Formalization of All Examples from [1]

Jasmin Christian Blanchette   Andrei Popescu   Dmitriy Traytel

January 30, 2015

Contents

References

[1] R. Hinze and D. W. H. James. Proving the unique fixed-point principle correct. Technical Report CS-RR-11-03, Department of Computer Science, University of Oxford, 2011.

1 Stream Examples

1.1 one

definition *one* :: *stream* **where**
one = *dtor_corec_J* (λ_. (*1*, *Inr* ())) ()

lemma *head_one*[*simp*]: *head one* = *1*
unfolding *one_def J.dtor_corec*
by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def*)

lemma *tail_one*[*simp*]: *tail one* = *one*
unfolding *one_def J.dtor_corec*
by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def*)

lemma *one_code*[*code*]: *one* = *SCons 1 one*
by (*metis J.ctor_dtor prod.collapse head_one tail_one*)

1.2 plus

definition *plus* :: *stream* ⇒ *stream* ⇒ *stream* **where**
plus s t = *dtor_corec_J* (λ(*s*, *t*). (*head s* + *head t*, *Inr* (*tail s*, *tail t*))) (*s*, *t*)

lemma *head_plus*[*simp*]: *head* (*plus s t*) = *head s* + *head t*
unfolding *plus_def J.dtor_corec map_pre_J_def BNF_Comp.id_bnf_comp_def* **by** *simp*

lemma *tail_plus*[*simp*]: *tail* (*plus s t*) = *plus* (*tail s*) (*tail t*)
unfolding *plus_def J.dtor_corec map_pre_J_def BNF_Comp.id_bnf_comp_def* **by** *simp*

lemma *plus_code*[*code*]: *plus* (*SCons m s*) (*SCons n t*) = *SCons* (*m* + *n*) (*plus s t*)
by (*smt2 J.ctor_dtor J.dtor_ctor fst_conv snd_conv prod.collapse head_plus tail_plus*)

lemma *plus_uniform*: *plus xs ys* = *algϱ1* (*xs*, *ys*)
unfolding *plus_def*
apply (*rule fun_cong*[*OF sym*[*OF J.dtor_corec_unique*]])
unfolding *algϱ1*
by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def fun_eq_iff convol_def ϱ1_def algϱ1_def*)

1.3 nat

definition *nat* :: *stream* **where**
nat = *corecUU1* (λ_. *GUARD1* (*0*, *PLS1* (*CONT1* (), *END1 one*))) ()

lemma *head_nat*[*simp*]: *head nat* = *0*
unfolding *nat_def corecUU1*
by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def J.dtor_ctor eval1_leaf1'*)

lemma *tail_nat*[*simp*]: *tail nat* = *plus nat one*
apply (*subst nat_def*) **unfolding** *corecUU1*
by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def J.dtor_ctor eval1_leaf1'*
eval1_𝔬𝔭1 algΛ1_Inr o_eq_dest[*OF Abs_Σ1_natural*] *plus_uniform nat_def*)

lemma *nat_code*[*code*]: *nat* = *SCons 0* (*plus nat one*)
by (*metis J.ctor_dtor prod.collapse head_nat tail_nat*)

1.4 id

definition *id* :: *stream* ⇒ *stream* **where**
id s = *dtor_corec_J* (λ*s*. (*head s*, *Inl* (*tail s*))) *s*

lemma *head_id*[*simp*]: *head* (*id s*) = *head s*
  unfolding *id_def J.dtor_corec*
  by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def*)

lemma *tail_id*[*simp*]: *tail* (*id s*) = *tail s*
  unfolding *id_def J.dtor_corec*
  by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def*)

lemma *id_code*[*code*]: *id* (*SCons m s*) = *SCons m s*
  by (*metis J.ctor_dtor prod.collapse head_id tail_id*)

1.5 fib

definition *fib* :: *stream* **where**
  fib = *corecUU1* (*λxs. GUARD1* (*0*, *PLS1* (*SCONS1* (*1*, *CONT1 xs*), *CONT1 xs*))) ()

lemma *head_fib*[*simp*]: *head fib* = *0*
  unfolding *fib_def corecUU1*
  by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def J.dtor_ctor eval1_leaf1'*)

lemma *tail_fib*[*simp*]: *tail fib* = *plus* (*SCons 1 fib*) *fib*
  apply (*subst fib_def*) **unfolding** *corecUU1*
  by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def J.dtor_ctor eval1_leaf1'*
    eval1_𝔬𝔭1 algΛ1_Inr o_eq_dest[*OF Abs_Σ1_natural*] *o_eq_dest*[*OF gg1_natural*]
    o_eq_dest[*OF eval1_gg1*] *plus_uniform fib_def*)

lemma *fib_code*[*code*]: *fib* = *SCons 0* (*plus* (*SCons 1 fib*) *fib*)
  by (*metis J.ctor_dtor prod.collapse head_fib tail_fib*)

1.6 sum

definition *sum* :: *stream* ⇒ *stream* **where**
  sum s = *corecUU1* (*λs. GUARD1* (*0*, *PLS1* (*END1 s*, *CONT1 s*))) *s*

lemma *head_sum*[*simp*]: *head* (*sum s*) = *0*
  unfolding *sum_def corecUU1*
  by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def J.dtor_ctor eval1_leaf1'*)

lemma *tail_sum*[*simp*]: *tail* (*sum s*) = *plus s* (*sum s*)
  apply (*subst sum_def*) **unfolding** *corecUU1*
  by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def J.dtor_ctor eval1_leaf1'*
    eval1_𝔬𝔭1 algΛ1_Inr o_eq_dest[*OF Abs_Σ1_natural*] *o_eq_dest*[*OF gg1_natural*]
    o_eq_dest[*OF eval1_gg1*] *plus_uniform sum_def*)

lemma *sum_code*[*code*]: *sum s* = *SCons 0* (*plus s* (*sum s*))
  by (*metis J.ctor_dtor prod.collapse head_sum tail_sum*)

1.7 alternate

definition *alternate* :: *stream* ⇒ *stream* ⇒ *stream* **where**
  alternate s t = *dtor_corec_J* (*λ*(*s*, *t*). (*head s*, *Inr* (*tail t*, *tail s*))) (*s*, *t*)

lemma *head_alternate*[*simp*]: *head* (*alternate s t*) = *head s*
  unfolding *alternate_def J.dtor_corec*
  by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def*)

lemma *tail_alternate*[*simp*]: *tail* (*alternate s t*) = *alternate* (*tail t*) (*tail s*)
  apply (*subst alternate_def*) **unfolding** *J.dtor_corec*
  by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def alternate_def*)

lemma *alternate_code*[*code*]: *alternate* (*SCons m s*) (*SCons n t*) = *SCons m* (*alternate t s*)
  by (*metis J.ctor_dtor J.dtor_ctor fst_conv snd_conv prod.collapse head_alternate tail_alternate*)

1.8 interleave

definition *interleave* :: *stream* ⇒ *stream* ⇒ *stream* **where**
  interleave s t = *dtor_corec_J* (λ(*s*, *t*). (*head s*, *Inr* (*t*, *tail s*))) (*s*, *t*)

lemma *head_interleave*[*simp*]: *head* (*interleave s t*) = *head s*
  unfolding *interleave_def J.dtor_corec*
  by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def*)

lemma *tail_interleave*[*simp*]: *tail* (*interleave s t*) = *interleave t* (*tail s*)
  apply (*subst interleave_def*) **unfolding** *J.dtor_corec*
  by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def interleave_def*)

lemma *interleave_code*[*code*]: *interleave* (*SCons m s*) (*SCons n t*) = *SCons m* (*interleave* (*SCons n t*) *s*)
  by (*metis J.ctor_dtor J.dtor_ctor fst_conv snd_conv prod.collapse head_interleave tail_interleave*)

lemma *interleave_uniform*: *interleave s t* = *alg*ϱ*6* (*s*, *t*)
  unfolding *interleave_def*
  apply (*rule fun_cong*[*OF sym*[*OF J.dtor_corec_unique*]])
  unfolding *alg*ϱ*6 o_def*[*symmetric*] *o_assoc*
  apply (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def fun_eq_iff J.dtor_ctor*
    ϱ*6_def Let_def convol_def eval6_*$\mathfrak{op}$*6 eval6_leaf6'*
    o_eq_dest[*OF Abs_*Σ*6_natural*] *alg*Λ*6_Inr alg*ϱ*6_def*)
  done

1.9 merge

definition *merge* **where**
  merge s t = *dtor_corec_J* (λ(*s*, *t*).
    if head s ≤ *head t then* (*head s*, *Inr* (*tail s*, *t*)) *else* (*head t*, *Inr* (*s*, *tail t*))) (*s*, *t*)

lemma *head_merge*[*simp*]: *head* (*merge s t*) = (*if head s* ≤ *head t then head s else head t*)
  unfolding *merge_def J.dtor_corec*
  by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def*)

lemma *tail_merge*[*simp*]: *tail* (*merge s t*) = (*if head s* ≤ *head t then merge* (*tail s*) *t else merge s* (*tail t*))
  apply (*subst merge_def*) **unfolding** *J.dtor_corec*
  by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def merge_def*)

lemma *merge_code*[*code*]:
  merge (*SCons m s*) (*SCons n t*) =
  (*if m* ≤ *n then SCons m* (*merge s* (*SCons n t*)) *else SCons n* (*merge* (*SCons m s*) *t*))
  by (*smt2 J.ctor_dtor J.dtor_ctor fst_conv snd_conv prod.collapse head_merge tail_merge*)

lemma *merge_uniform*: *merge s t* = *alg*ϱ*8* (*s*, *t*)
  unfolding *merge_def*
  apply (*rule fun_cong*[*OF sym*[*OF J.dtor_corec_unique*]])
  unfolding *alg*ϱ*8 o_def*[*symmetric*] *o_assoc*
  apply (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def fun_eq_iff J.dtor_ctor*
    ϱ*8_def Let_def convol_def eval8_*$\mathfrak{op}$*8 eval8_leaf8'*
    o_eq_dest[*OF Abs_*Σ*8_natural*] *alg*Λ*8_Inr alg*ϱ*8_def*)
  done

1.10 dup

definition dup **where**
dup s = dtor_corec_J (λs. (head s, Inl s)) s

lemma head_dup[simp]: head (dup s) = head s
unfolding dup_def J.dtor_corec
by (simp add: map_pre_J_def BNF_Comp.id_bnf_comp_def)

lemma tail_dup[simp]: tail (dup s) = s
unfolding dup_def J.dtor_corec
by (simp add: map_pre_J_def BNF_Comp.id_bnf_comp_def)

lemma dup_code[code]: dup (SCons m s) = SCons m (SCons m s)
by (metis J.ctor_dtor J.dtor_ctor fst_conv prod.collapse head_dup tail_dup)

1.11 inv

definition inv :: stream ⇒ stream **where**
inv s = dtor_corec_J (λs. (1 − head s, Inr (tail s))) s

lemma head_inv[simp]: head (inv s) = 1 − head s
unfolding inv_def J.dtor_corec
by (simp add: map_pre_J_def BNF_Comp.id_bnf_comp_def)

lemma tail_inv[simp]: tail (inv s) = inv (tail s)
apply (subst inv_def) **unfolding** J.dtor_corec
by (simp add: map_pre_J_def BNF_Comp.id_bnf_comp_def inv_def)

lemma inv_code[code]: inv (SCons m s) = SCons (1 − m) (inv s)
by (metis J.ctor_dtor J.dtor_ctor fst_conv snd_conv prod.collapse head_inv tail_inv)

lemma inv_uniform: inv s = algϱ7 (K7.I s)
unfolding inv_def
apply (rule fun_cong[OF sym[OF J.dtor_corec_unique]])
unfolding algϱ7 o_def[symmetric] o_assoc
apply (simp add: map_pre_J_def BNF_Comp.id_bnf_comp_def fun_eq_iff J.dtor_ctor
ϱ7_def Let_def convol_def eval7_𝔬𝔭7 eval7_leaf7′
o_eq_dest[OF Abs_Σ7_natural] algΛ7_Inr algϱ7_def)
done

1.12 thue

definition thue′ :: stream **where**
thue′ = corecUU7 (λ_. GUARD7 (1, INTERLEAVE7 (CONT7 (), INV7 (I (CONT7 ()))))) ()

lemma head_thue′[simp]: head thue′ = 1
unfolding thue′_def corecUU7
by (simp add: map_pre_J_def BNF_Comp.id_bnf_comp_def J.dtor_ctor eval7_leaf7′)

lemma tail_thue′[simp]: tail thue′ = interleave thue′ (inv thue′)
apply (subst thue′_def) **unfolding** corecUU7
by (simp add: map_pre_J_def BNF_Comp.id_bnf_comp_def J.dtor_ctor eval7_leaf7′ eval7_𝔬𝔭7
algΛ7_Inl algΛ7_Inr algΛ6_Inr o_eq_dest[OF Abs_Σ7_natural] o_eq_dest[OF Abs_Σ6_natural]
interleave_uniform inv_uniform thue′_def)

lemma thue′_code[code]: thue′ = SCons 1 (interleave thue′ (inv thue′))
by (metis J.ctor_dtor prod.collapse head_thue′ tail_thue′)

definition thue :: stream **where**

thue = *SCons 0 thue'*

1.13 times

definition *times* :: *nat* ⇒ *stream* ⇒ *stream* **where**
times n s = *dtor_corec_J* (*λs*. (*n* ∗ *head s*, *Inr* (*tail s*))) *s*

lemma *head_times*[*simp*]: *head* (*times n s*) = *n* ∗ *head s*
unfolding *times_def J.dtor_corec*
by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def*)

lemma *tail_times*[*simp*]: *tail* (*times n s*) = *times n* (*tail s*)
apply (*subst times_def*) **unfolding** *J.dtor_corec*
by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def times_def*)

lemma *times_code*[*code*]: *times n* (*SCons m s*) = *SCons* (*n* ∗ *m*) (*times n s*)
by (*metis J.ctor_dtor J.dtor_ctor fst_conv snd_conv prod.collapse head_times tail_times*)

lemma *times_uniform*: *times m s* = *alg*ϱ*9* (*m*, *s*)
unfolding *times_def*
apply (*rule fun_cong*[*OF sym*[*OF J.dtor_corec_unique*]])
unfolding *alg*ϱ*9 o_def*[*symmetric*] *o_assoc*
apply (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def fun_eq_iff J.dtor_ctor*
ϱ*9_def Let_def convol_def eval9_*𝔬𝔭*9 eval9_leaf9'*
o_eq_dest[*OF Abs_*Σ*9_natural*] *alg*Λ*9_Inr alg*ϱ*9_def*)
done

1.14 ham

definition *ham* :: *stream* **where**
ham = *corecUU9* (*λ_*. *GUARD9* (*1*, *MERGE9* (*TIMES9* (*2*, *CONT9* ()), *TIMES9* (*3*, *CONT9* ())))) ()

lemma *head_ham*[*simp*]: *head ham* = *1*
unfolding *ham_def corecUU9*
by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def J.dtor_ctor eval9_leaf9'*)

lemma *tail_ham*[*simp*]: *tail ham* = *merge* (*times 2 ham*) (*times 3 ham*)
apply (*subst ham_def*) **unfolding** *corecUU9*
by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def J.dtor_ctor eval9_leaf9' eval9_*𝔬𝔭*9*
*alg*Λ*9_Inl alg*Λ*9_Inr alg*Λ*8_Inr o_eq_dest*[*OF Abs_*Σ*9_natural*] *o_eq_dest*[*OF Abs_*Σ*8_natural*]
merge_uniform times_uniform ham_def)

lemma *ham_code*[*code*]: *ham* = *SCons 1* (*merge* (*times 2 ham*) (*times 3 ham*))
by (*metis J.ctor_dtor prod.collapse head_ham tail_ham*)

1.15 even

definition *even* :: *stream* ⇒ *stream* **where**
even s = *dtor_corec_J* (*λs*. (*head s*, *Inr* (*tail* (*tail s*)))) *s*

lemma *head_even*[*simp*]: *head* (*even s*) = *head s*
unfolding *even_def J.dtor_corec*
by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def*)

lemma *tail_even*[*simp*]: *tail* (*even s*) = *even* (*tail* (*tail s*))
apply (*subst even_def*) **unfolding** *J.dtor_corec*
by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def even_def*)

lemma *even_code*[*code*]: *even* (*SCons n* (*SCons m s*)) = *SCons n* (*even s*)

by (*metis J.ctor_dtor J.dtor_ctor fst_conv snd_conv prod.collapse head_even tail_even*)

1.16 odd

definition *odd* :: *stream* ⇒ *stream* **where**
odd s = *dtor_corec_J* (λ*s*. (*head* (*tail s*), *Inr* (*tail* (*tail s*)))) *s*

lemma *head_odd*[*simp*]: *head* (*odd s*) = *head* (*tail s*)
unfolding *odd_def J.dtor_corec*
by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def*)

lemma *tail_odd*[*simp*]: *tail* (*odd s*) = *odd* (*tail* (*tail s*))
apply (*subst odd_def*) **unfolding** *J.dtor_corec*
by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def odd_def*)

lemma *odd_code*[*code*]: *odd* (*SCons n* (*SCons m s*)) = *SCons m* (*odd s*)
by (*metis J.ctor_dtor J.dtor_ctor fst_conv snd_conv prod.collapse head_odd tail_odd*)

1.17 drop

definition *drop* :: *nat* ⇒ *nat* ⇒ *stream* ⇒ *stream* **where**
drop i l s = *dtor_corec_J* (λ(*i*, *l*, *s*).
case i of
Suc i ⇒ (*head s*, *Inr* (*i*, *l*, *tail s*))
| *0* ⇒ (*head* (*tail s*), *Inr* (*l* − *2*, *l*, *tail* (*tail s*)))) (*i*, *l*, *s*)

lemma *head_drop*[*simp*]: *head* (*drop i l s*) = (*case i of Suc* _ ⇒ *head s* | *0* ⇒ *head* (*tail s*))
unfolding *drop_def J.dtor_corec*
by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def split*: *nat.splits*)

lemma *tail_drop*[*simp*]:
tail (*drop i l s*) = (*case i of Suc i* ⇒ *drop i l* (*tail s*) | *0* ⇒ *drop* (*l* − *2*) *l* (*tail* (*tail s*)))
unfolding *drop_def J.dtor_corec*
by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def split*: *nat.splits*)

lemma *drop_code*[*code*]:
drop (*Suc i*) *l* (*SCons m s*) = *SCons m* (*drop i l s*)
drop 0 l (*SCons m* (*SCons n s*)) = *SCons n* (*drop* (*l* − *2*) *l s*)
by (*smt2 J.ctor_dtor J.dtor_ctor fst_conv snd_conv prod.collapse head_drop tail_drop nat.case*)+

1.18 diff

definition *diff* :: *stream* ⇒ *stream* **where**
diff s = *dtor_corec_J* (λ*s*. (*head* (*tail s*) − *head s*, *Inr* (*tail s*))) *s*

lemma *head_diff*[*simp*]: *head* (*diff s*) = *head* (*tail s*) − *head s*
unfolding *diff_def J.dtor_corec*
by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def*)

lemma *tail_diff*[*simp*]: *tail* (*diff s*) = *diff* (*tail s*)
apply (*subst diff_def*) **unfolding** *J.dtor_corec*
by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def diff_def*)

lemma *diff_code*[*code*]: *diff* (*SCons m* (*SCons n s*)) = *SCons* (*n* − *m*) (*diff* (*SCons n s*))
by (*metis J.ctor_dtor J.dtor_ctor fst_conv snd_conv prod.collapse head_diff tail_diff*)

1.19 Some Proofs

theorem *thue_code*[*code*]: *thue* = *SCons 0* (*interleave* (*inv thue*) (*tail thue*))
unfolding *thue_def J.ctor_inject Pair_eq*

by (*rule conjI*[*OF refl*], *coinduction rule*: *stream_coinduct0*) (*auto simp*: *J.dtor_ctor*)

theorem *plus_commute*: *plus s t* = *plus t s*
by (*coinduction arbitrary*: *s t rule*: *stream_coinduct*) *auto*

theorem *plus_assoc*: *plus* (*plus s t*) *u* = *plus s* (*plus t u*)
by (*coinduction arbitrary*: *s t u rule*: *stream_coinduct*) *auto*

theorem *plus_commute_assoc*: *plus s* (*plus t u*) = *plus t* (*plus s u*)
by (*metis plus_assoc plus_commute*)

theorem *even_plus*[*simp*]: *even* (*plus s t*) = *plus* (*even s*) (*even t*)
by (*coinduction arbitrary*: *s t rule*: *stream_coinduct*) *auto*

theorem *odd_alt*: *odd s* = *even* (*tail s*)
by (*coinduction arbitrary*: *s rule*: *stream_coinduct*) *auto*

theorem *even_alt*: *even s* = *SCons* (*head s*) (*odd* (*tail s*))
by (*coinduction arbitrary*: *s rule*: *stream_coinduct0*) (*auto simp*: *J.dtor_ctor odd_alt*)

theorem *sum_odd_fib*: *sum* (*odd fib*) = *even fib*
by (*coinduction rule*: *stream_coinduct1*)
(*auto simp*: *J.dtor_ctor plus_assoc plus_commute_assoc odd_alt*
intro: *genCngdd1_alg*ϱ*1*[*folded plus_uniform*])

2 Binary Tree Examples

2.1 one

definition *one* :: *btree* **where**
one = *dtor_corec_J* (λ_. (*1*, *Inr* (), *Inr* ())) ()

lemma *val_one*[*simp*]: *val one* = *1*
unfolding *one_def J.dtor_corec*
by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def*)

lemma *left_one*[*simp*]: *left one* = *one*
unfolding *one_def J.dtor_corec*
by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def*)

lemma *right_one*[*simp*]: *right one* = *one*
unfolding *one_def J.dtor_corec*
by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def*)

lemma *one_code*[*code*]:
one = *Node 1 one one*
by (*metis J.ctor_dtor prod.collapse val_one left_one right_one*)

2.2 plus

definition *plus* :: *btree* ⇒ *btree* ⇒ *btree* **where**
plus t u = *dtor_corec_J*
(λ(*t*, *u*). (*val t* + *val u*, *Inr* (*left t*, *left u*), *Inr* (*right t*, *right u*))) (*t*, *u*)

lemma *val_plus*[*simp*]: *val* (*plus t u*) = *val t* + *val u*
unfolding *plus_def J.dtor_corec*
by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def*)

lemma left_plus[simp]: left (plus t u) = plus (left t) (left u)
unfolding plus_def J.dtor_corec
by (simp add: map_pre_J_def BNF_Comp.id_bnf_comp_def)

lemma right_plus[simp]: right (plus t u) = plus (right t) (right u)
unfolding plus_def J.dtor_corec
by (simp add: map_pre_J_def BNF_Comp.id_bnf_comp_def)

lemma plus_code[code]:
plus (Node x1 l1 r1) (Node x2 l2 r2) = Node (x1 + x2) (plus l1 l2) (plus r1 r2)
by (smt2 J.ctor_dtor J.dtor_ctor fst_conv snd_conv prod.collapse val_plus left_plus right_plus)

lemma plus_uniform: plus s t = algϱ1 (s, t)
unfolding plus_def
apply (rule fun_cong[OF sym[OF J.dtor_corec_unique]])
unfolding algϱ1 o_def[symmetric] o_assoc
apply (simp add: map_pre_J_def BNF_Comp.id_bnf_comp_def fun_eq_iff J.dtor_ctor
ϱ1_def Let_def convol_def eval1_𝔬𝔭1 eval1_leaf1′
o_eq_dest[OF Abs_Σ1_natural] algΛ1_Inr algϱ1_def)
done

2.3 divide

definition divide :: btree ⇒ btree ⇒ btree **where**
divide t u = dtor_corec_J
(λ(t, u). (val t / val u, Inr (left t, left u), Inr (right t, right u))) (t, u)

lemma val_divide[simp]: val (divide t u) = val t / val u
unfolding divide_def J.dtor_corec
by (simp add: map_pre_J_def BNF_Comp.id_bnf_comp_def)

lemma left_divide[simp]: left (divide t u) = divide (left t) (left u)
unfolding divide_def J.dtor_corec
by (simp add: map_pre_J_def BNF_Comp.id_bnf_comp_def)

lemma right_divide[simp]: right (divide t u) = divide (right t) (right u)
unfolding divide_def J.dtor_corec
by (simp add: map_pre_J_def BNF_Comp.id_bnf_comp_def)

lemma divide_code[code]:
divide (Node x1 l1 r1) (Node x2 l2 r2) = Node (x1 / x2) (divide l1 l2) (divide r1 r2)
by (smt2 J.ctor_dtor J.dtor_ctor fst_conv snd_conv prod.collapse val_divide left_divide right_divide)

lemma divide_uniform: divide s t = algϱ2 (s, t)
unfolding divide_def
apply (rule fun_cong[OF sym[OF J.dtor_corec_unique]])
unfolding algϱ2 o_def[symmetric] o_assoc
apply (simp add: map_pre_J_def BNF_Comp.id_bnf_comp_def fun_eq_iff J.dtor_ctor
ϱ2_def Let_def convol_def eval2_𝔬𝔭2 eval2_leaf2′
o_eq_dest[OF Abs_Σ2_natural] algΛ2_Inr algϱ2_def)
done

2.4 bird

definition bird **where**
bird = corecUU2 (λ_. GUARD2 (1,
DIV2 (END2 one, PLS2 (CONT2 (), END2 one)),
PLS2 (DIV2 (END2 one, CONT2 ()), END2 one))) ()

lemma *val_bird*[*simp*]: *val bird* = *1*
  unfolding *bird_def corecUU2*
  by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def J.dtor_ctor eval2_leaf2'*)

lemma *left_bird*[*simp*]: *left bird* = *divide one* (*plus bird one*)
  apply (*subst bird_def*) **unfolding** *corecUU2*
  by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def J.dtor_ctor eval2_leaf2' eval2_op2*
    algΛ2_Inl algΛ2_Inr algΛ1_Inr o_eq_dest[*OF Abs_Σ2_natural*] *o_eq_dest*[*OF Abs_Σ1_natural*]
    plus_uniform divide_uniform bird_def)

lemma *right_bird*[*simp*]: *right bird* = *plus* (*divide one bird*) *one*
  apply (*subst bird_def*) **unfolding** *corecUU2*
  by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def J.dtor_ctor eval2_leaf2' eval2_op2*
    algΛ2_Inl algΛ2_Inr algΛ1_Inr o_eq_dest[*OF Abs_Σ2_natural*] *o_eq_dest*[*OF Abs_Σ1_natural*]
    plus_uniform divide_uniform bird_def)

lemma *bird_code*[*code*]: *bird* = *Node 1* (*divide one* (*plus bird one*)) (*plus* (*divide one bird*) *one*)
  by (*metis J.ctor_dtor prod.collapse val_bird left_bird right_bird*)

2.5 mirror

definition *mirror* :: *btree* ⇒ *btree* **where**
  mirror t = *dtor_corec_J* (*λt.* (*val t*, *Inr* (*right t*), *Inr* (*left t*))) *t*

lemma *val_mirror*[*simp*]: *val* (*mirror t*) = *val t*
  unfolding *mirror_def J.dtor_corec*
  by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def*)

lemma *left_mirror*[*simp*]: *left* (*mirror t*) = *mirror* (*right t*)
  unfolding *mirror_def J.dtor_corec*
  by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def*)

lemma *right_mirror*[*simp*]: *right* (*mirror t*) = *mirror* (*left t*)
  unfolding *mirror_def J.dtor_corec*
  by (*simp add*: *map_pre_J_def BNF_Comp.id_bnf_comp_def*)

lemma *mirror_code*[*code*]:
  mirror (*Node x l r*) = *Node x* (*mirror r*) (*mirror l*)
  by (*metis J.ctor_dtor J.dtor_ctor fst_conv snd_conv prod.collapse val_mirror left_mirror right_mirror*)

2.6 Some Proofs

theorem *mirror_one*[*simp*]: *mirror one* = *one*
  by (*coinduction rule*: *btree_coinduct*) *auto*

theorem *mirror_plus*[*simp*]: *mirror* (*plus r s*) = *plus* (*mirror r*) (*mirror s*)
  by (*coinduction arbitrary*: *r s rule*: *btree_coinduct*) *auto*

theorem *mirror_divide*[*simp*]: *mirror* (*divide r s*) = *divide* (*mirror r*) (*mirror s*)
  by (*coinduction arbitrary*: *r s rule*: *btree_coinduct*) *auto*

theorem *divide_divide_one*[*simp*]: *divide one* (*divide one r*) = *r*
  by (*coinduction arbitrary*: *r rule*: *btree_coinduct*) *auto*

theorem *mirror_bird*: *mirror bird* = *divide one bird*
  by (*coinduction rule*: *btree_coinduct2*)
    (*auto intro*!: *genCngdd2_algϱ1*[*folded plus_uniform*] *intro*: *genCngdd2_algϱ2*[*folded divide_uniform*]
      genCngdd2_trans[*OF genCngdd2_algϱ2*[*folded divide_uniform*], *of _ _ _ _ divide one bird*])