Height

By Christian Urban

June 1, 2006

Contents

theory *Height*
imports *Lam-substs*

begin

definition of the height-function by cases

constdefs
height-Var :: *name* ⇒ *int*
height-Var ≡ λ(*a*::*name*). *1*

height-App :: *int* ⇒ *int* ⇒ *int*
height-App ≡ λ*n1 n2*. (*max n1 n2*)+*1*

height-Lam :: *name* ⇒ *int* ⇒ *int*
height-Lam ≡ λ(*a*::*name*) *n*. *n*+*1*

height :: *lam* ⇒ *int*
height ≡ *itfun height-Var height-App height-Lam*

show that height is a function

lemma *supp-height-Lam*:
shows ((*supp height-Lam*)::*name set*)={}
apply(*simp add*: *height-Lam-def supp-def perm-fun-def perm-int-def*)
done

lemma *fin-supp-height*:
shows *finite* ((*supp* (*height-Var*,*height-App*,*height-Lam*))::*name set*)
by (*finite-guess add*: *height-Var-def height-App-def height-Lam-def perm-int-def fs-name1*)

lemma *FCB-height-Lam*:

shows $\exists(a::name).\ a\sharp height\text{-}Lam \wedge (\forall n.\ a\sharp height\text{-}Lam\ a\ n)$
apply(*simp add*: *height-Lam-def fresh-def supp-def perm-fun-def perm-int-def*)
done

derive the characteristic equations for height from the iteration combinator

lemma *height-Var*:
shows *height* (*Var c*) = *1*
apply(*simp add*: *height-def itfun-Var*[*OF fin-supp-height*, *OF FCB-height-Lam*])
apply(*simp add*: *height-Var-def*)
done

lemma *height-App*:
shows *height* (*App t1 t2*) = (*max* (*height t1*) (*height t2*))+*1*
apply(*simp add*: *height-def itfun-App*[*OF fin-supp-height*, *OF FCB-height-Lam*])
apply(*simp add*: *height-App-def*)
done

lemma *height-Lam*:
shows *height* (*Lam* [*a*].*t*) = (*height t*)+*1*
apply(*simp add*: *height-def*)
apply(*rule trans*)
apply(*rule itfun-Lam*[*OF fin-supp-height*, *OF FCB-height-Lam*])
apply(*simp add*: *fresh-def supp-prod supp-height-Lam*)
apply(*simp add*: *supp-def height-Var-def height-App-def perm-fun-def perm-int-def*)

apply(*simp add*: *height-Lam-def*)
done

add the characteristic equations of height to the simplifier

declare *height-Var*[*simp*] *height-App*[*simp*] *height-Lam*[*simp*]

the next lemma is needed in the Var-case of the theorem

lemma *height-ge-one*:
shows *1* ≤ (*height e*)
by (*nominal-induct e rule*: *lam.induct*) (*simp* | *arith*)+

unlike the proplem suggested by Wang, the theorem is formulated here entirely by using functions

theorem *height-subst*:
shows *height* (*e*[*x*::=*e'*]) ≤ (((*height e*) − *1*) + (*height e'*))
proof (*nominal-induct e avoiding*: *x e' rule*: *lam.induct*)
case (*Var y*)
have *1* ≤ *height e'* **by** (*rule height-ge-one*)
then show *height* (*Var y*[*x*::=*e'*]) ≤ *height* (*Var y*) − *1* + *height e'* **by** *simp*
next
case (*Lam y e1*)
hence *ih*: *height* (*e1*[*x*::=*e'*]) ≤ (((*height e1*) − *1*) + (*height e'*)) **by** *simp*
moreover

have *fresh*: $y\sharp x$ $y\sharp e'$ **by** *fact*
ultimately show *height* $((Lam\ [y].e1)[x::=e'])$ $\leq$ *height* $(Lam\ [y].e1)$ $-$ *1* $+$ *height* e' **by** *simp*
next
case $(App\ e1\ e2)$
hence *ih1*: *height* $(e1[x::=e'])$ $\leq$ $(((height\ e1) - 1) + (height\ e'))$
and *ih2*: *height* $(e2[x::=e'])$ $\leq$ $(((height\ e2) - 1) + (height\ e'))$ **by** *simp-all*
then show *height* $((App\ e1\ e2)[x::=e'])$ $\leq$ *height* $(App\ e1\ e2)$ $-$ *1* $+$ *height* e' **by** (*simp*, *arith*)
qed

end