0$

**shows** *akra-bazzi-term*  $x_0$   $x_1$   $b$   $(\lambda x. \text{nat } \lfloor b * \text{real } x \rfloor + c)$

**proof** –

**from** *assms* **have** *akra-bazzi-term*  $x_0$   $x_1$   $b$   $(\lambda x. \text{nat } \lfloor b * \text{real } x + \text{real } c \rfloor)$

**by** (*rule akra-bazzi-term-floor-add*)

**also have**  $(\lambda x. \text{nat } \lfloor b * \text{real } x + \text{real } c \rfloor) = (\lambda x::\text{nat}. \text{nat } \lfloor b * \text{real } x \rfloor + c)$

**proof**

**fix**  $x :: \text{nat}$

**have**  $\lfloor b * \text{real } x + \text{real } c \rfloor = \lfloor b * \text{real } x \rfloor + \text{int } c$  **by** *linarith*

**also from** *assms* **have**  $\text{nat } \dots = \text{nat } \lfloor b * \text{real } x \rfloor + c$  **by** (*simp add: nat-add-distrib*)

**finally show**  $\text{nat } \lfloor b * \text{real } x + \text{real } c \rfloor = \text{nat } \lfloor b * \text{real } x \rfloor + c$  .



qed  
 finally show ?thesis .  
 qed

**lemma** *akra-bazzi-term-floor-subtract* [*akra-bazzi-term-intros*]:  
 assumes  $(b::real) > 0$   $b < 1$   $real\ x_0 \leq b * real\ x_1 - c$   $0 < c + (1 - b) * real\ x_1$   $x_1 > 0$   
 shows *akra-bazzi-term*  $x_0\ x_1\ b\ (\lambda x. nat\ \lfloor b * real\ x - c \rfloor)$   
 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  $(b::real) > 0$   $b < 1$   $real\ x_0 \leq b * real\ x_1 - real\ c$   $0 < real\ c + (1 - b) * real\ x_1$   $x_1 > 0$   
 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 (*intro akra-bazzi-term-floor-subtract*) *simp-all*  
 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  $nat\ \dots = nat\ \lfloor b * real\ x \rfloor - c$  **by** (*simp add: nat-diff-distrib*)  
 finally show  $nat\ \lfloor b * real\ x - real\ c \rfloor = nat\ \lfloor b * real\ x \rfloor - c$  .  
 qed  
 finally show ?thesis .  
 qed

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

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

**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**  $b * \text{real } x + c + 1 \leq \text{real } x$  **using** *assms* **by** (*simp add: algebra-simps*)  
**with** *bx-nonneg* **show**  $\lceil b * \text{real } x + c \rceil < x$  **by** (*subst nat-less-iff*) (*simp-all add: ceiling-less-iff*)

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

**show**  $b * \text{real } x + (\lceil b * \text{real } x + c \rceil - b * \text{real } x) = \text{real } (\text{nat } \lceil b * \text{real } x + c \rceil)$   
**using** *assms bx-nonneg* **by** *simp*

**next**

**have**  $(\lambda x :: \text{nat}. \text{real-of-int } \lceil b * \text{real } x + c \rceil - b * \text{real } x) \in O(\lambda \cdot. |c| + 1)$   
**by** (*intro landau-o.big-mono always-eventually allI, unfold real-norm-def*) *linarith*  
**also have**  $(\lambda \cdot :: \text{nat}. |c| + 1) \in O(\lambda x. \text{real } x / \ln (\text{real } x) \text{ powr } (1 + 1))$  **by** *force*  
**finally show**  $(\lambda x :: \text{nat}. \text{real-of-int } \lceil b * \text{real } x + c \rceil - b * \text{real } x) \in O(\lambda x. \text{real } x / \ln (\text{real } x) \text{ powr } (1+1))$  .

**qed**

**lemma** *akra-bazzi-term-ceiling-add'* [*akra-bazzi-term-intros*]:

**assumes**  $(b :: \text{real}) > 0$   $b < 1$   $\text{real } x_0 \leq b * \text{real } x_1 + \text{real } c$   $\text{real } c + 1 \leq (1 - b) * x_1$

**shows** *akra-bazzi-term*  $x_0$   $x_1$   $b$   $(\lambda x. \text{nat } \lceil b * \text{real } x \rceil + c)$

**proof** –

**from** *assms* **have** *akra-bazzi-term*  $x_0$   $x_1$   $b$   $(\lambda x. \text{nat } \lceil b * \text{real } x + \text{real } c \rceil)$

**by** (*rule akra-bazzi-term-ceiling-add*)

**also have**  $(\lambda x. \text{nat } \lceil b * \text{real } x + \text{real } c \rceil) = (\lambda x :: \text{nat}. \text{nat } \lceil b * \text{real } x \rceil + c)$

**proof**

**fix**  $x :: \text{nat}$

**from** *assms* **have**  $0 \leq b * \text{real } x$  **by** *simp*

**also have**  $b * \text{real } x \leq \text{real-of-int } \lceil b * \text{real } x \rceil$  **by** *linarith*

**finally have** *bx-nonneg*:  $\lceil b * \text{real } x \rceil \geq 0$  **by** *simp*

**have**  $\lceil b * \text{real } x + \text{real } c \rceil = \lceil b * \text{real } x \rceil + \text{int } c$  **by** *linarith*

**also from** *assms bx-nonneg* **have**  $\text{nat } \dots = \text{nat } \lceil b * \text{real } x \rceil + c$

**by** (*subst nat-add-distrib*) *simp-all*

**finally show**  $\text{nat } \lceil b * \text{real } x + \text{real } c \rceil = \text{nat } \lceil b * \text{real } x \rceil + c$  .

**qed**

**finally show** *?thesis* .

**qed**

**lemma** *akra-bazzi-term-ceiling-subtract* [*akra-bazzi-term-intros*]:

**assumes**  $(b :: \text{real}) > 0$   $b < 1$   $\text{real } x_0 \leq b * \text{real } x_1 - c$   $1 \leq c + (1 - b) * x_1$

**shows** *akra-bazzi-term*  $x_0$   $x_1$   $b$   $(\lambda x. \text{nat } \lceil b * \text{real } x - c \rceil)$

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  $(b::real) > 0$   $b < 1$   $real\ x_0 \leq b * real\ x_1 - real\ c$   $1 \leq real\ c + (1 - b) * x_1$

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  $0 \leq b * real\ x$  by simp

also have  $b * real\ x \leq real\ of\ int\ \lceil b * real\ x \rceil$  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  $nat\ \dots = nat\ \lceil b * real\ x \rceil - c$  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  $(b::real) > 0$   $b < 1$   $real\ x_0 \leq b * real\ x_1$   $1 \leq (1 - b) * x_1$

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**

HOL-Analysis.Equivalence-Lebesgue-Henstock-Integration

Akra-Bazzi-Library

Akra-Bazzi

**begin**

**lemma** fundamental-theorem-of-calculus-real:

$a \leq b \implies \forall x \in \{a..b\}. (f\ has\ real\ derivative\ f'\ x) (at\ x\ within\ \{a..b\}) \implies (f'\ has\ integral\ (f\ b - f\ a)) \{a..b\}$

by (intro fundamental-theorem-of-calculus ballI)

(simp-all add: has-real-derivative-iff-has-vector-derivative[symmetric])

**lemma** integral-powr:

$y \neq -1 \implies a \leq b \implies a > 0 \implies integral\ \{a..b\}\ (\lambda x. x\ powr\ y :: real) =$

```

    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 ≠ -1 ⇒ a ≤ b ⇒ a > 1 ⇒ integral {a..b} (λx. 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 ≤ b ⇒ a > 1 ⇒ integral {a..b} (λx. 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 (λf a b. f integrable-on {a..b}) (λf 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 ⇒ 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

interpretation akra-bazzi-integral λf a b. f integrable-on {a..b} λf a b. integral
{a..b} f
  by (rule akra-bazzi-integral-kurzweil-henstock)

sublocale akra-bazzi-function x₀ x₁ k as bs ts f λf a b. f integrable-on {a..b}
  λf 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 (λx. g x ≥ 0) at-top
  using g-nonneg by (force simp: eventually-at-top-linorder)

```

**private lemma** *g-pos*:  
**assumes**  $g \in \Omega(h)$   
**assumes** *eventually*  $(\lambda x. h\ x > 0)$  *at-top*  
**shows** *eventually*  $(\lambda x. g\ x > 0)$  *at-top*  
**proof** –  
**from** *landau-omega.bigE-nonneg-real*[*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**  $g \in \Omega(h)$   
**assumes** *eventually*  $(\lambda x. h\ x > 0)$  *at-top*  
**shows** *eventually*  $(\lambda x. f\ x > 0)$  *at-top*  
**using** *g-pos*[*OF assms(1,2)*] *eventually-ge-at-top*[*of x<sub>1</sub>*]  
**by** (*eventually-elim*) (*subst f-rec*, *insert step-ge-x0*,  
*auto intro!*: *add-pos-nonneg sum-nonneg mult-nonneg-nonneg*[*OF a-ge-0*]  
*f-nonneg*)

**lemma** *bs-lower-bound*:  $\exists C > 0. \forall b \in \text{set } bs. C < b$   
**proof** (*intro exI conjI ballI*)  
**from** *b-pos* **show**  $A: \text{Min } (\text{set } bs) / 2 > 0$  **by** *auto*  
**fix**  $b$  **assume**  $b: b \in \text{set } bs$   
**from**  $A$  **have**  $\text{Min } (\text{set } bs) / 2 < \text{Min } (\text{set } bs)$  **by** *simp*  
**also from**  $b$  **have**  $\dots \leq b$  **by** *simp*  
**finally show**  $\text{Min } (\text{set } bs) / 2 < b$  .  
**qed**

**private lemma** *powr-growth2*:  
 $\exists C\ c2. 0 < c2 \wedge C < \text{Min } (\text{set } bs) \wedge$   
*eventually*  $(\lambda x. \forall u \in \{C * x..x\}. c2 * x\ \text{powr } p' \geq u\ \text{powr } p')$  *at-top*  
**proof** (*intro exI conjI allI ballI*)  
**define**  $C$  **where**  $C = \text{Min } (\text{set } bs) / 2$   
**from** *b-bounds bs-nonempty* **have** *C-pos*:  $C > 0$  **unfolding** *C-def* **by** *auto*  
**thus**  $C < \text{Min } (\text{set } bs)$  **unfolding** *C-def* **by** *simp*  
**show**  $\max (C\ \text{powr } p')\ 1 > 0$  **by** *simp*  
**show** *eventually*  $(\lambda x. \forall u \in \{C * x..x\}. \max ((\text{Min } (\text{set } bs) / 2)\ \text{powr } p')\ 1 * x\ \text{powr } p' \geq u\ \text{powr } p')$  *at-top*  
**using** *eventually-gt-at-top*[*of 0::real*] **apply** *eventually-elim*  
**proof** *clarify*  
**fix**  $x\ u$  **assume**  $x: x > 0$  **and**  $u \in \{C * x..x\}$   
**hence**  $u: u \geq C * x$  **and**  $u \leq x$  **unfolding** *C-def* **by** *simp-all*  
**from**  $u$  **have**  $u\ \text{powr } p' \leq \max ((C * x)\ \text{powr } p')\ (x\ \text{powr } p')$  **using** *C-pos x*  
**by** (*intro powr-upper-bound mult-pos-pos*) *simp-all*  
**also from**  $u\ x\ \text{C-pos}$  **have**  $\max ((C * x)\ \text{powr } p')\ (x\ \text{powr } p') = x\ \text{powr } p' * \max (C\ \text{powr } p')\ 1$   
**by** (*subst max-mult-left*) (*simp-all add: powr-mult algebra-simps*)  
**finally show**  $u\ \text{powr } p' \leq \max ((\text{Min } (\text{set } bs) / 2)\ \text{powr } p')\ 1 * x\ \text{powr } p'$

by (simp add: C-def algebra-simps)  
qed  
qed

**private lemma** *powr-growth1*:

$\exists C c1. 0 < c1 \wedge C < \text{Min}(\text{set } bs) \wedge$   
*eventually*  $(\lambda x. \forall u \in \{C * x..x\}. c1 * x \text{ powr } p' \leq u \text{ powr } p')$  *at-top*

**proof** (intro exI conjI allI ballI)

**define** *C* **where**  $C = \text{Min}(\text{set } bs) / 2$

**from** *b-bounds bs-nonempty* **have** *C-pos*:  $C > 0$  **unfolding** *C-def* **by** *auto*

**thus**  $C < \text{Min}(\text{set } bs)$  **unfolding** *C-def* **by** *simp*

**from** *C-pos* **show**  $\text{min}(C \text{ powr } p') > 0$  **by** *simp*

**show** *eventually*  $(\lambda x. \forall u \in \{C * x..x\}.$

$\text{min}((\text{Min}(\text{set } bs)/2) \text{ powr } p') * x \text{ powr } p' \leq u \text{ powr } p')$  *at-top*

**using** *eventually-gt-at-top*[of *0::real*] **apply** *eventually-elim*

**proof** *clarify*

**fix** *x u* **assume**  $x > 0$  **and**  $u \in \{C * x..x\}$

**hence**  $u \geq C * x$   $u \leq x$  **unfolding** *C-def* **by** *simp-all*

**from** *u x C-pos* **have**  $x \text{ powr } p' * \text{min}(C \text{ powr } p') = \text{min}((C * x) \text{ powr } p')$   
(*x powr p'*)

**by** (*subst min-mult-left*) (*simp-all add: powr-mult algebra-simps*)

**also from** *u* **have**  $u \text{ powr } p' \geq \text{min}((C * x) \text{ powr } p') (x \text{ powr } p')$  **using** *C-pos x*

**by** (*intro powr-lower-bound mult-pos-pos*) *simp-all*

**finally show**  $u \text{ powr } p' \geq \text{min}((\text{Min}(\text{set } bs)/2) \text{ powr } p') * x \text{ powr } p'$

**by** (*simp add: C-def algebra-simps*)

qed

qed

**private lemma** *powr-ln-powr-lower-bound*:

$a > 1 \implies a \leq x \implies x \leq b \implies$

$\text{min}(a \text{ powr } p) (b \text{ powr } p) * \text{min}(\ln a \text{ powr } p') (\ln b \text{ powr } p') \leq x \text{ powr } p * \ln x \text{ powr } p'$

**by** (*intro mult-mono powr-lower-bound*) (*auto intro: min.coboundedI1*)

**private lemma** *powr-ln-powr-upper-bound*:

$a > 1 \implies a \leq x \implies x \leq b \implies$

$\text{max}(a \text{ powr } p) (b \text{ powr } p) * \text{max}(\ln a \text{ powr } p') (\ln b \text{ powr } p') \geq x \text{ powr } p * \ln x \text{ powr } p'$

**by** (*intro mult-mono powr-upper-bound*) (*auto intro: max.coboundedI1*)

**private lemma** *powr-ln-powr-upper-bound'*:

*eventually*  $(\lambda a. \forall b > a. \exists c. \forall x \in \{a..b\}. x \text{ powr } p * \ln x \text{ powr } p' \leq c)$  *at-top*

**by** (*subst eventually-at-top-dense*) (*force intro: powr-ln-powr-upper-bound*)

**private lemma** *powr-upper-bound'*:

*eventually*  $(\lambda a::\text{real}. \forall b > a. \exists c. \forall x \in \{a..b\}. x \text{ powr } p' \leq c)$  *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**  $(C::\text{real}) > 0$

**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*:  $\exists C c1. 0 < c1 \wedge C < \text{Min} (\text{set } bs) \wedge$

*eventually*  $(\lambda x. \forall u \in \{C * x..x\}. c1 * (x \text{ powr } r * \ln x \text{ powr } r') \leq u \text{ powr } r * \ln$   
 $u \text{ powr } r')$  *at-top*

**proof** (*intro exI conjI*)

**let**  $?C = \text{Min} (\text{set } bs) / 2$  **and**  $?f = \lambda x. x \text{ powr } r * \ln x \text{ powr } r'$

**define**  $C$  **where**  $C = ?C$

**from** *b-bounds* **have**  $C\text{-pos}$ :  $C > 0$  **unfolding**  $C\text{-def}$  **by** *simp*

**let**  $?T = \min (C \text{ powr } r) (1 \text{ powr } r) * \min ((1/2) \text{ powr } r') (1 \text{ powr } r')$

**from**  $C\text{-pos}$  **show**  $?T > 0$  **unfolding**  $\text{min-def}$  **by** (*auto split: if-split*)

**from** *bs-nonempty b-bounds* **have**  $C\text{-pos}$ :  $C > 0$  **unfolding**  $C\text{-def}$  **by** *simp*

**thus**  $C < \text{Min} (\text{set } bs)$  **by** (*simp add: C-def*)

**show** *eventually*  $(\lambda x. \forall u \in \{C*x..x\}. ?T * ?f x \leq ?f u)$  *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**  $x: x > \text{max } 1 (\text{inverse } C)$  **and**  $u: u \in \{C*x..x\}$

**hence**  $x'$ :  $x > 1$  **by** (*simp add: field-simps*)

**with**  $C\text{-pos}$  **have**  $x\text{-pos}$ :  $x > 0$  **by** (*simp add: field-simps*)

**from**  $x u C\text{-pos}$  **have**  $u'$ :  $u > 1$  **by** (*simp add: field-simps*)

**assume**  $A$ :  $\ln (C*x) / \ln x > 1/2$

**have**  $\min (C \text{ powr } r) (1 \text{ powr } r) \leq (u/x) \text{ powr } r$

**using**  $x u u' C\text{-pos}$  **by** (*intro powr-lower-bound*) (*simp-all add: field-simps*)

**moreover** {

**note**  $A$

**also from**  $C\text{-pos}$   $x' u u'$  **have**  $\ln (C*x) \leq \ln u$  **by** (*subst ln-le-cancel-iff*)

*simp-all*

**with**  $x'$  **have**  $\ln (C*x) / \ln x \leq \ln u / \ln x$  **by** (*simp add: field-simps*)

**finally have**  $\min ((1/2) \text{ powr } r') (1 \text{ powr } r') \leq (\ln u / \ln x) \text{ powr } r'$

**using**  $x u u' C\text{-pos} A$  **by** (*intro powr-lower-bound*) *simp-all*

}

**ultimately have**  $?T \leq (u/x) \text{ powr } r * (\ln u / \ln x) \text{ powr } r'$

**using**  $x\text{-pos}$  **by** (*intro mult-mono*) *simp-all*

**also from**  $x u u'$  **have**  $\dots = ?f u / ?f x$  **by** (*simp add: powr-divide*)

**finally show**  $?T * ?f x \leq ?f u$  **using**  $x'$  **by** (*simp add: field-simps*)

qed  
qed

**private lemma** *powr-ln-powr-growth2*:  $\exists C c1. 0 < c1 \wedge C < \text{Min}(\text{set } bs) \wedge$   
*eventually*  $(\lambda x. \forall u \in \{C * x..x\}. c1 * (x \text{ powr } r * \ln x \text{ powr } r') \geq u \text{ powr } r * \ln$   
 $u \text{ powr } r')$  *at-top*

**proof** (*intro exI conjI*)

let  $?C = \text{Min}(\text{set } bs) / 2$  and  $?f = \lambda x. x \text{ powr } r * \ln x \text{ powr } r'$

define  $C$  where  $C = ?C$

let  $?T = \max(C \text{ powr } r) (1 \text{ powr } r) * \max((1/2) \text{ powr } r') (1 \text{ powr } r')$

show  $?T > 0$  by *simp*

from *b-bounds bs-nonempty* have  $C\text{-pos}: C > 0$  unfolding *C-def* by *simp*

thus  $C < \text{Min}(\text{set } bs)$  by (*simp add: C-def*)

show *eventually*  $(\lambda x. \forall u \in \{C * x..x\}. ?T * ?f x \geq ?f u)$  *at-top*

using *eventually-gt-at-top*[of  $\max 1 (\text{inverse } C)$ ] *eventually-ln-const*[OF *C-pos*]

apply *eventually-elim*

**proof** *clarify*

fix  $x u$  assume  $x: x > \max 1 (\text{inverse } C)$  and  $u: u \in \{C * x..x\}$

hence  $x': x > 1$  by (*simp add: field-simps*)

with *C-pos* have  $x\text{-pos}: x > 0$  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  $?f u / ?f x = (u/x) \text{ powr } r * (\ln u / \ln x) \text{ powr } r'$  by (*simp*  
*add: powr-divide*)

also {

have  $(u/x) \text{ powr } r \leq \max(C \text{ powr } r) (1 \text{ powr } r)$

using  $x u u' C\text{-pos}$  by (*intro powr-upper-bound*) (*simp-all add: field-simps*)

moreover {

note  $A$

also from *C-pos*  $x' u u'$  have  $\ln(C * x) \leq \ln u$  by (*subst ln-le-cancel-iff*)

*simp-all*

with  $x'$  have  $\ln(C * x) / \ln x \leq \ln u / \ln x$  by (*simp add: field-simps*)

finally have  $(\ln u / \ln x) \text{ powr } r' \leq \max((1/2) \text{ powr } r') (1 \text{ powr } r')$

using  $x u u' C\text{-pos} A$  by (*intro powr-upper-bound*) *simp-all*

} ultimately have  $(u/x) \text{ powr } r * (\ln u / \ln x) \text{ powr } r' \leq ?T$

using  $x\text{-pos}$  by (*intro mult-mono*) *simp-all*

}

finally show  $?T * ?f x \geq ?f u$  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*:

$\exists a::\text{real}. \forall b \geq a. (\lambda u. u \text{ powr } r * \ln u \text{ powr } s / u \text{ powr } t)$  *integrable-on*  $\{a..b\}$

$\exists a::\text{real}. \forall b \geq a. (\lambda u. u \text{ powr } r / u \text{ powr } s)$  *integrable-on*  $\{a..b\}$

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



**private lemma** *master-integral*:

**fixes**  $a p p' :: \text{real}$

**assumes**  $p: p \neq p'$  **and**  $a: a > 0$

**obtains**  $c d$  **where**  $c \neq 0 p > p' \longrightarrow d \neq 0$

$(\lambda x :: \text{nat. } x \text{ powr } p * (1 + \text{integral } \{a..x\} (\lambda u. u \text{ powr } p' / u \text{ powr } (p+1)))) \in$   
 $\Theta(\lambda x :: \text{nat. } d * x \text{ powr } p + c * x \text{ powr } p')$

**proof** –

**define**  $e$  **where**  $e = a \text{ powr } (p' - p)$

**from** *assms* **have**  $e: e \geq 0$  **by** (*simp add: e-def*)

**define**  $c$  **where**  $c = \text{inverse } (p' - p)$

**define**  $d$  **where**  $d = 1 - \text{inverse } (p' - p) * e$

**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**  $\text{integral } \{a..x\} (\lambda u. u \text{ powr } p' / u \text{ powr } (p+1)) =$   
 $\text{integral } \{a..x\} (\lambda u. u \text{ powr } (p' - (p + 1)))$

**by** (*intro Henstock-Kurzweil-Integration.integral-cong*) (*simp-all add: powr-diff*  
*[symmetric]*)

**also** **have**  $\dots = \text{inverse } (p' - p) * (x \text{ powr } (p' - p) - a \text{ powr } (p' - p))$

**using**  $p x0-less-x1 a x$  **by** (*simp add: integral-powr*)

**also** **have**  $x \text{ powr } p * (1 + \dots) = d * x \text{ powr } p + c * x \text{ powr } p'$

**using**  $p$  **unfolding** *c-def d-def* **by** (*simp add: algebra-simps powr-diff e-def*)

**finally** **show**  $x \text{ powr } p * (1 + \text{integral } \{a..x\} (\lambda u. u \text{ powr } p' / u \text{ powr } (p+1)))$

$=$

$$d * x \text{ powr } p + c * x \text{ powr } p' .$$

**qed**

**qed**

**private lemma** *master-integral'*:

**fixes**  $a p p' :: \text{real}$

**assumes**  $p': p' \neq 0$  **and**  $a: a > 1$

**obtains**  $c d :: \text{real}$  **where**  $p' < 0 \longrightarrow c \neq 0 d \neq 0$

$(\lambda x :: \text{nat. } x \text{ powr } p * (1 + \text{integral } \{a..x\} (\lambda u. u \text{ powr } p * \ln u \text{ powr } (p'-1) / u$   
 $\text{powr } (p+1)))) \in$

$\Theta(\lambda x :: \text{nat. } c * x \text{ powr } p + d * x \text{ powr } p * \ln x \text{ powr } p')$

**proof** –

**define**  $e$  **where**  $e = \ln a \text{ powr } p'$

**from** *assms* **have**  $e: e > 0$  **by** (*simp add: e-def*)

**define**  $c$  **where**  $c = 1 - \text{inverse } p' * e$

**define**  $d$  **where**  $d = \text{inverse } p'$

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

**qed**  
**qed**

**private lemma** *master-integral''*:

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

**qed**  
**also have**  $(\lambda x. (1 - \ln(\ln a)) * x^{\text{powr } p} + x^{\text{powr } p} * \ln(\ln x)) \in$   
 $\Theta(\lambda x. x^{\text{powr } p} * \ln(\ln x))$  **by** *simp*  
**finally show**  $?f \in \Theta(\lambda a. a^{\text{powr } p} * \ln(\ln a))$  .  
**qed**

**lemma** *master1-bigo*:  
**assumes** *g-bigo*:  $g \in O(\lambda x. \text{real } x \text{ powr } p')$   
**assumes** *less-p'*:  $(\sum i < k. \text{as!}i * \text{bs!}i \text{ powr } p') > 1$   
**shows**  $f \in O(\lambda x. \text{real } x \text{ powr } p)$   
**proof** –  
**interpret** *akra-bazzi-upper*  $x_0 \ x_1 \ k \ \text{as} \ \text{bs} \ \text{ts} \ f$   
 $\lambda f \ a \ b. f \ \text{integrable-on} \ \{a..b\} \ \lambda f \ a \ b. \ \text{integral} \ \{a..b\} \ f \ g \ \lambda x. \ x \ \text{powr} \ p'$   
**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**  $a = \text{this}$   
**note**  $a(2)$   
**also from**  $a(1)$  *less-p* *x0-less-x1* **have**  $p \neq p'$  **by** *simp-all*  
**from** *master-integral*[*OF*  $\text{this } a(1)$ ] **guess**  $c \ d$  . **note**  $cd = \text{this}$   
**note**  $cd(3)$   
**also from**  $cd(1,2)$  *less-p*  
**have**  $(\lambda x::\text{nat}. d * \text{real } x \text{ powr } p + c * \text{real } x \text{ powr } p') \in \Theta(\lambda x. \text{real } x \text{ powr } p)$   
**by** *force*  
**finally show**  $f \in O(\lambda x::\text{nat}. x \text{ powr } p)$  .  
**qed**

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

**lemma** *master2-3*:  
**assumes** *g-bigheta*:  $g \in \Theta(\lambda x. \text{real } x \text{ powr } p * \ln(\text{real } x) \text{ powr } (p' - 1))$   
**assumes** *p'*:  $p' > 0$   
**shows**  $f \in \Theta(\lambda x. \text{real } x \text{ powr } p * \ln(\text{real } x) \text{ powr } p')$   
**proof** –  
**have** *eventually*  $(\lambda x::\text{real}. x \text{ powr } p * \ln \ x \ \text{powr} \ (p' - 1) > 0)$  *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* *bighetaD2*[*OF* *g-bigheta*] *eventually-nat-real*])  
**then interpret** *akra-bazzi*  $x_0 \ x_1 \ k \ \text{as} \ \text{bs} \ \text{ts} \ f$   
 $\lambda f \ a \ b. f \ \text{integrable-on} \ \{a..b\} \ \lambda f \ a \ b. \ \text{integral} \ \{a..b\} \ f \ g \ \lambda x. \ x \ \text{powr} \ p * \ln \ x \ \text{powr} \ (p' - 1)$   
**using** *assms* *growths* *bounds* *master-integrable* **by** *unfold-locales* (*assumption* | *simp*)+  
**from** *bigheta-f*[*of* 1] **guess**  $a$  . **note**  $a = \text{this}$

**note**  $a(2)$   
**also from**  $a(1)$   $p' \neq 0$  **by** *simp-all*  
**from** *master-integral'*[*OF*  $a(1)$ , *of*  $p$ ] **guess**  $c d$  . **note**  $cd = \text{this}$   
**note**  $cd(3)$   
**also have**  $(\lambda x::\text{nat. } c * \text{real } x \text{ powr } p + d * \text{real } x \text{ powr } p * \ln (\text{real } x) \text{ powr } p') \in$   
 $\Theta(\lambda x::\text{nat. } x \text{ powr } p * \ln x \text{ powr } p')$  **using**  $cd(1,2)$   $p'$  **by** *force*  
**finally show**  $f \in \Theta(\lambda x. \text{real } x \text{ powr } p * \ln (\text{real } x) \text{ powr } p')$  .  
**qed**

**lemma** *master2-1*:

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

**lemma** *master2-2*:

**assumes**  $g\text{-bigtheta}$ :  $g \in \Theta(\lambda x. \text{real } x \text{ powr } p / \ln (\text{real } x))$   
**shows**  $f \in \Theta(\lambda x. \text{real } x \text{ powr } p * \ln (\ln (\text{real } x)))$   
**proof** –  
**have** *eventually*  $(\lambda x::\text{real. } x \text{ powr } p / \ln x > 0)$  *at-top*  
**using** *eventually-gt-at-top*[*of*  $1::\text{real}$ ] **by** *eventually-elim simp*  
**hence** *eventually*  $(\lambda x. f x > 0)$  *at-top*  
**by** (*rule* *f-pos*[*OF* *bigthetaD2*[*OF*  $g\text{-bigtheta}$ ] *eventually-nat-real*])  
**moreover from**  $g\text{-bigtheta}$  **have**  $g\text{-bigtheta}'$ :  $g \in \Theta(\lambda x. \text{real } x \text{ powr } p * \ln (\text{real } x) \text{ powr } -1)$   
**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  $\{a..b\}$   $\lambda f a b. \text{integral } \{a..b\} f g \lambda x. x \text{ powr } p * \ln x \text{ powr}$ 
   $-1$ 
  using assms growths bounds master-integrable by unfold-locales (assumption |
simp)+
  from bigheta-f[of 1] guess  $a$  . note  $a = \text{this}$ 
  note  $a(2)$ 
  also note master-integral''[OF a(1)]
  finally show  $f \in \Theta(\lambda x::\text{nat}. x \text{ powr } p * \ln (\ln x))$  .
qed

```

**lemma** *master3*:

```

assumes g-bigheta:  $g \in \Theta(\lambda x. \text{real } x \text{ powr } p')$ 
assumes p'-greater':  $(\sum_{i < k}. as!i * bs!i \text{ powr } p') < 1$ 
shows  $f \in \Theta(\lambda x. \text{real } x \text{ powr } p')$ 
proof -
  have eventually  $(\lambda x::\text{real}. x \text{ powr } p' > 0)$  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 bighetaD2[OF g-bigheta] eventually-nat-real])
  then interpret akra-bazzi  $x_0 x_1 k$  as  $bs ts f$ 
     $\lambda f a b. f$  integrable-on  $\{a..b\}$   $\lambda f a b. \text{integral } \{a..b\} f g \lambda x. x \text{ powr } p'$ 
    using assms growths bounds master-integrable by unfold-locales (assumption |
simp)+
    from p'-greater' have p'-greater:  $p' > p$  by (rule p-lessI)
    from bigheta-f[of 0] guess  $a$  . note  $a = \text{this}$ 
    note  $a(2)$ 
    also from p'-greater have  $p \neq p'$  by simp
    from master-integral[OF this a(1)] guess  $c d$  . note  $cd = \text{this}$ 
    note  $cd(3)$ 
    also have  $(\lambda x::\text{nat}. d * x \text{ powr } p + c * x \text{ powr } p') \in \Theta(\lambda x::\text{real}. x \text{ powr } p')$ 
      using p'-greater cd(1,2) by force
    finally show  $f \in \Theta(\lambda x. \text{real } x \text{ powr } p')$  .
qed

```

**end**

**end**

**end**

## 6 Evaluating expressions with rational numerals

**theory** *Eval-Numeral*

**imports**

*Complex-Main*

**begin**

**lemma** *real-numeral-to-Ratreal*:

$(0::\text{real}) = \text{Ratreal } (\text{Frct } (0, 1))$

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

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

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

**lemma** *uminus-pow-Numerals1*:  $(-(x:::\text{monoid-mult})) \wedge \text{Numerals1} = -x$  **by** *simp*

**lemmas** *power-numeral-simps = power-0 uminus-pow-Numerals1 power-minus-Bit0 power-minus-Bit1*

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

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

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

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

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

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

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

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

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

**by** *simp*

**lemma** *Frct-le*:  $\text{Frct } (a, \text{numeral } b) \leq \text{Frct } (c, \text{numeral } d) \longleftrightarrow a * \text{numeral } d \leq c * \text{numeral } b$

**by** *simp*

**lemma** *Frct-equals*:  $\text{Frct } (a, \text{numeral } b) = \text{Frct } (c, \text{numeral } d) \longleftrightarrow a * \text{numeral } d = c * \text{numeral } b$

**apply** (*intro iffI antisym*)

**apply** (*subst Frct-le[symmetric], simp*)<sup>+</sup>

**apply** (*subst Frct-le, simp*)<sup>+</sup>

**done**

**lemma** *real-power-code*:  $(\text{Ratreal } x) ^ y = \text{Ratreal } (x ^ y)$  **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** *gcd-numeral-red*:  $\text{gcd } (\text{numeral } x :: \text{int}) (\text{numeral } y) = \text{gcd } (\text{numeral } y) (\text{numeral } x \bmod \text{numeral } y)$

**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*

*fst-divmod [symmetric]*

*snd-divmod [symmetric]*

*divmod-cancel*

*divmod-steps [simplified rel-simps if-True] divmod-trivial*

*rel-simps*

**lemma** *Suc-0-to-numeral*:  $\text{Suc } 0 = \text{Numeral1}$  **by** *simp*

**lemmas** *Suc-to-numeral* = *Suc-0-to-numeral Num.Suc-1 Num.Suc-numeral*

**lemma** *rat-powr*:

*0 powr y = 0*

$x > 0 \implies x \text{ powr } \text{Ratreal } (\text{Frct } (0, \text{Numeral1})) = \text{Ratreal } (\text{Frct } (\text{Numeral1}, \text{Numeral1}))$   
 $x > 0 \implies x \text{ powr } \text{Ratreal } (\text{Frct } (\text{numeral } a, \text{Numeral1})) = x \wedge \text{numeral } a$   
 $x > 0 \implies x \text{ powr } \text{Ratreal } (\text{Frct } (-\text{numeral } a, \text{Numeral1})) = \text{inverse } (x \wedge \text{numeral } a)$   
**by** (*simp-all add: rat-number-collapse 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.bottom-right-bottom*  
*gcd.bottom-left-bottom*  
*gcd-neg1-int gcd-neg2-int gcd-numeral-red zmod-numeral-Bit0 zmod-numeral-Bit1*  
*power-numeral-simps*  
*divmod-numeral-simps numeral-One [symmetric] Groups.Let-0 Num.Let-numeral*  
*Suc-to-numeral power-numeral*  
*greaterThanLessThan-iff atLeastAtMost-iff atLeastLessThan-iff greaterThanAt-*  
*Most-iff rat-powr*  
*Num.pow.simps Num.sqr.simps Product-Type.split of-int-numeral of-int-neg-numeral*  
*of-nat-numeral*

**ML**  $\langle$   
*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*  
 $\rangle$

**lemma**  $21254387548659589512 * 314213523632464357453884361 * 2342523623324234 * 56432743858724173474$   
 $12561712738645824362329316482973164398214286 \text{ powr } 2 /$   
 $(1130246312978423123 + 231212374631082764842731842 * 122474378389424362347451251263)$   
 $>$   
 $(12313244512931247243543279768645745929475829310651205623844 :: \text{real})$   
**by** (*tactic*  $\langle \text{Eval-Numeral.eval-numeral-tac } @\{\text{context}\} 1 \rangle$ )

**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*  $L L' Lr$

**assumes**  $\bigwedge x::'a::\text{linordered-semidom. } x \geq 3 \implies f x = g x$

**shows**  $L \text{ at-top } (f) = L \text{ at-top } (g)$

**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*:  $\text{exp } (1::\text{real}) < 3$

**proof** –

**from** *Taylor-up*[of 3  $\lambda\cdot. \text{exp exp } 0 1 0$ ]

**obtain**  $t :: \text{real}$  **where**  $t > 0 \ t < 1 \ \text{exp } 1 = 5/2 + \text{exp } t / 6$  **by** (*auto simp: eval-nat-numeral*)

**note** *this*(3)

**also from**  $\langle t < 1 \rangle$  **have**  $\text{exp } t < \text{exp } 1$  **by** *simp*

**finally show**  $\text{exp } (1::\text{real}) < 3$  **by** (*simp add: field-simps*)

**qed**

**lemma** *ln-ln-pos*:

**assumes**  $(x::\text{real}) \geq 3$

**shows**  $\ln (\ln x) > 0$

**proof** (*subst ln-gt-zero-iff*)

**from** *assms exp-1-lt-3* **have**  $\ln x > \ln (\text{exp } 1)$  **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*  $x_0 \ x_1 \ bs \ ts = (\forall i < \text{length } bs. \text{akra-bazzi-term } x_0 \ x_1 \ (bs!i) \ (ts!i))$

**lemma** *akra-bazzi-termsI*:

$(\bigwedge i. i < \text{length } bs \implies \text{akra-bazzi-term } x_0 \ x_1 \ (bs!i) \ (ts!i)) \implies \text{akra-bazzi-terms } x_0 \ x_1 \ bs \ ts$

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

**lemma**  $\text{akra-bazzi-term-measure}$ :

$x \geq x_1 \implies \text{akra-bazzi-term } 0 x_1 b t \implies (t x, x) \in \text{Wellfounded.measure } (\lambda n :: \text{nat}. n)$

$x > x_1 \implies \text{akra-bazzi-term } 0 (\text{Suc } x_1) b t \implies (t x, x) \in \text{Wellfounded.measure } (\lambda n :: \text{nat}. n)$

**unfolding**  $\text{akra-bazzi-term-def by auto}$

**lemma**  $\text{measure-prod-conv}$ :

$((a, b), (c, d)) \in \text{Wellfounded.measure } (\lambda x. t (\text{fst } x)) \longleftrightarrow (a, c) \in \text{Wellfounded.measure } t$

$((e, f), (g, h)) \in \text{Wellfounded.measure } (\lambda x. t (\text{snd } x)) \longleftrightarrow (f, h) \in \text{Wellfounded.measure } t$

**by**  $\text{simp-all}$

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

**lemma**  $\text{akra-bazzi-termination-simps}$ :

**fixes**  $x :: \text{nat}$

**shows**  $a * \text{real } x / b = a/b * \text{real } x \text{ real } x / b = 1/b * \text{real } x$

**by**  $\text{simp-all}$

**lemma**  $\text{akra-bazzi-params-nonzeroI}$ :

$\text{length } as = \text{length } bs \implies$

$(\forall a \in \text{set } as. a \geq 0) \implies (\forall b \in \text{set } bs. b \in \{0 < .. < 1\}) \implies (\exists a \in \text{set } as. a > 0) \implies \text{akra-bazzi-params-nonzero } (\text{length } as) as bs \text{ by (unfold-locales, simp-all) } \square$

**lemmas**  $\text{akra-bazzi-p-rel-intros} =$

$\text{akra-bazzi-params-nonzero.p-lessI}[\text{rotated, OF - akra-bazzi-params-nonzeroI}]$

$\text{akra-bazzi-params-nonzero.p-greaterI}[\text{rotated, OF - akra-bazzi-params-nonzeroI}]$

$\text{akra-bazzi-params-nonzero.p-leI}[\text{rotated, OF - akra-bazzi-params-nonzeroI}]$

$\text{akra-bazzi-params-nonzero.p-geI}[\text{rotated, OF - akra-bazzi-params-nonzeroI}]$

$\text{akra-bazzi-params-nonzero.p-boundsI}[\text{rotated, OF - akra-bazzi-params-nonzeroI}]$

$\text{akra-bazzi-params-nonzero.p-boundsI}'[\text{rotated, OF - akra-bazzi-params-nonzeroI}]$

**lemma**  $\text{eval-length}$ :  $\text{length } [] = 0 \text{ length } (x \# xs) = \text{Suc } (\text{length } xs) \text{ by simp-all}$

**lemma**  $\text{eval-akra-bazzi-sum}$ :

$(\sum i < 0. as!i * bs!i \text{ powr } x) = 0$

$(\sum i < \text{Suc } 0. (a\#as)!i * (b\#bs)!i \text{ powr } x) = a * b \text{ powr } x$   
 $(\sum i < \text{Suc } k. (a\#as)!i * (b\#bs)!i \text{ powr } x) = a * b \text{ powr } x + (\sum i < k. as!i * bs!i \text{ powr } x)$   
**apply simp**  
**apply simp**  
**apply (induction k arbitrary: a as b bs)**  
**apply simp-all**  
**done**

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

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

**lemma ball-set-intros:**  $(\forall x \in \text{set } []. P \ x) \ P \ x \implies (\forall x \in \text{set } xs. P \ x) \implies (\forall x \in \text{set } (x\#xs). P \ x)$   
**by auto**

**lemma ball-set-simps:**  $(\forall x \in \text{set } []. P \ x) = \text{True} \ (\forall x \in \text{set } (x\#xs). P \ x) = (P \ x \wedge (\forall x \in \text{set } xs. P \ x))$   
**by auto**

**lemma bex-set-simps:**  $(\exists x \in \text{set } []. P \ x) = \text{False} \ (\exists x \in \text{set } (x\#xs). P \ x) = (P \ x \vee (\exists x \in \text{set } xs. P \ x))$   
**by auto**

**lemma eval-akra-bazzi-le-list-ex:**  
 $\text{list-ex } P \ (x\#y\#xs) \longleftrightarrow P \ x \vee \text{list-ex } P \ (y\#xs)$   
 $\text{list-ex } P \ [x] \longleftrightarrow P \ x$   
 $\text{list-ex } P \ [] \longleftrightarrow \text{False}$   
**by (auto simp: list-ex-iff)**

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

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

**lemma** *master-theorem-preprocess*:

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

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

$\Theta(\lambda n::nat. \ln(\text{real } n)) = \Theta(\lambda n::nat. real\ n\ powr\ 0 * \ln(\text{real } n)\ powr\ 1)$   
 $\Theta(\lambda n::nat. real\ n * \ln(\text{real } n)) = \Theta(\lambda n::nat. real\ n\ powr\ 1 * \ln(\text{real } n)\ powr\ 1)$   
 $\Theta(\lambda n::nat. real\ n\ powr\ p * \ln(\text{real } n)) = \Theta(\lambda n::nat. real\ n\ powr\ p * \ln(\text{real } n)\ powr\ 1)$   
 $\Theta(\lambda n::nat. \ln(\text{real } n)\ powr\ p') = \Theta(\lambda n::nat. real\ n\ powr\ 0 * \ln(\text{real } n)\ powr\ p')$   
 $\Theta(\lambda n::nat. real\ n * \ln(\text{real } n)\ powr\ p') = \Theta(\lambda n::nat. real\ n\ powr\ 1 * \ln(\text{real } n)\ powr\ p')$

**apply** (*simp-all*)

**apply** (*simp-all cong: landau-symbols[THEN landau-symbol-ge-3-cong]*)?

**done**

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

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

**definition** *CLAMP* ( $f :: nat \Rightarrow real$ )  $x = (\text{if } x < 3 \text{ then } 0 \text{ else } f\ x)$

**definition** *CLAMP'* ( $f :: nat \Rightarrow real$ )  $x = (\text{if } x < 3 \text{ then } 0 \text{ else } f\ x)$

**definition** *MASTER-BOUND*  $a\ b\ c\ x = real\ x\ powr\ a * \ln(\text{real } x)\ powr\ b * \ln(\ln(\text{real } x))\ powr\ c$

**definition** *MASTER-BOUND'*  $a\ b\ x = real\ x\ powr\ a * \ln(\text{real } x)\ powr\ b$

**definition** *MASTER-BOUND''*  $a\ x = real\ x\ powr\ a$

**lemma** *ln-1-imp-less-3*:

$\ln\ x = (1::real) \implies x < 3$

**proof** –

**assume**  $\ln\ x = 1$

**also have**  $(1::real) \leq \ln(\text{exp } 1)$  **by** *simp*

**finally have**  $\ln\ x \leq \ln(\text{exp } 1)$  **by** *simp*

**hence**  $x \leq \text{exp } 1$

**by** (*cases*  $x > 0$ ) (*force simp del: ln-exp simp add: not-less intro: order.trans*)+

**also have**  $\dots < 3$  **by** (*rule exp-1-lt-3*)

**finally show** *?thesis* .

**qed**

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

*simp*

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

**lemma** *MASTER-BOUND-postproc*:

$CLAMP (MASTER-BOUND' a 0) = CLAMP (MASTER-BOUND'' a)$   
 $CLAMP (MASTER-BOUND' a 1) = CLAMP (\lambda x. CLAMP (MASTER-BOUND'' a) x * CLAMP (\lambda x. \ln (\text{real } x)) x)$   
 $CLAMP (MASTER-BOUND' a (\text{numeral } n)) =$   
 $CLAMP (\lambda x. CLAMP (MASTER-BOUND'' a) x * CLAMP (\lambda x. \ln (\text{real } x) ^{\wedge} \text{numeral } n) x)$   
 $CLAMP (MASTER-BOUND' a (-1)) =$   
 $CLAMP (\lambda x. CLAMP (MASTER-BOUND'' a) x / CLAMP (\lambda x. \ln (\text{real } x) x)) x)$   
 $CLAMP (MASTER-BOUND' a (-\text{numeral } n)) =$   
 $CLAMP (\lambda x. CLAMP (MASTER-BOUND'' a) x / CLAMP (\lambda x. \ln (\text{real } x) ^{\wedge} \text{numeral } n) x)$   
 $CLAMP (MASTER-BOUND' a b) =$   
 $CLAMP (\lambda x. CLAMP (MASTER-BOUND'' a) x * CLAMP (\lambda x. \ln (\text{real } x) \text{ powr } b) x)$

$CLAMP (MASTER-BOUND'' 0) = CLAMP (\lambda x. 1)$   
 $CLAMP (MASTER-BOUND'' 1) = CLAMP (\lambda x. (\text{real } x))$   
 $CLAMP (MASTER-BOUND'' (\text{numeral } n)) = CLAMP (\lambda x. (\text{real } x) ^{\wedge} \text{numeral } n)$   
 $CLAMP (MASTER-BOUND'' (-1)) = CLAMP (\lambda x. 1 / (\text{real } x))$   
 $CLAMP (MASTER-BOUND'' (-\text{numeral } n)) = CLAMP (\lambda x. 1 / (\text{real } x) ^{\wedge} \text{numeral } n)$   
 $CLAMP (MASTER-BOUND'' a) = CLAMP (\lambda x. (\text{real } x) \text{ powr } a)$

**and** *MASTER-BOUND-UNCLAMP*:

$CLAMP (\lambda x. CLAMP f x * CLAMP g x) = CLAMP (\lambda x. f x * g x)$   
 $CLAMP (\lambda x. CLAMP f x / CLAMP g x) = CLAMP (\lambda x. f x / g x)$   
 $CLAMP (CLAMP f) = CLAMP f$

**unfolding** *CLAMP-def*[*abs-def*] *MASTER-BOUND'-def*[*abs-def*] *MASTER-BOUND''-def*[*abs-def*]  
**by** (*simp-all add: powr-minus divide-inverse fun-eq-iff*)

**context**

**begin**

**private lemma** *CLAMP-*:

*landau-symbol*  $L L' Lr \implies L \text{ at-top } (f::nat \Rightarrow real) \equiv L \text{ at-top } (\lambda x. CLAMP f x)$

**unfolding** *CLAMP-def*[*abs-def*]

**by** (*intro landau-symbol.cong eq-reflection*)

(*auto intro: eventually-mono[OF eventually-ge-at-top[of 3::nat]]*)

**private lemma UNCLAMP'-:**

*landau-symbol*  $L L' Lr \implies L \text{ at-top } (CLAMP' (MASTER-BOUND a b c)) \equiv L$   
*at-top*  $(MASTER-BOUND a b c)$   
**unfolding**  $CLAMP'-def[abs-def]$   $CLAMP-def[abs-def]$   
**by**  $(intro \text{ landau-symbol.cong eq-reflection})$   
 $(auto \text{ intro: eventually-mono}[OF \text{ eventually-ge-at-top}[of 3::nat]])$

**private lemma UNCLAMP-:**

*landau-symbol*  $L L' Lr \implies L \text{ at-top } (CLAMP f) \equiv L \text{ at-top } (f)$   
**using**  $\text{eventually-ge-at-top}[of 3::nat]$  **unfolding**  $CLAMP'-def[abs-def]$   $CLAMP-def[abs-def]$   
**by**  $(intro \text{ landau-symbol.cong eq-reflection})$   
 $(auto \text{ intro: eventually-mono}[OF \text{ eventually-ge-at-top}[of 3::nat]])$

**lemmas**  $CLAMP = \text{landau-symbols}[THEN CLAMP-]$

**lemmas**  $UNCLAMP' = \text{landau-symbols}[THEN UNCLAMP'-]$

**lemmas**  $UNCLAMP = \text{landau-symbols}[THEN UNCLAMP-]$

**end**

**lemma propagate-CLAMP:**

$CLAMP (\lambda x. f x * g x) = CLAMP' (\lambda x. CLAMP f x * CLAMP g x)$   
 $CLAMP (\lambda x. f x / g x) = CLAMP' (\lambda x. CLAMP f x / CLAMP g x)$   
 $CLAMP (\lambda x. \text{inverse } (f x)) = CLAMP' (\lambda x. \text{inverse } (CLAMP f x))$   
 $CLAMP (\lambda x. \text{real } x) = CLAMP' (MASTER-BOUND 1 0 0)$   
 $CLAMP (\lambda x. \text{real } x \text{ powr } a) = CLAMP' (MASTER-BOUND a 0 0)$   
 $CLAMP (\lambda x. \text{real } x \wedge a') = CLAMP' (MASTER-BOUND (\text{real } a') 0 0)$   
 $CLAMP (\lambda x. \ln (\text{real } x)) = CLAMP' (MASTER-BOUND 0 1 0)$   
 $CLAMP (\lambda x. \ln (\text{real } x) \text{ powr } b) = CLAMP' (MASTER-BOUND 0 b 0)$   
 $CLAMP (\lambda x. \ln (\text{real } x) \wedge b') = CLAMP' (MASTER-BOUND 0 (\text{real } b') 0)$   
 $CLAMP (\lambda x. \ln (\ln (\text{real } x))) = CLAMP' (MASTER-BOUND 0 0 1)$   
 $CLAMP (\lambda x. \ln (\ln (\text{real } x)) \text{ powr } c) = CLAMP' (MASTER-BOUND 0 0 c)$   
 $CLAMP (\lambda x. \ln (\ln (\text{real } x)) \wedge c') = CLAMP' (MASTER-BOUND 0 0 (\text{real } c'))$   
 $CLAMP' (CLAMP f) = CLAMP' f$   
 $CLAMP' (\lambda x. CLAMP' (MASTER-BOUND a1 b1 c1) x * CLAMP' (MASTER-BOUND$   
 $a2 b2 c2) x) =$   
 $CLAMP' (MASTER-BOUND (a1+a2) (b1+b2) (c1+c2))$   
 $CLAMP' (\lambda x. CLAMP' (MASTER-BOUND a1 b1 c1) x / CLAMP' (MASTER-BOUND$   
 $a2 b2 c2) x) =$   
 $CLAMP' (MASTER-BOUND (a1-a2) (b1-b2) (c1-c2))$   
 $CLAMP' (\lambda x. \text{inverse } (MASTER-BOUND a1 b1 c1 x)) = CLAMP' (MASTER-BOUND$   
 $(-a1) (-b1) (-c1))$   
**by**  $(\text{insert } \text{ln-1-imp-less-3'})$   
 $(\text{rule ext, simp add: } CLAMP-def \text{ } CLAMP'-def \text{ } MASTER-BOUND-def$   
 $\text{ powr-realpow powr-one}[OF \text{ ln-ln-nonneg}] \text{ powr-realpow}[OF \text{ ln-ln-pos}] \text{ powr-add}$   
 $\text{ powr-diff powr-minus})+$

**lemma numeral-assoc-simps:**

$((a::\text{real}) + \text{numeral } b) + \text{numeral } c = a + \text{numeral } (b + c)$   
 $(a + \text{numeral } b) - \text{numeral } c = a + \text{neg-numeral-class.sub } b c$

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

**lemmas** *CLAMP-aux* =  
*arith-simps numeral-assoc-simps of-nat-power of-nat-mult of-nat-numeral*  
*one-add-one numeral-One [symmetric]*

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

**context** *master-theorem-function*  
**begin**

**lemma** *master1-bigo-automation:*

**assumes**  $g \in O(\lambda x. \text{real } x \text{ powr } p')$   $1 < (\sum i < k. \text{as } ! i * \text{bs } ! i \text{ powr } p')$   
**shows**  $f \in O(\text{MASTER-BOUND } p \ 0 \ 0)$

**proof** –

**have**  $\text{MASTER-BOUND } p \ 0 \ 0 \in \Theta(\lambda x :: \text{nat}. x \text{ powr } p)$  **unfolding** *MASTER-BOUND-def[abs-def]*  
**by** (*intro landau-real-nat-transfer bigthetaI-cong*  
*eventually-mono[OF eventually-ge-at-top[of 3::real]]*) (*auto 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**  $g \in O(\text{MASTER-BOUND}'' p')$   $1 < (\sum i < k. \text{as } ! i * \text{bs } ! i \text{ powr } p')$   
*eventually*  $(\lambda x. f \ x > 0)$  *at-top*

**shows**  $f \in \Theta(\text{MASTER-BOUND } p \ 0 \ 0)$

**proof** –

**have**  $A: \text{MASTER-BOUND } p \ 0 \ 0 \in \Theta(\lambda x :: \text{nat}. x \text{ powr } p)$  **unfolding** *MAS-TER-BOUND-def[abs-def]*

**by** (*intro landau-real-nat-transfer bigthetaI-cong*  
*eventually-mono[OF eventually-ge-at-top[of 3::real]]*) (*auto dest!: ln-1-imp-less-3*)

**have**  $B: O(\text{MASTER-BOUND}'' p') = O(\lambda x :: \text{nat}. \text{real } x \text{ powr } p')$

**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**  $g \in \Theta(\text{MASTER-BOUND}' p \ p')$   $p' < -1$

**shows**  $f \in \Theta(\text{MASTER-BOUND } p \ 0 \ 0)$

**proof** –

**have**  $A: \text{MASTER-BOUND } p \ 0 \ 0 \in \Theta(\lambda x :: \text{nat}. x \text{ powr } p)$  **unfolding** *MAS-TER-BOUND-def[abs-def]*

**by** (*intro landau-real-nat-transfer bigthetaI-cong*  
*eventually-mono[OF eventually-ge-at-top[of 3::real]]*) (*auto dest!: ln-1-imp-less-3*)

**have**  $B: \Theta(\text{MASTER-BOUND}' p \ p') = \Theta(\lambda x :: \text{nat}. \text{real } x \text{ powr } p * \ln(\text{real } x) \text{ powr } p')$

**qed**

**by** (*subst CLAMP*, (*subst MASTER-BOUND-postproc MASTER-BOUND-UNCLAMP*)<sub>+</sub>,  
*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**  $g \in \Theta(\text{MASTER-BOUND}' p (-1))$   
**shows**  $f \in \Theta(\text{MASTER-BOUND } p 0 1)$   
**proof** –  
**have**  $A: \text{MASTER-BOUND } p 0 1 \in \Theta(\lambda x::\text{nat. } x \text{ powr } p * \ln (\ln x))$  **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**  $B: \Theta(\text{MASTER-BOUND}' p (-1)) = \Theta(\lambda x::\text{nat. } \text{real } x \text{ powr } p / \ln (\text{real } x))$   
**by** (*subst CLAMP*, (*subst MASTER-BOUND-postproc MASTER-BOUND-UNCLAMP*)<sub>+</sub>,  
*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**  $g \in \Theta(\text{MASTER-BOUND}' p (p' - 1))$   $p' > 0$   
**shows**  $f \in \Theta(\text{MASTER-BOUND } p p' 0)$   
**proof** –  
**have**  $A: \text{MASTER-BOUND } p p' 0 \in \Theta(\lambda x::\text{nat. } x \text{ powr } p * \ln x \text{ powr } p')$  **un-**  
**folding** *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**  $B: \Theta(\text{MASTER-BOUND}' p (p' - 1)) = \Theta(\lambda x::\text{nat. } \text{real } x \text{ powr } p * \ln x$   
*powr (p' - 1))  
**by** (*subst CLAMP*, (*subst MASTER-BOUND-postproc MASTER-BOUND-UNCLAMP*)<sub>+</sub>,  
*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**  $g \in \Theta(\text{MASTER-BOUND}'' p')$   $1 > (\sum i < k. \text{as } ! i * \text{bs } ! i \text{ powr } p')$   
**shows**  $f \in \Theta(\text{MASTER-BOUND } p' 0 0)$   
**proof** –  
**have**  $A: \text{MASTER-BOUND } p' 0 0 \in \Theta(\lambda x::\text{nat. } x \text{ powr } p')$  **unfolding** *MAS-*  
*TER-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:  $\Theta(\text{MASTER-BOUND'' } p') = \Theta(\lambda x::\text{nat. real } x \text{ powr } p')$ 
  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 (*) :: real => real => real} $ a $ b)
    val sum = @{term sum :: (nat => real) => nat set => real} $ f $ @{term
{..<k}}
    val cprop = HOLogic.mk-Trueprop (HOLogic.mk-eq (sum, @{term 1::real}))
    val cprop = Thm.cterm-of ctxt′ cprop
  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* ( $x :: \text{real}$ ) ( $y :: \text{nat}$ ) =  
 (if  $\neg (-x) + 3 / x * 5 - 1 \leq x \wedge \text{True} \vee \text{True} \longrightarrow \text{True}$  then  
 $x < \text{inverse } 3 \text{ powr } 21$  else  $x = \text{real } (\text{Suc } 0 \wedge 2 +$   
 (if  $42 - x \leq 1 \wedge 1 \text{ div } y = y \text{ mod } 2 \vee y < \text{Numeral1}$  then  $0$  else  $0$ )) + *Numeral1*)

**ML-file**  $\langle \text{akra-bazzi.ML} \rangle$

**hide-const** *arith-consts*

**method-setup** *master-theorem* =  $\langle$   
*Akra-Bazzi.setup-master-theorem*  
 $\rangle$  *automatically apply the Master theorem for recursive functions*

**method-setup** *akra-bazzi-termination* =  $\langle$   
*Scan.succeed (fn ctxt => SIMPLE-METHOD' (Akra-Bazzi.akra-bazzi-termination-tac*  
*ctxt))*  
 $\rangle$  *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*

*HOL–Decision-Procs.Approximation*

**begin**

**context** *akra-bazzi-params-nonzero*

**begin**

**lemma** *sum-alt*:  $(\sum i < k. \text{as!}i * \text{bs!}i \text{ powr } p') = (\sum i < k. \text{as!}i * \text{exp } (p' * \ln (\text{bs!}i)))$

**proof** (*intro sum.cong*)

**fix**  $i$  **assume**  $i \in \{..<k\}$

**with** *b-bounds* **have**  $\text{bs!}i > 0$  **by** *simp*

**thus**  $\text{as!}i * \text{bs!}i \text{ powr } p' = \text{as!}i * \text{exp } (p' * \ln (\text{bs!}i))$  **by** (*simp add: powr-def*)

**qed** *simp*

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

$1 < (\sum i < k. \text{as!}i * \text{exp } (p' * \ln (\text{bs!}i))) \implies p' < p$

$1 > (\sum i < k. \text{as!}i * \text{exp } (p' * \ln (\text{bs!}i))) \implies p' > p$

```

1 ≤ (∑ i<k. as!i * exp (p' * ln (bs!i))) ⇒ p' ≤ p
1 ≥ (∑ i<k. as!i * exp (p' * ln (bs!i))) ⇒ p' ≥ p
(∑ i<k. as!i * exp (x * ln (bs!i))) ≤ 1 ∧ (∑ i<k. as!i * exp (y * ln (bs!i))) ≥
1 ⇒ p ∈ {y..x}
(∑ i<k. as!i * exp (x * ln (bs!i))) < 1 ∧ (∑ i<k. as!i * exp (y * ln (bs!i))) >
1 ⇒ p ∈ {y<..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-sum:

```

```

(∑ i<0. as!i * exp (x * ln (bs!i))) = 0
(∑ i<Suc 0. (a#as)!i * exp (x * ln ((b#bs)!i))) = a * exp (x * ln b)
(∑ i<Suc k. (a#as)!i * exp (x * ln ((b#bs)!i))) = a * exp (x * ln b) +
(∑ 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-sum 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 ⇒ real) ⇒ nat ⇒ real where
  merge-sort-cost t 0 = 0
| merge-sort-cost t 1 = 1
| n ≥ 2 ⇒ 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]: (∧n. t n ≥ 0) ⇒ merge-sort-cost t x ≥ 0
by (induction t x rule: merge-sort-cost.induct) (simp-all del: One-nat-def)

```

```

lemma t ∈ Θ(λn. real n) ⇒ (∧n. t n ≥ 0) ⇒ merge-sort-cost t ∈ Θ(λn. real
n * ln (real n))
by (master-theorem 2.3) simp-all

```

### 8.2 Karatsuba multiplication

```

function karatsuba-cost :: nat ⇒ real where

```

$karatsuba-cost\ 0 = 0$   
 $| karatsuba-cost\ 1 = 1$   
 $| n \geq 2 \implies karatsuba-cost\ n =$   
 $\quad 3 * karatsuba-cost\ (nat\ \lceil real\ n / 2 \rceil) + real\ n$   
**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(λn. 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(λn. 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*  
 $strassen-cost\ 0 = 0$   
 $| strassen-cost\ 1 = 1$   
 $| n \geq 2 \implies strassen-cost\ n = 7 * strassen-cost\ (nat\ \lceil real\ n / 2 \rceil) + real\ (n^2)$   
**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(λn. 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(λn. 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*  
 $n \leq 20 \implies select-cost\ n = 0$   
 $| n > 20 \implies select-cost\ n =$   
 $\quad select-cost\ (nat\ \lfloor real\ n / 5 \rfloor) + select-cost\ (nat\ \lfloor 7 * real\ n / 10 \rfloor + 6) + 12$   
 $* real\ n / 5$

by force simp-all  
 termination by akra-bazzi-termination simp-all

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

## 8.5 Decreasing function

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

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

## 8.6 Example taken from Drmota and Szpakowski

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

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

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

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

lemma boncelet-phrase-length:  
 fixes  $p \delta :: \text{real}$  assumes  $p: p > 0 \ p < 1$  and  $\delta: \delta > 0 \ \delta < 1 \ 2 * p + \delta < 2$   
 fixes  $d :: \text{nat} \Rightarrow \text{real}$   
 defines  $q \equiv 1 - p$   
 assumes  $d\text{-nonneg}: \bigwedge n. d \ n \geq 0$   
 assumes  $d\text{-rec}: \bigwedge n. n \geq 2 \implies d \ n = 1 + p * d (\text{nat } \lfloor p * \text{real } n + \delta \rfloor) + q * d (\text{nat } \lfloor q * \text{real } n - \delta \rfloor)$   
 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*  $\Rightarrow$  *real* **where**

*n* < 200  $\implies$  *foo-cost* *n* = 0

| *n*  $\geq$  200  $\implies$  *foo-cost* *n* =

*foo-cost* (nat [real *n* / 3]) + *foo-cost* (nat [3 \* real *n* / 4] + 42) + real *n*

**by** *force simp-all*

**termination by** *akra-bazzi-termination simp-all*

**lemma** *foo-cost-nonneg* [*simp*]: *foo-cost* *n*  $\geq$  0

**by** (*induction n rule: foo-cost.induct*) *simp-all*

**lemma** *foo-cost*  $\in$   $\Theta(\lambda n. \text{real } n \text{ powr } \text{akra-bazzi-exponent } [1,1] [1/3,3/4])$

**proof** (*master-theorem 1 p': 1*)

**have**  $\forall n \geq 200. \text{foo-cost } n > 0$  **by** (*simp add: add-nonneg-pos*)

**thus** *eventually* ( $\lambda n. \text{foo-cost } n > 0$ ) *at-top* **unfolding** *eventually-at-top-linorder*

**by** *blast*

**qed** *simp-all*

**lemma** *akra-bazzi-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*  $\Rightarrow$  *real* **where**

*n*  $\leq$  20  $\implies$  *select-cost'* *n* = 0

| *n* > 20  $\implies$  *select-cost'* *n* =

*select-cost'* (nat [real *n* / 5]) + *select-cost'* (nat [*b* \* real *n*] + 6) + 6 \* real *n* + 5

**by** *force simp-all*

**termination using** *b* **by** *akra-bazzi-termination simp-all*

**lemma** *a*  $\geq$  0  $\implies$  *select-cost'*  $\in$   $\Theta(\lambda n. \text{real } n)$

**using** *b* **by** (*master-theorem 3, force+*)

**end**

## 8.9 Non-curried functions

**function** *baz-cost* :: *nat*  $\times$  *nat*  $\Rightarrow$  *real* **where**

*n*  $\leq$  2  $\implies$  *baz-cost* (*a*, *n*) = 0

| *n* > 2  $\implies$  *baz-cost* (*a*, *n*) = 3 \* *baz-cost* (*a*, nat [real *n* / 2]) + real *a*

**by** *force simp-all*

**termination by** *akra-bazzi-termination simp-all*

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

**lemma**

**assumes**  $a > 0$

**shows**  $(\lambda x. \text{baz-cost } (a, x)) \in \Theta(\lambda x. x \text{ powr } \log 2 3)$

**proof** (*master-theorem 1 p': 0*)

**from** *assms* **have**  $\forall x \geq 3. \text{baz-cost } (a, x) > 0$  **by** (*auto intro: add-nonneg-pos*)

**thus** *eventually*  $(\lambda x. \text{baz-cost } (a, x) > 0)$  *at-top* **by** (*force simp: eventually-at-top-linorder*)

**qed** (*insert assms, simp-all add: powr-divide*)

**function** *bar-cost* ::  $\text{nat} \times \text{nat} \Rightarrow \text{real}$  **where**

$n \leq 2 \implies \text{bar-cost } (a, n) = 0$

|  $n > 2 \implies \text{bar-cost } (a, n) = 3 * \text{bar-cost } (2 * a, \text{nat } \lfloor \text{real } n / 2 \rfloor) + \text{real } a$

**by** *force simp-all*

**termination** **by** *akra-bazzi-termination simp-all*

## 8.10 Ham-sandwich trees

**function** *ham-sandwich-cost* ::  $\text{nat} \Rightarrow \text{real}$  **where**

$n < 4 \implies \text{ham-sandwich-cost } n = 1$

|  $n \geq 4 \implies \text{ham-sandwich-cost } n =$

$\text{ham-sandwich-cost } (\text{nat } \lfloor n/4 \rfloor) + \text{ham-sandwich-cost } (\text{nat } \lfloor n/2 \rfloor) + 1$

**by** *force simp-all*

**termination** **by** *akra-bazzi-termination simp-all*

**lemma** *ham-sandwich-cost-pos* [*simp*]:  $\text{ham-sandwich-cost } n > 0$

**by** (*induction n rule: ham-sandwich-cost.induct*) *simp-all*

The golden ratio

**definition**  $\varphi = ((1 + \text{sqrt } 5) / 2 :: \text{real})$

**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*  $\in \Theta(\lambda n. n \text{ powr } (\log 2 \varphi))$

**proof** (*master-theorem 1 p': 0*)

**have**  $(1 / 4) \text{ powr } \log 2 \varphi + (1 / 2) \text{ powr } \log 2 \varphi =$

$\text{inverse } (2 \text{ powr } \log 2 \varphi)^{\wedge} 2 + \text{inverse } (2 \text{ powr } \log 2 \varphi)$

**by** (*simp add: powr-divide field-simps powr-powr power2-eq-square powr-mult[symmetric]*)

*del: powr-log-cancel*)

**also** **have**  $\dots = \text{inverse } (\varphi^{\wedge} 2) + \text{inverse } \varphi$  **by** (*simp add: power2-eq-square*)

**also** **have**  $\varphi + 1 = \varphi * \varphi$  **by** (*simp add: φ-def field-simps*)



**hence**  $\text{inverse } (\varphi \hat{=} 2) + \text{inverse } \varphi = 1$  **by** (*simp add: field-simps power2-eq-square*)  
**finally show**  $(1 / 4) \text{ powr log } 2 \varphi + (1 / 2) \text{ powr log } 2 \varphi = 1$  **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.