Operations on Bounded Natural Functors

Jasmin Christian Blanchette   Andrei Popescu   Dmitriy Traytel

March 17, 2025

Abstract

This entry formalizes the closure property of bounded natural functors (BNFs) under seven operations. These operations and the corresponding proofs constitute the core of Isabelle's (co)datatype package. To be close to the implemented tactics, the proofs are deliberately formulated as detailed apply scripts. The (co)datatypes together with (co)induction principles and (co)recursors are byproducts of the fixpoint operations LFP and GFP. Composition of BNFs is subdivided into four simpler operations: Compose, Kill, Lift, and Permute. The N2M operation provides mutual (co)induction principles and (co)recursors for nested (co)datatypes.

Contents

1 Least Fixpoint (a.k.a. Datatype)

unbundle *cardinal_syntax*

⟨*ML*⟩
notation *BNF_Def.convol* (‹<_ , _>›)

'b1 = ('a, 'b1, 'b2) F1
'b2 = ('a, 'b1, 'b2) F2

To build a witness scenario, let us assume

('a, 'b1, 'b2) F1 = 'a * 'b1 + 'a * 'b2
('a, 'b1, 'b2) F2 = unit + 'b1 * 'b2

declare [[*bnf_internals*]]
bnf-axiomatization (*F1set1*: *'a*, *F1set2*: *'b1*, *F1set3*: *'b2*) *F1*
[*wits*: *'a* ⇒ *'b1* ⇒ (*'a*, *'b1*, *'b2*) *F1* *'a* ⇒ *'b2* ⇒ (*'a*, *'b1*, *'b2*) *F1*]
for *map*: *F1map* *rel*: *F1rel*
bnf-axiomatization (*F2set1*: *'a*, *F2set2*: *'b1*, *F2set3*: *'b2*) *F2*
[*wits*: (*'a*, *'b1*, *'b2*) *F2*]
for *map*: *F2map* *rel*: *F2rel*

abbreviation *F1in* :: *'a1 set* ⇒ *'a2 set* ⇒ *'a3 set* ⇒ ((*'a1*, *'a2*, *'a3*) *F1*) *set* **where**
F1in A1 A2 A3 ≡ {*x*. *F1set1 x* ⊆ *A1* ∧ *F1set2 x* ⊆ *A2* ∧ *F1set3 x* ⊆ *A3*}
abbreviation *F2in* :: *'a1 set* ⇒ *'a2 set* ⇒ *'a3 set* ⇒ ((*'a1*, *'a2*, *'a3*) *F2*) *set* **where**
F2in A1 A2 A3 ≡ {*x*. *F2set1 x* ⊆ *A1* ∧ *F2set2 x* ⊆ *A2* ∧ *F2set3 x* ⊆ *A3*}

lemma *F1map_comp_id*: *F1map g1 g2 g3* (*F1map id f2 f3 x*) = *F1map g1* (*g2 o f2*) (*g3 o f3*) *x*
⟨*proof*⟩

lemmas *F1in_mono23* = *F1.in_mono*[*OF subset_refl*]

lemma *F1map_congL*: ⟦∀ *a* ∈ *F1set2 x*. *f a* = *a*; ∀ *a* ∈ *F1set3 x*. *g a* = *a*⟧ ⟹
F1map id f g x = *x*
⟨*proof*⟩

lemma *F2map_comp_id*: *F2map g1 g2 g3* (*F2map id f2 f3 x*) = *F2map g1* (*g2 o f2*) (*g3 o f3*) *x*
⟨*proof*⟩

lemmas *F2in_mono23* = *F2.in_mono*[*OF subset_refl*]

lemma *F2map_congL*: ⟦∀ *a* ∈ *F2set2 x*. *f a* = *a*; ∀ *a* ∈ *F2set3 x*. *g a* = *a*⟧ ⟹
F2map id f g x = *x*
⟨*proof*⟩

1.1 Algebra

definition *alg* **where**
alg B1 B2 s1 s2 =
((∀ *x* ∈ *F1in* (*UNIV* :: *'a set*) *B1 B2*. *s1 x* ∈ *B1*) ∧ (∀ *y* ∈ *F2in* (*UNIV* :: *'a set*) *B1 B2*. *s2 y* ∈ *B2*))

lemma *alg_F1set*: ⟦*alg B1 B2 s1 s2*; *F1set2 x* ⊆ *B1*; *F1set3 x* ⊆ *B2*⟧ ⟹ *s1 x* ∈ *B1*
⟨*proof*⟩

lemma *alg_F2set*: $[\![alg\ B1\ B2\ s1\ s2;\ F2set2\ x \subseteq B1;\ F2set3\ x \subseteq B2]\!] \Longrightarrow s2\ x \in B2$
⟨*proof*⟩

lemma *alg_not_empty*:
$alg\ B1\ B2\ s1\ s2 \Longrightarrow B1 \neq \{\} \wedge B2 \neq \{\}$
⟨*proof*⟩

1.2 Morphism

definition *mor* **where**
$mor\ B1\ B2\ s1\ s2\ B1'\ B2'\ s1'\ s2'\ f\ g =$
$(((\forall a \in B1.\ f\ a \in B1') \wedge (\forall a \in B2.\ g\ a \in B2')) \wedge$
$((\forall z \in F1in\ (UNIV :: 'a\ set)\ B1\ B2.\ f\ (s1\ z) = s1'\ (F1map\ id\ f\ g\ z)) \wedge$
$(\forall z \in F2in\ (UNIV :: 'a\ set)\ B1\ B2.\ g\ (s2\ z) = s2'\ (F2map\ id\ f\ g\ z))))$

lemma *morE1*: $[\![mor\ B1\ B2\ s1\ s2\ B1'\ B2'\ s1'\ s2'\ f\ g;\ z \in F1in\ UNIV\ B1\ B2]\!]$
$\Longrightarrow f\ (s1\ z) = s1'\ (F1map\ id\ f\ g\ z)$
⟨*proof*⟩

lemma *morE2*: $[\![mor\ B1\ B2\ s1\ s2\ B1'\ B2'\ s1'\ s2'\ f\ g;\ z \in F2in\ UNIV\ B1\ B2]\!]$
$\Longrightarrow g\ (s2\ z) = s2'\ (F2map\ id\ f\ g\ z)$
⟨*proof*⟩

lemma *mor_incl*: $[\![B1 \subseteq B1';\ B2 \subseteq B2']\!] \Longrightarrow mor\ B1\ B2\ s1\ s2\ B1'\ B2'\ s1\ s2\ id\ id$
⟨*proof*⟩

lemma *mor_comp*:
$[\![mor\ B1\ B2\ s1\ s2\ B1'\ B2'\ s1'\ s2'\ f\ g;$
$mor\ B1'\ B2'\ s1'\ s2'\ B1''\ B2''\ s1''\ s2''\ f'\ g']\!] \Longrightarrow$
$mor\ B1\ B2\ s1\ s2\ B1''\ B2''\ s1''\ s2''\ (f'\ o\ f)\ (g'\ o\ g)$
⟨*proof*⟩

lemma *mor_cong*: $[\![\ f' = f;\ g' = g;\ mor\ B1\ B2\ s1\ s2\ B1'\ B2'\ s1'\ s2'\ f\ g]\!] \Longrightarrow$
$mor\ B1\ B2\ s1\ s2\ B1'\ B2'\ s1'\ s2'\ f'\ g'$
⟨*proof*⟩

lemma *mor_str*:
$mor\ UNIV\ UNIV\ (F1map\ id\ s1\ s2)\ (F2map\ id\ s1\ s2)\ UNIV\ UNIV\ s1\ s2\ s1\ s2$
⟨*proof*⟩

1.3 Bounds

type-synonym $bd\_type\_F1' = bd\_type\_F1 + (bd\_type\_F1,\ bd\_type\_F1,\ bd\_type\_F1)\ F1$
type-synonym $bd\_type\_F2' = bd\_type\_F2 + (bd\_type\_F2,\ bd\_type\_F2,\ bd\_type\_F2)\ F2$
type-synonym $SucFbd\_type = ((bd\_type\_F1' + bd\_type\_F2')\ set)$
type-synonym $'a1\ ASucFbd\_type = (SucFbd\_type \Rightarrow ('a1 + bool))$

abbreviation $F1bd' \equiv bd\_F1 +c\ |UNIV :: (bd\_type\_F1,\ bd\_type\_F1,\ bd\_type\_F1)\ F1\ set|$
lemma *F1set1_bd_incr*: $\bigwedge x.\ |F1set1\ x| <o\ F1bd'$
⟨*proof*⟩
lemma *F1set2_bd_incr*: $\bigwedge x.\ |F1set2\ x| <o\ F1bd'$
⟨*proof*⟩
lemma *F1set3_bd_incr*: $\bigwedge x.\ |F1set3\ x| <o\ F1bd'$
⟨*proof*⟩

lemmas $F1bd'\_Card\_order = Card\_order\_csum$
lemmas $F1bd'\_Cinfinite = Cinfinite\_csum1[OF\ F1.bd\_Cinfinite]$
lemmas $F1bd'\_Cnotzero = Cinfinite\_Cnotzero[OF\ F1bd'\_Cinfinite]$
lemmas $F1bd'\_card\_order = card\_order\_csum[OF\ F1.bd\_card\_order\ card\_of\_card\_order\_on]$

abbreviation $F2bd' \equiv bd\_F2 +c\ |UNIV :: (bd\_type\_F2,\ bd\_type\_F2,\ bd\_type\_F2)\ F2\ set|$
lemma *F2set1_bd_incr*: $\bigwedge x.\ |F2set1\ x| <o\ F2bd'$
⟨*proof*⟩

lemma $F2set2\_bd\_incr$: $\bigwedge x.\ |F2set2\ x| <o\ F2bd'$
⟨proof⟩
lemma $F2set3\_bd\_incr$: $\bigwedge x.\ |F2set3\ x| <o\ F2bd'$
⟨proof⟩

lemmas $F2bd'\_Card\_order = Card\_order\_csum$
lemmas $F2bd'\_Cinfinite = Cinfinite\_csum1[OF\ F2.bd\_Cinfinite]$
lemmas $F2bd'\_Cnotzero = Cinfinite\_Cnotzero[OF\ F2bd'\_Cinfinite]$
lemmas $F2bd'\_card\_order = card\_order\_csum[OF\ F2.bd\_card\_order\ card\_of\_card\_order\_on]$

abbreviation $SucFbd$ **where** $SucFbd \equiv cardSuc\ (F1bd' +c\ F2bd')$
abbreviation $ASucFbd$ **where** $ASucFbd \equiv (\ |UNIV| +c\ ctwo\ )\ \widehat{}c\ SucFbd$

lemma $F1set1\_bd$: $|F1set1\ x| <o\ bd\_F1 +c\ bd\_F2$
⟨proof⟩

lemma $F1set2\_bd$: $|F1set2\ x| <o\ bd\_F1 +c\ bd\_F2$
⟨proof⟩

lemma $F1set3\_bd$: $|F1set3\ x| <o\ bd\_F1 +c\ bd\_F2$
⟨proof⟩

lemma $F2set1\_bd$: $|F2set1\ x| <o\ bd\_F1 +c\ bd\_F2$
⟨proof⟩

lemma $F2set2\_bd$: $|F2set2\ x| <o\ bd\_F1 +c\ bd\_F2$
⟨proof⟩

lemma $F2set3\_bd$: $|F2set3\ x| <o\ bd\_F1 +c\ bd\_F2$
⟨proof⟩

lemmas $SucFbd\_Card\_order = cardSuc\_Card\_order[OF\ Card\_order\_csum]$
lemmas $SucFbd\_Cinfinite = Cinfinite\_cardSuc[OF\ Cinfinite\_csum1[OF\ F1bd'\_Cinfinite]]$
lemmas $SucFbd\_Cnotzero = Cinfinite\_Cnotzero[OF\ SucFbd\_Cinfinite]$
lemmas $worel\_SucFbd = Card\_order\_wo\_rel[OF\ SucFbd\_Card\_order]$
lemmas $ASucFbd\_Cinfinite = Cinfinite\_cexp[OF\ ordLeq\_csum2[OF\ Card\_order\_ctwo]\ SucFbd\_Cinfinite]$

1.4 Minimal Algebras

abbreviation min_G1 **where**
$min\_G1\ As1\_As2\ i \equiv (\bigcup j \in underS\ SucFbd\ i.\ fst\ (As1\_As2\ j))$

abbreviation min_G2 **where**
$min\_G2\ As1\_As2\ i \equiv (\bigcup j \in underS\ SucFbd\ i.\ snd\ (As1\_As2\ j))$

abbreviation min_H **where**
$min\_H\ s1\ s2\ As1\_As2\ i \equiv$
$(min\_G1\ As1\_As2\ i \cup s1\ `\ (F1in\ (UNIV :: 'a\ set)\ (min\_G1\ As1\_As2\ i)\ (min\_G2\ As1\_As2\ i)),$
$min\_G2\ As1\_As2\ i \cup s2\ `\ (F2in\ (UNIV :: 'a\ set)\ (min\_G1\ As1\_As2\ i)\ (min\_G2\ As1\_As2\ i)))$

abbreviation min_algs **where**
$min\_algs\ s1\ s2 \equiv wo\_rel.worec\ SucFbd\ (min\_H\ s1\ s2)$

definition min_alg1 **where**
$min\_alg1\ s1\ s2 = (\bigcup i \in Field\ SucFbd.\ fst\ (min\_algs\ s1\ s2\ i))$

definition min_alg2 **where**
$min\_alg2\ s1\ s2 = (\bigcup i \in Field\ SucFbd.\ snd\ (min\_algs\ s1\ s2\ i))$

lemma min_algs:
$i \in Field\ SucFbd \Longrightarrow min\_algs\ s1\ s2\ i = min\_H\ s1\ s2\ (min\_algs\ s1\ s2)\ i$
⟨proof⟩

corollary min_algs1: $i \in Field\ SucFbd \Longrightarrow fst\ (min\_algs\ s1\ s2\ i) =$

$min\_G1\ (min\_algs\ s1\ s2)\ i \cup$
$s1\ `\ (F1in\ UNIV\ (min\_G1\ (min\_algs\ s1\ s2)\ i)\ (min\_G2\ (min\_algs\ s1\ s2)\ i))$
⟨*proof*⟩

corollary min_algs2: $i \in Field\ SucFbd \Longrightarrow snd\ (min\_algs\ s1\ s2\ i) =$
$min\_G2\ (min\_algs\ s1\ s2)\ i \cup$
$s2\ `\ (F2in\ UNIV\ (min\_G1\ (min\_algs\ s1\ s2)\ i)\ (min\_G2\ (min\_algs\ s1\ s2)\ i))$
⟨*proof*⟩

lemma min_algs_mono1: $relChain\ SucFbd\ (\%i.\ fst\ (min\_algs\ s1\ s2\ i))$
⟨*proof*⟩

lemma min_algs_mono2: $relChain\ SucFbd\ (\%i.\ snd\ (min\_algs\ s1\ s2\ i))$
⟨*proof*⟩

lemma $SucFbd\_limit$: $[\![x1 \in Field\ SucFbd\ \&\ x2 \in Field\ SucFbd]\!]$
$\Longrightarrow \exists\, y \in Field\ SucFbd.\ (x1 \neq y \wedge (x1,\ y) \in SucFbd) \wedge (x2 \neq y \wedge (x2,\ y) \in SucFbd)$
⟨*proof*⟩

lemma alg_min_alg: $alg\ (min\_alg1\ s1\ s2)\ (min\_alg2\ s1\ s2)\ s1\ s2$
⟨*proof*⟩

lemmas $SucFbd\_ASucFbd = ordLess\_ordLeq\_trans[OF$
$ordLess\_ctwo\_cexp$
$cexp\_mono1[OF\ ordLeq\_csum2[OF\ Card\_order\_ctwo]],$
$OF\ SucFbd\_Card\_order\ SucFbd\_Card\_order]$

lemma $card\_of\_min\_algs$:
fixes $s1 :: ('a,\ 'b,\ 'c)\ F1 \Rightarrow 'b$ **and** $s2 :: ('a,\ 'b,\ 'c)\ F2 \Rightarrow 'c$
shows $i \in Field\ SucFbd \longrightarrow$
$(\ |fst\ (min\_algs\ s1\ s2\ i)| \le o\ (ASucFbd :: 'a\ ASucFbd\_type\ rel) \wedge |snd\ (min\_algs\ s1\ s2\ i)| \le o\ (ASucFbd :: 'a\ ASucFbd\_type\ rel)\ )$
⟨*proof*⟩

lemma $card\_of\_min\_alg1$:
fixes $s1 :: ('a,\ 'b,\ 'c)\ F1 \Rightarrow 'b$ **and** $s2 :: ('a,\ 'b,\ 'c)\ F2 \Rightarrow 'c$
shows $|min\_alg1\ s1\ s2| \le o\ (ASucFbd :: 'a\ ASucFbd\_type\ rel)$
⟨*proof*⟩

lemma $card\_of\_min\_alg2$:
fixes $s1 :: ('a,\ 'b,\ 'c)\ F1 \Rightarrow 'b$ **and** $s2 :: ('a,\ 'b,\ 'c)\ F2 \Rightarrow 'c$
shows $|min\_alg2\ s1\ s2| \le o\ (ASucFbd :: 'a\ ASucFbd\_type\ rel)$
⟨*proof*⟩

lemma $least\_min\_algs$: $alg\ B1\ B2\ s1\ s2 \Longrightarrow$
$i \in Field\ SucFbd \longrightarrow$
$fst\ (min\_algs\ s1\ s2\ i) \subseteq B1 \wedge snd\ (min\_algs\ s1\ s2\ i) \subseteq B2$
⟨*proof*⟩

lemma $least\_min\_alg1$: $alg\ B1\ B2\ s1\ s2 \Longrightarrow min\_alg1\ s1\ s2 \subseteq B1$
⟨*proof*⟩

lemma $least\_min\_alg2$: $alg\ B1\ B2\ s1\ s2 \Longrightarrow min\_alg2\ s1\ s2 \subseteq B2$
⟨*proof*⟩

lemma $mor\_incl\_min\_alg$:
$alg\ B1\ B2\ s1\ s2 \Longrightarrow$
$mor\ (min\_alg1\ s1\ s2)\ (min\_alg2\ s1\ s2)\ s1\ s2\ B1\ B2\ s1\ s2\ id\ id$
⟨*proof*⟩

1.5 Initiality

The following “happens" to be the type (for our particular construction) of the initial algebra carrier:

type-synonym $'a1\ F1init\_type = ('a1,\ 'a1\ ASucFbd\_type,\ 'a1\ ASucFbd\_type)\ F1 \Rightarrow 'a1\ ASucFbd\_type$
type-synonym $'a1\ F2init\_type = ('a1,\ 'a1\ ASucFbd\_type,\ 'a1\ ASucFbd\_type)\ F2 \Rightarrow 'a1\ ASucFbd\_type$

typedef $'a1\ IIT =$
$UNIV ::$
$(('a1\ ASucFbd\_type\ set \times 'a1\ ASucFbd\_type\ set) \times ('a1\ F1init\_type \times 'a1\ F2init\_type))\ set$
$\langle proof \rangle$

1.6 Initial Algebras

abbreviation $II :: 'a1\ IIT\ set$ **where**
$II \equiv \{Abs\_IIT\ ((B1,\ B2),\ (s1,\ s2))\ |B1\ B2\ s1\ s2.\ alg\ B1\ B2\ s1\ s2\}$
definition str_init1 **where**
$str\_init1\ (dummy :: 'a1)$
$(y::('a1,\ 'a1\ IIT \Rightarrow 'a1\ ASucFbd\_type,\ 'a1\ IIT \Rightarrow 'a1\ ASucFbd\_type)\ F1)$
$(i :: 'a1\ IIT) =$
$fst\ (snd\ (Rep\_IIT\ i))$
$(F1map\ id\ (\lambda f :: 'a1\ IIT \Rightarrow 'a1\ ASucFbd\_type.\ f\ i)\ (\lambda f.\ f\ i)\ y)$
definition str_init2 **where**
$str\_init2\ (dummy :: 'a1)\ y\ (i :: 'a1\ IIT) =$
$snd\ (snd\ (Rep\_IIT\ i))\ (F2map\ id\ (\lambda f.\ f\ i)\ (\lambda f.\ f\ i)\ y)$
abbreviation car_init1 **where**
$car\_init1\ dummy \equiv min\_alg1\ (str\_init1\ dummy)\ (str\_init2\ dummy)$
abbreviation car_init2 **where**
$car\_init2\ dummy \equiv min\_alg2\ (str\_init1\ dummy)\ (str\_init2\ dummy)$

lemma alg_select:
$\forall i \in II.\ alg\ (fst\ (fst\ (Rep\_IIT\ i)))\ (snd\ (fst\ (Rep\_IIT\ i)))$
$(fst\ (snd\ (Rep\_IIT\ i)))\ (snd\ (snd\ (Rep\_IIT\ i)))$
$\langle proof \rangle$

lemma mor_select:
$[\![i \in II;$
$mor\ (fst\ (fst\ (Rep\_IIT\ i)))\ (snd\ (fst\ (Rep\_IIT\ i)))$
$(fst\ (snd\ (Rep\_IIT\ i)))\ (snd\ (snd\ (Rep\_IIT\ i)))\ UNIV\ UNIV\ s1'\ s2'\ f\ g]\!] \Longrightarrow$
$mor\ (car\_init1\ dummy)\ (car\_init2\ dummy)\ (str\_init1\ dummy)\ (str\_init2\ dummy)\ UNIV\ UNIV\ s1'\ s2'\ (f \circ (\lambda h.\ h\ i))\ (g \circ (\lambda h.\ h\ i))$
$\langle proof \rangle$

lemma $init\_unique\_mor$:
$[\![a1 \in car\_init1\ dummy;\ a2 \in car\_init2\ dummy;$
$mor\ (car\_init1\ dummy)\ (car\_init2\ dummy)\ (str\_init1\ dummy)\ (str\_init2\ dummy)\ B1\ B2\ s1\ s2\ f1\ f2;$
$mor\ (car\_init1\ dummy)\ (car\_init2\ dummy)\ (str\_init1\ dummy)\ (str\_init2\ dummy)\ B1\ B2\ s1\ s2\ g1\ g2]\!] \Longrightarrow$
$f1\ a1 = g1\ a1 \wedge f2\ a2 = g2\ a2$
$\langle proof \rangle$

abbreviation $closed$ **where**
$closed\ dummy\ phi1\ phi2 \equiv ((\forall x \in F1in\ UNIV\ (car\_init1\ dummy)\ (car\_init2\ dummy).$
$(\forall z \in F1set2\ x.\ phi1\ z) \wedge (\forall z \in F1set3\ x.\ phi2\ z) \longrightarrow phi1\ (str\_init1\ dummy\ x)) \wedge$
$(\forall x \in F2in\ UNIV\ (car\_init1\ dummy)\ (car\_init2\ dummy).$
$(\forall z \in F2set2\ x.\ phi1\ z) \wedge (\forall z \in F2set3\ x.\ phi2\ z) \longrightarrow phi2\ (str\_init2\ dummy\ x)))$

lemma $init\_induct$: $closed\ dummy\ phi1\ phi2 \Longrightarrow$
$(\forall x \in car\_init1\ dummy.\ phi1\ x) \wedge (\forall x \in car\_init2\ dummy.\ phi2\ x)$
$\langle proof \rangle$

1.7 The datatype

typedef (**overloaded**) $'a1\ IF1 = car\_init1\ (undefined :: 'a1)$
$\langle proof \rangle$

typedef (**overloaded**) $'a1\ IF2 = car\_init2\ (undefined :: 'a1)$
$\langle proof \rangle$

definition *ctor1* **where** *ctor1 = Abs_IF1 o str_init1 undefined o F1map id Rep_IF1 Rep_IF2*
definition *ctor2* **where** *ctor2 = Abs_IF2 o str_init2 undefined o F2map id Rep_IF1 Rep_IF2*

lemma *mor_Rep_IF*:
mor (UNIV :: 'a IF1 set) (UNIV :: 'a IF2 set) ctor1 ctor2
(car_init1 undefined) (car_init2 undefined) (str_init1 undefined) (str_init2 undefined) Rep_IF1 Rep_IF2
⟨*proof*⟩

lemma *mor_Abs_IF*:
mor (car_init1 undefined) (car_init2 undefined)
(str_init1 undefined) (str_init2 undefined) UNIV UNIV ctor1 ctor2 Abs_IF1 Abs_IF2
⟨*proof*⟩

lemma *copy*:
$[\![$*alg B1 B2 s1 s2*; *bij_betw f B1' B1*; *bij_betw g B2' B2*$]\!] \Longrightarrow$
$\exists$ *f' g'*. *alg B1' B2' f' g'* $\wedge$ *mor B1' B2' f' g' B1 B2 s1 s2 f g*
⟨*proof*⟩

lemma *init_ex_mor*:
$\exists$ *f g*. *mor UNIV UNIV ctor1 ctor2 UNIV UNIV s1 s2 f g*
⟨*proof*⟩

Iteration

abbreviation *fold* **where**
fold s1 s2 $\equiv$ *(SOME f. mor UNIV UNIV ctor1 ctor2 UNIV UNIV s1 s2 (fst f) (snd f))*

definition *fold1* **where** *fold1 s1 s2 = fst (fold s1 s2)*
definition *fold2* **where** *fold2 s1 s2 = snd (fold s1 s2)*

lemma *mor_fold*:
mor UNIV UNIV ctor1 ctor2 UNIV UNIV s1 s2 (fold1 s1 s2) (fold2 s1 s2)
⟨*proof*⟩

⟨*ML*⟩

theorem *fold1*:
(fold1 s1 s2) (ctor1 x) = s1 (F1map id (fold1 s1 s2) (fold2 s1 s2) x)
⟨*proof*⟩

theorem *fold2*:
(fold2 s1 s2) (ctor2 x) = s2 (F2map id (fold1 s1 s2) (fold2 s1 s2) x)
⟨*proof*⟩

lemma *mor_UNIV*: *mor UNIV UNIV s1 s2 UNIV UNIV s1' s2' f g* $\longleftrightarrow$
f o s1 = s1' o F1map id f g $\wedge$ *g o s2 = s2' o F2map id f g*
⟨*proof*⟩

lemma *fold_unique_mor*: *mor UNIV UNIV ctor1 ctor2 UNIV UNIV s1 s2 f g* $\Longrightarrow$
f = fold1 s1 s2 $\wedge$ *g = fold2 s1 s2*
⟨*proof*⟩

lemmas *fold_unique = fold_unique_mor[OF iffD2[OF mor_UNIV], OF conjI]*

lemmas *fold1_ctor = sym[OF conjunct1[OF fold_unique_mor[OF mor_incl[OF subset_UNIV subset_UNIV]]]]*
lemmas *fold2_ctor = sym[OF conjunct2[OF fold_unique_mor[OF mor_incl[OF subset_UNIV subset_UNIV]]]]*

Case distinction

lemmas *ctor1_o_fold1 =*
trans[OF conjunct1[OF fold_unique_mor[OF mor_comp[OF mor_fold mor_str]]] fold1_ctor]
lemmas *ctor2_o_fold2 =*
trans[OF conjunct2[OF fold_unique_mor[OF mor_comp[OF mor_fold mor_str]]] fold2_ctor]

definition *dtor1* = *fold1* (*F1map id ctor1 ctor2*) (*F2map id ctor1 ctor2*)
definition *dtor2* = *fold2* (*F1map id ctor1 ctor2*) (*F2map id ctor1 ctor2*)

⟨*ML*⟩

lemma *ctor1_o_dtor1*: *ctor1 o dtor1* = *id*
⟨*proof*⟩

lemma *ctor2_o_dtor2*: *ctor2 o dtor2* = *id*
⟨*proof*⟩
lemma *dtor1_o_ctor1*: *dtor1 o ctor1* = *id*
⟨*proof*⟩

lemma *dtor2_o_ctor2*: *dtor2 o ctor2* = *id*
⟨*proof*⟩

lemmas *dtor1_ctor1* = *pointfree_idE*[*OF dtor1_o_ctor1*]
lemmas *dtor2_ctor2* = *pointfree_idE*[*OF dtor2_o_ctor2*]
lemmas *ctor1_dtor1* = *pointfree_idE*[*OF ctor1_o_dtor1*]
lemmas *ctor2_dtor2* = *pointfree_idE*[*OF ctor2_o_dtor2*]

lemmas *bij_dtor1* = *o_bij*[*OF ctor1_o_dtor1 dtor1_o_ctor1*]
lemmas *inj_dtor1* = *bij_is_inj*[*OF bij_dtor1*]
lemmas *surj_dtor1* = *bij_is_surj*[*OF bij_dtor1*]
lemmas *dtor1_nchotomy* = *surjD*[*OF surj_dtor1*]
lemmas *dtor1_diff* = *inj_eq*[*OF inj_dtor1*]
lemmas *dtor1_cases* = *exE*[*OF dtor1_nchotomy*]
lemmas *bij_dtor2* = *o_bij*[*OF ctor2_o_dtor2 dtor2_o_ctor2*]
lemmas *inj_dtor2* = *bij_is_inj*[*OF bij_dtor2*]
lemmas *surj_dtor2* = *bij_is_surj*[*OF bij_dtor2*]
lemmas *dtor2_nchotomy* = *surjD*[*OF surj_dtor2*]
lemmas *dtor2_diff* = *inj_eq*[*OF inj_dtor2*]
lemmas *dtor2_cases* = *exE*[*OF dtor2_nchotomy*]

lemmas *bij_ctor1* = *o_bij*[*OF dtor1_o_ctor1 ctor1_o_dtor1*]
lemmas *inj_ctor1* = *bij_is_inj*[*OF bij_ctor1*]
lemmas *surj_ctor1* = *bij_is_surj*[*OF bij_ctor1*]
lemmas *ctor1_nchotomy* = *surjD*[*OF surj_ctor1*]
lemmas *ctor1_diff* = *inj_eq*[*OF inj_ctor1*]
lemmas *ctor1_cases* = *exE*[*OF ctor1_nchotomy*]
lemmas *bij_ctor2* = *o_bij*[*OF dtor2_o_ctor2 ctor2_o_dtor2*]
lemmas *inj_ctor2* = *bij_is_inj*[*OF bij_ctor2*]
lemmas *surj_ctor2* = *bij_is_surj*[*OF bij_ctor2*]
lemmas *ctor2_nchotomy* = *surjD*[*OF surj_ctor2*]
lemmas *ctor2_diff* = *inj_eq*[*OF inj_ctor2*]
lemmas *ctor2_cases* = *exE*[*OF ctor2_nchotomy*]

Primitive recursion

definition *rec1* **where**
rec1 s1 s2 = *snd o fold1* (<*ctor1 o F1map id fst fst*, *s1*>) (<*ctor2 o F2map id fst fst*, *s2*>)
definition *rec2* **where**
rec2 s1 s2 = *snd o fold2* (<*ctor1 o F1map id fst fst*, *s1*>) (<*ctor2 o F2map id fst fst*, *s2*>)

lemma *fold1_o_ctor1*: *fold1 s1 s2* ∘ *ctor1* = *s1* ∘ *F1map id* (*fold1 s1 s2*) (*fold2 s1 s2*)
⟨*proof*⟩
lemma *fold2_o_ctor2*: *fold2 s1 s2* ∘ *ctor2* = *s2* ∘ *F2map id* (*fold1 s1 s2*) (*fold2 s1 s2*)
⟨*proof*⟩

lemmas *fst_rec1_pair* =
trans[*OF conjunct1*[*OF fold_unique*[*OF*
trans[*OF o_assoc*[*symmetric*] *trans*[*OF arg_cong2*[*of* _ _ _ _ (*o*), *OF refl*
trans[*OF fold1_o_ctor1 convol_o*]]], *OF trans*[*OF fst_convol*]]
trans[*OF o_assoc*[*symmetric*] *trans*[*OF arg_cong2*[*of* _ _ _ _ (*o*), *OF refl*

trans[OF fold2_o_ctor2 convol_o]]], OF trans[OF fst_convol]]]]
fold1_ctor, unfolded F1.map_comp0[of id, unfolded id_o] F2.map_comp0[of id, unfolded id_o] o_assoc,
OF refl refl]

lemmas fst_rec2_pair =
trans[OF conjunct2[OF fold_unique[OF
trans[OF o_assoc[symmetric] trans[OF arg_cong2[of _ _ _ _ (o), OF refl
trans[OF fold1_o_ctor1 convol_o]]], OF trans[OF fst_convol]]
trans[OF o_assoc[symmetric] trans[OF arg_cong2[of _ _ _ _ (o), OF refl
trans[OF fold2_o_ctor2 convol_o]]], OF trans[OF fst_convol]]]]
fold2_ctor, unfolded F1.map_comp0[of id, unfolded id_o] F2.map_comp0[of id, unfolded id_o] o_assoc,
OF refl refl]

theorem rec1: rec1 s1 s2 (ctor1 x) = s1 (F1map id (<id, rec1 s1 s2>) (<id, rec2 s1 s2>) x)
⟨proof⟩

theorem rec2: rec2 s1 s2 (ctor2 x) = s2 (F2map id (<id, rec1 s1 s2>) (<id, rec2 s1 s2>) x)
⟨proof⟩

lemma rec_unique:
f ∘ ctor1 = s1 ∘ F1map id <id , f> <id , g> ⟹
g ∘ ctor2 = s2 ∘ F2map id <id , f> <id , g> ⟹ f = rec1 s1 s2 ∧ g = rec2 s1 s2
⟨proof⟩

Induction

theorem ctor_induct:
⟦⋀x. (⋀a. a ∈ F1set2 x ⟹ phi1 a) ⟹ (⋀a. a ∈ F1set3 x ⟹ phi2 a) ⟹ phi1 (ctor1 x);
⋀x. (⋀a. a ∈ F2set2 x ⟹ phi1 a) ⟹ (⋀a. a ∈ F2set3 x ⟹ phi2 a) ⟹ phi2 (ctor2 x)⟧ ⟹
phi1 a ∧ phi2 b
⟨proof⟩

theorem ctor_induct2:
⟦⋀x y. (⋀a b. a ∈ F1set2 x ⟹ b ∈ F1set2 y ⟹ phi1 a b) ⟹
(⋀a b. a ∈ F1set3 x ⟹ b ∈ F1set3 y ⟹ phi2 a b) ⟹ phi1 (ctor1 x) (ctor1 y);
⋀x y. (⋀a b. a ∈ F2set2 x ⟹ b ∈ F2set2 y ⟹ phi1 a b) ⟹
(⋀a b. a ∈ F2set3 x ⟹ b ∈ F2set3 y ⟹ phi2 a b) ⟹ phi2 (ctor2 x) (ctor2 y)⟧ ⟹
phi1 a1 b1 ∧ phi2 a2 b2
⟨proof⟩

1.8 The Result as an BNF

The map operator

abbreviation IF1map **where** IF1map f ≡ fold1 (ctor1 o (F1map f id id)) (ctor2 o (F2map f id id))
abbreviation IF2map **where** IF2map f ≡ fold2 (ctor1 o (F1map f id id)) (ctor2 o (F2map f id id))

theorem IF1map:
(IF1map f) o ctor1 = ctor1 o (F1map f (IF1map f) (IF2map f))
⟨proof⟩

theorem IF2map:
(IF2map f) o ctor2 = ctor2 o (F2map f (IF1map f) (IF2map f))
⟨proof⟩

lemmas IF1map_simps = o_eq_dest[OF IF1map]
lemmas IF2map_simps = o_eq_dest[OF IF2map]

lemma IFmap_unique:
⟦u o ctor1 = ctor1 o F1map f u v; v o ctor2 = ctor2 o F2map f u v⟧ ⟹
u = IF1map f ∧ v = IF2map f
⟨proof⟩

theorem IF1map_id: IF1map id = id
⟨proof⟩

theorem $IF2map\_id$: $IF2map\ id = id$
$\langle proof \rangle$

theorem $IF1map\_comp$: $IF1map\ (g\ o\ f) = IF1map\ g\ o\ IF1map\ f$
$\langle proof \rangle$

theorem $IF2map\_comp$: $IF2map\ (g\ o\ f) = IF2map\ g\ o\ IF2map\ f$
$\langle proof \rangle$

The bound

abbreviation $IFbd$ **where** $IFbd \equiv bd\_F1\ +c\ bd\_F2$

theorem $IFbd\_card\_order$: $card\_order\ IFbd$
$\langle proof \rangle$

lemma $IFbd\_Cinfinite$: $Cinfinite\ IFbd$
$\langle proof \rangle$

lemma $IFbd\_regularCard$: $regularCard\ IFbd$
$\langle proof \rangle$

lemmas $IFbd\_cinfinite = conjunct1[OF\ IFbd\_Cinfinite]$

The set operator

abbreviation $IF1col$ **where** $IF1col \equiv (\lambda X.\ F1set1\ X \cup (\bigcup (F1set2\ X) \cup \bigcup (F1set3\ X)))$
abbreviation $IF2col$ **where** $IF2col \equiv (\lambda X.\ F2set1\ X \cup (\bigcup (F2set2\ X) \cup \bigcup (F2set3\ X)))$

abbreviation $IF1set$ **where** $IF1set \equiv fold1\ IF1col\ IF2col$
abbreviation $IF2set$ **where** $IF2set \equiv fold2\ IF1col\ IF2col$

abbreviation $IF1in$ **where** $IF1in\ A \equiv \{x.\ IF1set\ x \subseteq A\}$
abbreviation $IF2in$ **where** $IF2in\ A \equiv \{x.\ IF2set\ x \subseteq A\}$

lemma $IF1set$: $IF1set\ o\ ctor1 = IF1col\ o\ (F1map\ id\ IF1set\ IF2set)$
$\langle proof \rangle$

lemma $IF2set$: $IF2set\ o\ ctor2 = IF2col\ o\ (F2map\ id\ IF1set\ IF2set)$
$\langle proof \rangle$

theorem $IF1set\_simps$:
$IF1set\ (ctor1\ x) = F1set1\ x \cup ((\bigcup a \in F1set2\ x.\ IF1set\ a) \cup (\bigcup a \in F1set3\ x.\ IF2set\ a))$
$\langle proof \rangle$

theorem $IF2set\_simps$:
$IF2set\ (ctor2\ x) = F2set1\ x \cup ((\bigcup a \in F2set2\ x.\ IF1set\ a) \cup (\bigcup a \in F2set3\ x.\ IF2set\ a))$
$\langle proof \rangle$

lemmas $F1set1\_IF1set = xt1(3)[OF\ IF1set\_simps\ Un\_upper1]$
lemmas $F1set2\_IF1set = subset\_trans[OF\ UN\_upper\ subset\_trans[OF\ Un\_upper1\ xt1(3)[OF\ IF1set\_simps\ Un\_upper2]]]$
lemmas $F1set3\_IF1set = subset\_trans[OF\ UN\_upper\ subset\_trans[OF\ Un\_upper2\ xt1(3)[OF\ IF1set\_simps\ Un\_upper2]]]$

lemmas $F2set1\_IF2set = xt1(3)[OF\ IF2set\_simps\ Un\_upper1]$
lemmas $F2set2\_IF2set = subset\_trans[OF\ UN\_upper\ subset\_trans[OF\ Un\_upper1\ xt1(3)[OF\ IF2set\_simps\ Un\_upper2]]]$
lemmas $F2set3\_IF2set = subset\_trans[OF\ UN\_upper\ subset\_trans[OF\ Un\_upper2\ xt1(3)[OF\ IF2set\_simps\ Un\_upper2]]]$

The BNF conditions for IF

lemma $IFset\_natural$:
$f\ `\ (IF1set\ x) = IF1set\ (IF1map\ f\ x) \wedge f\ `\ (IF2set\ y) = IF2set\ (IF2map\ f\ y)$
$\langle proof \rangle$

theorem $IF1set\_natural$: $IF1set\ o\ (IF1map\ f) = image\ f\ o\ IF1set$
$\langle proof \rangle$

theorem *IF2set_natural*: $IF2set\ o\ (IF2map\ f) = image\ f\ o\ IF2set$
⟨*proof*⟩

lemma *IFmap_cong*:
$((\forall a \in IF1set\ x.\ f\ a = g\ a) \longrightarrow IF1map\ f\ x = IF1map\ g\ x) \wedge$
$((\forall a \in IF2set\ y.\ f\ a = g\ a) \longrightarrow IF2map\ f\ y = IF2map\ g\ y)$
⟨*proof*⟩

theorem *IF1map_cong*:
$(\bigwedge a.\ a \in IF1set\ x \Longrightarrow f\ a = g\ a) \Longrightarrow IF1map\ f\ x = IF1map\ g\ x$
⟨*proof*⟩

theorem *IF2map_cong*:
$(\bigwedge a.\ a \in IF2set\ x \Longrightarrow f\ a = g\ a) \Longrightarrow IF2map\ f\ x = IF2map\ g\ x$
⟨*proof*⟩

lemma *IFset_bd*:
$|IF1set\ (x :: 'a\ IF1)| <o\ IFbd \wedge |IF2set\ (y :: 'a\ IF2)| <o\ IFbd$
⟨*proof*⟩

lemmas *IF1set_bd* = *conjunct1*[*OF IFset_bd*]
lemmas *IF2set_bd* = *conjunct2*[*OF IFset_bd*]

definition *IF1rel* **where**
$IF1rel\ R =$
$(BNF\_Def.Grp\ (IF1in\ (Collect\ (case\_prod\ R)))\ (IF1map\ fst))\hat{}{-}{-}1\ OO$
$(BNF\_Def.Grp\ (IF1in\ (Collect\ (case\_prod\ R)))\ (IF1map\ snd))$

definition *IF2rel* **where**
$IF2rel\ R =$
$(BNF\_Def.Grp\ (IF2in\ (Collect\ (case\_prod\ R)))\ (IF2map\ fst))\hat{}{-}{-}1\ OO$
$(BNF\_Def.Grp\ (IF2in\ (Collect\ (case\_prod\ R)))\ (IF2map\ snd))$

lemma *in_IF1rel*:
$IF1rel\ R\ x\ y \longleftrightarrow (\exists\ z.\ z \in IF1in\ (Collect\ (case\_prod\ R)) \wedge IF1map\ fst\ z = x \wedge IF1map\ snd\ z = y)$
⟨*proof*⟩

lemma *in_IF2rel*:
$IF2rel\ R\ x\ y \longleftrightarrow (\exists\ z.\ z \in IF2in\ (Collect\ (case\_prod\ R)) \wedge IF2map\ fst\ z = x \wedge IF2map\ snd\ z = y)$
⟨*proof*⟩

lemma *IF1rel_F1rel*: $IF1rel\ R\ (ctor1\ a)\ (ctor1\ b) \longleftrightarrow F1rel\ R\ (IF1rel\ R)\ (IF2rel\ R)\ a\ b$
⟨*proof*⟩

lemma *IF2rel_F2rel*: $IF2rel\ R\ (ctor2\ a)\ (ctor2\ b) \longleftrightarrow F2rel\ R\ (IF1rel\ R)\ (IF2rel\ R)\ a\ b$
⟨*proof*⟩

lemma *Irel_induct*:
assumes *IH1*: $\forall x\ y.\ F1rel\ P1\ P2\ P3\ x\ y \longrightarrow P2\ (ctor1\ x)\ (ctor1\ y)$
and *IH2*: $\forall x\ y.\ F2rel\ P1\ P2\ P3\ x\ y \longrightarrow P3\ (ctor2\ x)\ (ctor2\ y)$
shows $IF1rel\ P1 \leq P2 \wedge IF2rel\ P1 \leq P3$
⟨*proof*⟩

lemma *le_IFrel_Comp*:
$((IF1rel\ R\ OO\ IF1rel\ S)\ x1\ y1 \longrightarrow IF1rel\ (R\ OO\ S)\ x1\ y1) \wedge$
$((IF2rel\ R\ OO\ IF2rel\ S)\ x2\ y2 \longrightarrow IF2rel\ (R\ OO\ S)\ x2\ y2)$
⟨*proof*⟩

lemma *le_IF1rel_Comp*: $IF1rel\ R1\ OO\ IF1rel\ R2 \leq IF1rel\ (R1\ OO\ R2)$
⟨*proof*⟩

lemma *le_IF2rel_Comp*: $IF2rel\ R1\ OO\ IF2rel\ R2 \leq IF2rel\ (R1\ OO\ R2)$
⟨*proof*⟩

context includes *lifting_syntax*
begin

lemma *fold_transfer*:
  ((*F1rel R S T* ===> *S*) ===> (*F2rel R S T* ===> *T*) ===> *IF1rel R* ===> *S*) *fold1 fold1* ∧
  ((*F1rel R S T* ===> *S*) ===> (*F2rel R S T* ===> *T*) ===> *IF2rel R* ===> *T*) *fold2 fold2*
  ⟨*proof*⟩

end

definition *IF1wit x* = *ctor1* (*wit2_F1 x* (*ctor2 wit_F2*))
definition *IF2wit* = *ctor2 wit_F2*

lemma *IF1wit*: $x \in$ *IF1set* (*IF1wit y*) $\Longrightarrow x = y$
  ⟨*proof*⟩

lemma *IF2wit*: $x \in$ *IF2set IF2wit* $\Longrightarrow$ *False*
  ⟨*proof*⟩

⟨*ML*⟩

bnf *'a IF1*
  map: *IF1map*
  sets: *IF1set*
  bd: *IFbd*
  wits: *IF1wit*
  rel: *IF1rel*
    ⟨*proof*⟩

bnf *'a IF2*
  map: *IF2map*
  sets: *IF2set*
  bd: *IFbd*
  wits: *IF2wit*
  rel: *IF2rel*
    ⟨*proof*⟩

2 Greatest Fixpoint (a.k.a. Codatatype)

unbundle *cardinal_syntax*

'b1 = ('a, 'b1, 'b2) F1
'b2 = ('a, 'b1, 'b2) F2

To build a witness scenario, let us assume

('a, 'b1, 'b2) F1 = 'a * 'b1 + 'a * 'b2
('a, 'b1, 'b2) F2 = unit + 'b1 * 'b2

⟨*ML*⟩
declare [[*bnf_internals*]]
bnf-axiomatization (*F1set1*: *'a*, *F1set2*: *'b1*, *F1set3*: *'b2*) *F1*
  [*wits*: *'a* ⇒ *'b1* ⇒ (*'a*, *'b1*, *'b2*) *F1* *'a* ⇒ *'b2* ⇒ (*'a*, *'b1*, *'b2*) *F1*]
  for *map*: *F1map rel*: *F1rel*

bnf-axiomatization (*F2set1*: *'a*, *F2set2*: *'b1*, *F2set3*: *'b2*) *F2*
  [*wits*: (*'a*, *'b1*, *'b2*) *F2*]
  for *map*: *F2map rel*: *F2rel*

lemma *F1rel_cong*: ⟦*R1* = *S1*; *R2* = *S2*; *R3* = *S3*⟧ $\Longrightarrow$ *F1rel R1 R2 R3* = *F1rel S1 S2 S3*
  ⟨*proof*⟩

lemma *F2rel_cong*: ⟦*R1* = *S1*; *R2* = *S2*; *R3* = *S3*⟧ $\Longrightarrow$ *F2rel R1 R2 R3* = *F2rel S1 S2 S3*

⟨proof⟩

abbreviation $F1in$ :: $'a1\ set \Rightarrow 'a2\ set \Rightarrow 'a3\ set \Rightarrow (('a1, 'a2, 'a3)\ F1)\ set$ **where**
$F1in\ A1\ A2\ A3 \equiv \{x.\ F1set1\ x \subseteq A1 \wedge F1set2\ x \subseteq A2 \wedge F1set3\ x \subseteq A3\}$
abbreviation $F2in$ :: $'a1\ set \Rightarrow 'a2\ set \Rightarrow 'a3\ set \Rightarrow (('a1, 'a2, 'a3)\ F2)\ set$ **where**
$F2in\ A1\ A2\ A3 \equiv \{x.\ F2set1\ x \subseteq A1 \wedge F2set2\ x \subseteq A2 \wedge F2set3\ x \subseteq A3\}$

lemma *F1map_comp_id*: $F1map\ g1\ g2\ g3\ (F1map\ id\ f2\ f3\ x) = F1map\ g1\ (g2\ o\ f2)\ (g3\ o\ f3)\ x$
⟨proof⟩

lemmas *F1in_mono23* = *F1.in_mono*[*OF subset_refl*]
lemmas *F1in_mono23'* = *subsetD*[*OF F1in_mono23*]

lemma *F1map_congL*: $[\![\forall a \in F1set2\ x.\ f\ a = a; \forall a \in F1set3\ x.\ g\ a = a]\!] \Longrightarrow$
$F1map\ id\ f\ g\ x = x$
⟨proof⟩

lemma *F2map_comp_id*: $F2map\ g1\ g2\ g3\ (F2map\ id\ f2\ f3\ x) = F2map\ g1\ (g2\ o\ f2)\ (g3\ o\ f3)\ x$
⟨proof⟩

lemmas *F2in_mono23* = *F2.in_mono*[*OF subset_refl*]
lemmas *F2in_mono23'* = *subsetD*[*OF F2in_mono23*]

lemma *F2map_congL*: $[\![\forall a \in F2set2\ x.\ f\ a = a; \forall a \in F2set3\ x.\ g\ a = a]\!] \Longrightarrow$
$F2map\ id\ f\ g\ x = x$
⟨proof⟩

2.1 Coalgebra

definition *coalg* **where**
$coalg\ B1\ B2\ s1\ s2 =$
$((\forall a \in B1.\ s1\ a \in F1in\ (UNIV :: 'a\ set)\ B1\ B2) \wedge (\forall a \in B2.\ s2\ a \in F2in\ (UNIV :: 'a\ set)\ B1\ B2))$

lemmas *coalg_F1in* = *bspec*[*OF conjunct1*[*OF iffD1*[*OF coalg_def*]]]
lemmas *coalg_F2in* = *bspec*[*OF conjunct2*[*OF iffD1*[*OF coalg_def*]]]

lemma *coalg_F1set2*:
$[\![coalg\ B1\ B2\ s1\ s2; a \in B1]\!] \Longrightarrow F1set2\ (s1\ a) \subseteq B1$
⟨proof⟩

lemma *coalg_F1set3*:
$[\![coalg\ B1\ B2\ s1\ s2; a \in B1]\!] \Longrightarrow F1set3\ (s1\ a) \subseteq B2$
⟨proof⟩

lemma *coalg_F2set2*:
$[\![coalg\ B1\ B2\ s1\ s2; a \in B2]\!] \Longrightarrow F2set2\ (s2\ a) \subseteq B1$
⟨proof⟩

lemma *coalg_F2set3*:
$[\![coalg\ B1\ B2\ s1\ s2; a \in B2]\!] \Longrightarrow F2set3\ (s2\ a) \subseteq B2$
⟨proof⟩

2.2 Type-coalgebra

abbreviation $tcoalg\ s1\ s2 \equiv coalg\ UNIV\ UNIV\ s1\ s2$

lemma *tcoalg*: $tcoalg\ s1\ s2$
⟨proof⟩

2.3 Morphism

definition *mor* **where**
$mor\ B1\ B2\ s1\ s2\ B1'\ B2'\ s1'\ s2'\ f\ g =$
$(((\forall a \in B1.\ f\ a \in B1') \wedge (\forall a \in B2.\ g\ a \in B2')) \wedge$

$((\forall z \in B1.\ F1map\ (id :: 'a \Rightarrow 'a)\ f\ g\ (s1\ z) = s1'\ (f\ z)) \wedge$
$(\forall z \in B2.\ F2map\ (id :: 'a \Rightarrow 'a)\ f\ g\ (s2\ z) = s2'\ (g\ z))))$

lemma *mor_image1*: $mor\ B1\ B2\ s1\ s2\ B1'\ B2'\ s1'\ s2'\ f\ g \Longrightarrow f\ `\ B1 \subseteq B1'$
⟨*proof*⟩

lemma *mor_image2*: $mor\ B1\ B2\ s1\ s2\ B1'\ B2'\ s1'\ s2'\ f\ g \Longrightarrow g\ `\ B2 \subseteq B2'$
⟨*proof*⟩

lemmas *mor_image1'* = *subsetD*[*OF mor_image1 imageI*]
lemmas *mor_image2'* = *subsetD*[*OF mor_image2 imageI*]

lemma *morE1*: $[\![mor\ B1\ B2\ s1\ s2\ B1'\ B2'\ s1'\ s2'\ f\ g;\ z \in B1]\!]$
$\Longrightarrow F1map\ id\ f\ g\ (s1\ z) = s1'\ (f\ z)$
⟨*proof*⟩

lemma *morE2*: $[\![mor\ B1\ B2\ s1\ s2\ B1'\ B2'\ s1'\ s2'\ f\ g;\ z \in B2]\!]$
$\Longrightarrow F2map\ id\ f\ g\ (s2\ z) = s2'\ (g\ z)$
⟨*proof*⟩

lemma *mor_incl*: $[\![B1 \subseteq B1';\ B2 \subseteq B2']\!] \Longrightarrow mor\ B1\ B2\ s1\ s2\ B1'\ B2'\ s1\ s2\ id\ id$
⟨*proof*⟩

lemmas *mor_id* = *mor_incl*[*OF subset_refl subset_refl*]

lemma *mor_comp*:
$[\![mor\ B1\ B2\ s1\ s2\ B1'\ B2'\ s1'\ s2'\ f\ g;$
$mor\ B1'\ B2'\ s1'\ s2'\ B1''\ B2''\ s1''\ s2''\ f'\ g']\!] \Longrightarrow$
$mor\ B1\ B2\ s1\ s2\ B1''\ B2''\ s1''\ s2''\ (f'\ o\ f)\ (g'\ o\ g)$
⟨*proof*⟩

lemma *mor_cong*: $[\![\ f' = f;\ g' = g;\ mor\ B1\ B2\ s1\ s2\ B1'\ B2'\ s1'\ s2'\ f\ g]\!] \Longrightarrow$
$mor\ B1\ B2\ s1\ s2\ B1'\ B2'\ s1'\ s2'\ f'\ g'$
⟨*proof*⟩

lemma *mor_UNIV*: $mor\ UNIV\ UNIV\ s1\ s2\ UNIV\ UNIV\ s1'\ s2'\ f1\ f2 \longleftrightarrow$
$F1map\ id\ f1\ f2\ o\ s1 = s1'\ o\ f1 \wedge F2map\ id\ f1\ f2\ o\ s2 = s2'\ o\ f2$
⟨*proof*⟩

lemma *mor_str*:
$mor\ UNIV\ UNIV\ s1\ s2\ UNIV\ UNIV\ (F1map\ id\ s1\ s2)\ (F2map\ id\ s1\ s2)\ s1\ s2$
⟨*proof*⟩

lemma *mor_case_sum*:
mor UNIV UNIV s1 s2 UNIV UNIV (*case_sum* (*F1map id Inl Inl o s1*) *s1'*) (*case_sum* (*F2map id Inl Inl o s2*) *s2'*) *Inl Inl*
⟨*proof*⟩

2.4 Bisimulations

definition *bis* **where**
$bis\ B1\ B2\ s1\ s2\ B1'\ B2'\ s1'\ s2'\ R1\ R2 =$
$((R1 \subseteq B1 \times B1' \wedge R2 \subseteq B2 \times B2') \wedge$
$((\forall b1\ b1'.\ (b1,\ b1') \in R1 \longrightarrow$
$(\exists z \in F1in\ UNIV\ R1\ R2.$
$F1map\ id\ fst\ fst\ z = s1\ b1 \wedge F1map\ id\ snd\ snd\ z = s1'\ b1')) \wedge$
$(\forall b2\ b2'.\ (b2,\ b2') \in R2 \longrightarrow$
$(\exists z \in F2in\ UNIV\ R1\ R2.$
$F2map\ id\ fst\ fst\ z = s2\ b2 \wedge F2map\ id\ snd\ snd\ z = s2'\ b2'))))$

lemma *bis_cong*: $[\![bis\ B1\ B2\ s1\ s2\ B1'\ B2'\ s1'\ s2'\ R1\ R2;\ R1' = R1;\ R2' = R2]\!] \Longrightarrow$
$bis\ B1\ B2\ s1\ s2\ B1'\ B2'\ s1'\ s2'\ R1'\ R2'$
⟨*proof*⟩

lemma *bis_Frel*:
bis B1 B2 s1 s2 B1′ B2′ s1′ s2′ R1 R2 ⟷
(*R1* ⊆ *B1* × *B1′* ∧ *R2* ⊆ *B2* × *B2′*) ∧
((∀ *b1 b1′*. (*b1*, *b1′*) ∈ *R1* ⟶ *F1rel* (=) (*in_rel R1*) (*in_rel R2*) (*s1 b1*) (*s1′ b1′*)) ∧
(∀ *b2 b2′*. (*b2*, *b2′*) ∈ *R2* ⟶ *F2rel* (=) (*in_rel R1*) (*in_rel R2*) (*s2 b2*) (*s2′ b2′*)))
⟨*proof*⟩

lemma *bis_converse*:
bis B1 B2 s1 s2 B1′ B2′ s1′ s2′ R1 R2 ⟹
bis B1′ B2′ s1′ s2′ B1 B2 s1 s2 (*R1^−1*) (*R2^−1*)
⟨*proof*⟩

lemma *bis_Comp*:
⟦*bis B1 B2 s1 s2 B1′ B2′ s1′ s2′ P1 P2*;
bis B1′ B2′ s1′ s2′ B1′′ B2′′ s1′′ s2′′ Q1 Q2⟧ ⟹
bis B1 B2 s1 s2 B1′′ B2′′ s1′′ s2′′ (*P1 O Q1*) (*P2 O Q2*)
⟨*proof*⟩

lemma *bis_Gr*: ⟦*coalg B1 B2 s1 s2*; *mor B1 B2 s1 s2 B1′ B2′ s1′ s2′ f1 f2*⟧ ⟹
bis B1 B2 s1 s2 B1′ B2′ s1′ s2′ (*BNF_Def.Gr B1 f1*) (*BNF_Def.Gr B2 f2*)
⟨*proof*⟩

lemmas *bis_image2* = *bis_cong*[*OF bis_Comp*[*OF bis_converse*[*OF bis_Gr*] *bis_Gr*] *image2_Gr image2_Gr*]
lemmas *bis_diag* = *bis_cong*[*OF bis_Gr*[*OF _ mor_id*] *Id_on_Gr Id_on_Gr*]

lemma *bis_Union*: ∀ *i* ∈ *I*. *bis B1 B2 s1 s2 B1 B2 s1 s2* (*R1i i*) (*R2i i*) ⟹
bis B1 B2 s1 s2 B1 B2 s1 s2 (⋃*i*∈ *I*. *R1i i*) (⋃*i*∈ *I*. *R2i i*)
⟨*proof*⟩

abbreviation *sbis B1 B2 s1 s2 R1 R2* ≡ *bis B1 B2 s1 s2 B1 B2 s1 s2 R1 R2*

definition *lsbis1* **where** *lsbis1 B1 B2 s1 s2* =
(⋃*R* ∈ {(*R1*, *R2*) | *R1 R2* . *sbis B1 B2 s1 s2 R1 R2*}. *fst R*)

definition *lsbis2* **where** *lsbis2 B1 B2 s1 s2* =
(⋃*R* ∈ {(*R1*, *R2*) | *R1 R2* . *sbis B1 B2 s1 s2 R1 R2*}. *snd R*)

lemma *sbis_lsbis*:
sbis B1 B2 s1 s2 (*lsbis1 B1 B2 s1 s2*) (*lsbis2 B1 B2 s1 s2*)
⟨*proof*⟩

lemmas *lsbis1_incl* = *conjunct1*[*OF conjunct1*[*OF iffD1*[*OF bis_def*]], *OF sbis_lsbis*]
lemmas *lsbis2_incl* = *conjunct2*[*OF conjunct1*[*OF iffD1*[*OF bis_def*]], *OF sbis_lsbis*]
lemmas *lsbisE1* =
mp[*OF spec*[*OF spec*[*OF conjunct1*[*OF conjunct2*[*OF iffD1*[*OF bis_def*]], *OF sbis_lsbis*]]]]
lemmas *lsbisE2* =
mp[*OF spec*[*OF spec*[*OF conjunct2*[*OF conjunct2*[*OF iffD1*[*OF bis_def*]], *OF sbis_lsbis*]]]]

lemma *incl_lsbis1*: *sbis B1 B2 s1 s2 R1 R2* ⟹ *R1* ⊆ *lsbis1 B1 B2 s1 s2*
⟨*proof*⟩

lemma *incl_lsbis2*: *sbis B1 B2 s1 s2 R1 R2* ⟹ *R2* ⊆ *lsbis2 B1 B2 s1 s2*
⟨*proof*⟩

lemma *equiv_lsbis1*: *coalg B1 B2 s1 s2* ⟹ *equiv B1* (*lsbis1 B1 B2 s1 s2*)
⟨*proof*⟩

lemma *equiv_lsbis2*: *coalg B1 B2 s1 s2* ⟹ *equiv B2* (*lsbis2 B1 B2 s1 s2*)
⟨*proof*⟩

2.5 The Tree Coalgebra

typedef bd_type_F = UNIV :: (bd_type_F1 + bd_type_F2) suc set
⟨proof⟩

type-synonym 'a carrier = ((bd_type_F + bd_type_F) list set ×
((bd_type_F + bd_type_F) list ⇒ (('a, bd_type_F, bd_type_F) F1 + ('a, bd_type_F, bd_type_F) F2)))

abbreviation bd_F ≡ dir_image (card_suc (bd_F1 +c bd_F2)) Abs_bd_type_F

lemmas sum_card_order = card_order_csum[OF F1.bd_card_order F2.bd_card_order]
lemmas sum_Cinfinite = Cinfinite_csum1[OF F1.bd_Cinfinite]
lemmas bd_F = dir_image[OF Abs_bd_type_F_inject[OF UNIV_I UNIV_I] Card_order_card_suc[OF sum_card_order]]
lemmas bd_F_Cinfinite = Cinfinite_cong[OF bd_F Cinfinite_card_suc[OF sum_Cinfinite sum_card_order]]
lemmas bd_F_Card_order = Card_order_ordIso[OF Card_order_card_suc[OF sum_card_order] ordIso_symmetric[OF bd_F]]
lemma bd_F_card_order: card_order bd_F
⟨proof⟩
lemmas bd_F_regularCard = regularCard_ordIso[OF bd_F Cinfinite_card_suc[OF sum_Cinfinite sum_card_order]
regularCard_card_suc[OF sum_card_order sum_Cinfinite]
]

lemmas F1set1_bd' = ordLess_transitive[OF F1.set_bd(1) ordLess_ordIso_trans[OF
ordLeq_ordLess_trans[OF ordLeq_csum1[OF F1.bd_Card_order] card_suc_greater[OF sum_card_order]]
bd_F]]
lemmas F1set2_bd' = ordLess_transitive[OF F1.set_bd(2) ordLess_ordIso_trans[OF
ordLeq_ordLess_trans[OF ordLeq_csum1[OF F1.bd_Card_order] card_suc_greater[OF sum_card_order]]
bd_F]]
lemmas F1set3_bd' = ordLess_transitive[OF F1.set_bd(3) ordLess_ordIso_trans[OF
ordLeq_ordLess_trans[OF ordLeq_csum1[OF F1.bd_Card_order] card_suc_greater[OF sum_card_order]]
bd_F]]

lemmas F2set1_bd' = ordLess_transitive[OF F2.set_bd(1) ordLess_ordIso_trans[OF
ordLeq_ordLess_trans[OF ordLeq_csum2[OF F2.bd_Card_order] card_suc_greater[OF sum_card_order]]
bd_F]]
lemmas F2set2_bd' = ordLess_transitive[OF F2.set_bd(2) ordLess_ordIso_trans[OF
ordLeq_ordLess_trans[OF ordLeq_csum2[OF F2.bd_Card_order] card_suc_greater[OF sum_card_order]]
bd_F]]
lemmas F2set3_bd' = ordLess_transitive[OF F2.set_bd(3) ordLess_ordIso_trans[OF
ordLeq_ordLess_trans[OF ordLeq_csum2[OF F2.bd_Card_order] card_suc_greater[OF sum_card_order]]
bd_F]]

abbreviation Succ1 Kl kl ≡ {k1. Inl k1 ∈ BNF_Greatest_Fixpoint.Succ Kl kl}
abbreviation Succ2 Kl kl ≡ {k2. Inr k2 ∈ BNF_Greatest_Fixpoint.Succ Kl kl}

definition isNode1 **where**
isNode1 Kl lab kl = (∃ x1. lab kl = Inl x1 ∧ F1set2 x1 = Succ1 Kl kl ∧ F1set3 x1 = Succ2 Kl kl)

definition isNode2 **where**
isNode2 Kl lab kl = (∃ x2. lab kl = Inr x2 ∧ F2set2 x2 = Succ1 Kl kl ∧ F2set3 x2 = Succ2 Kl kl)

abbreviation isTree **where**
isTree Kl lab ≡ ([] ∈ Kl ∧
(∀ kl ∈ Kl. (∀ k1 ∈ Succ1 Kl kl. isNode1 Kl lab (kl @ [Inl k1])) ∧
(∀ k2 ∈ Succ2 Kl kl. isNode2 Kl lab (kl @ [Inr k2]))))

definition carT1 **where**
carT1 = {(Kl :: (bd_type_F + bd_type_F) list set, lab) | Kl lab. isTree Kl lab ∧ isNode1 Kl lab []}

definition carT2 **where**
carT2 = {(Kl :: (bd_type_F + bd_type_F) list set, lab) | Kl lab. isTree Kl lab ∧ isNode2 Kl lab []}

definition strT1 **where**
strT1 = (case_prod (%Kl lab. case_sum (F1map id

(λ*k1*. (*BNF_Greatest_Fixpoint.Shift Kl* (*Inl k1*), *BNF_Greatest_Fixpoint.shift lab* (*Inl k1*)))
(λ*k2*. (*BNF_Greatest_Fixpoint.Shift Kl* (*Inr k2*), *BNF_Greatest_Fixpoint.shift lab* (*Inr k2*)))) *undefined* (*lab* []))))

definition *strT2* **where**
strT2 = (*case_prod* (%*Kl lab*. *case_sum undefined* (*F2map id*
(λ*k1*. (*BNF_Greatest_Fixpoint.Shift Kl* (*Inl k1*), *BNF_Greatest_Fixpoint.shift lab* (*Inl k1*)))
(λ*k2*. (*BNF_Greatest_Fixpoint.Shift Kl* (*Inr k2*), *BNF_Greatest_Fixpoint.shift lab* (*Inr k2*)))) (*lab* [])))

lemma *coalg_T*: *coalg carT1 carT2 strT1 strT2*
⟨*proof*⟩

abbreviation *tobd_F12* **where** *tobd_F12 s1 x* ≡ *toCard* (*F1set2* (*s1 x*)) *bd_F*
abbreviation *tobd_F13* **where** *tobd_F13 s1 x* ≡ *toCard* (*F1set3* (*s1 x*)) *bd_F*
abbreviation *tobd_F22* **where** *tobd_F22 s2 x* ≡ *toCard* (*F2set2* (*s2 x*)) *bd_F*
abbreviation *tobd_F23* **where** *tobd_F23 s2 x* ≡ *toCard* (*F2set3* (*s2 x*)) *bd_F*
abbreviation *frombd_F12* **where** *frombd_F12 s1 x* ≡ *fromCard* (*F1set2* (*s1 x*)) *bd_F*
abbreviation *frombd_F13* **where** *frombd_F13 s1 x* ≡ *fromCard* (*F1set3* (*s1 x*)) *bd_F*
abbreviation *frombd_F22* **where** *frombd_F22 s2 x* ≡ *fromCard* (*F2set2* (*s2 x*)) *bd_F*
abbreviation *frombd_F23* **where** *frombd_F23 s2 x* ≡ *fromCard* (*F2set3* (*s2 x*)) *bd_F*

lemmas *tobd_F12_inj* = *toCard_inj*[*OF ordLess_imp_ordLeq*[*OF F1set2_bd'*] *bd_F_Card_order*]
lemmas *tobd_F13_inj* = *toCard_inj*[*OF ordLess_imp_ordLeq*[*OF F1set3_bd'*] *bd_F_Card_order*]
lemmas *tobd_F22_inj* = *toCard_inj*[*OF ordLess_imp_ordLeq*[*OF F2set2_bd'*] *bd_F_Card_order*]
lemmas *tobd_F23_inj* = *toCard_inj*[*OF ordLess_imp_ordLeq*[*OF F2set3_bd'*] *bd_F_Card_order*]
lemmas *frombd_F12_tobd_F12* = *fromCard_toCard*[*OF ordLess_imp_ordLeq*[*OF F1set2_bd'*] *bd_F_Card_order*]
lemmas *frombd_F13_tobd_F13* = *fromCard_toCard*[*OF ordLess_imp_ordLeq*[*OF F1set3_bd'*] *bd_F_Card_order*]
lemmas *frombd_F22_tobd_F22* = *fromCard_toCard*[*OF ordLess_imp_ordLeq*[*OF F2set2_bd'*] *bd_F_Card_order*]
lemmas *frombd_F23_tobd_F23* = *fromCard_toCard*[*OF ordLess_imp_ordLeq*[*OF F2set3_bd'*] *bd_F_Card_order*]

definition *Lev* **where**
Lev s1 s2 = *rec_nat* (%*a*. {[]}, %*b*. {[]})
(%*n rec*.
(%*a1*.
{*Inl* (*tobd_F12 s1 a1 b1*) # *kl* | *b1 kl*. *b1* ∈ *F1set2* (*s1 a1*) ∧ *kl* ∈ *fst rec b1*} ∪
{*Inr* (*tobd_F13 s1 a1 b2*) # *kl* | *b2 kl*. *b2* ∈ *F1set3* (*s1 a1*) ∧ *kl* ∈ *snd rec b2*},
%*a2*.
{*Inl* (*tobd_F22 s2 a2 b1*) # *kl* | *b1 kl*. *b1* ∈ *F2set2* (*s2 a2*) ∧ *kl* ∈ *fst rec b1*} ∪
{*Inr* (*tobd_F23 s2 a2 b2*) # *kl* | *b2 kl*. *b2* ∈ *F2set3* (*s2 a2*) ∧ *kl* ∈ *snd rec b2*}))

abbreviation *Lev1* **where** *Lev1 s1 s2 n* ≡ *fst* (*Lev s1 s2 n*)
abbreviation *Lev2* **where** *Lev2 s1 s2 n* ≡ *snd* (*Lev s1 s2 n*)

lemmas *Lev1_0* = *fun_cong*[*OF fstI*[*OF rec_nat_0_imp*[*OF Lev_def*]]]
lemmas *Lev2_0* = *fun_cong*[*OF sndI*[*OF rec_nat_0_imp*[*OF Lev_def*]]]
lemmas *Lev1_Suc* = *fun_cong*[*OF fstI*[*OF rec_nat_Suc_imp*[*OF Lev_def*]]]
lemmas *Lev2_Suc* = *fun_cong*[*OF sndI*[*OF rec_nat_Suc_imp*[*OF Lev_def*]]]

definition *rv* **where**
rv s1 s2 = *rec_list* (%*b1*. *Inl b1*, %*b2*. *Inr b2*)
(%*k kl rec*.
(%*b1*. *case_sum* (%*k1*. *fst rec* (*frombd_F12 s1 b1 k1*)) (%*k2*. *snd rec* (*frombd_F13 s1 b1 k2*)) *k*,
%*b2*. *case_sum* (%*k1*. *fst rec* (*frombd_F22 s2 b2 k1*)) (%*k2*. *snd rec* (*frombd_F23 s2 b2 k2*)) *k*))

abbreviation *rv1* **where** *rv1 s1 s2 kl* ≡ *fst* (*rv s1 s2 kl*)
abbreviation *rv2* **where** *rv2 s1 s2 kl* ≡ *snd* (*rv s1 s2 kl*)

lemmas *rv1_Nil* = *fun_cong*[*OF fstI*[*OF rec_list_Nil_imp*[*OF rv_def*]]]
lemmas *rv2_Nil* = *fun_cong*[*OF sndI*[*OF rec_list_Nil_imp*[*OF rv_def*]]]
lemmas *rv1_Cons* = *fun_cong*[*OF fstI*[*OF rec_list_Cons_imp*[*OF rv_def*]]]
lemmas *rv2_Cons* = *fun_cong*[*OF sndI*[*OF rec_list_Cons_imp*[*OF rv_def*]]]

abbreviation *Lab1 s1 s2 b1 kl* ≡
  (*case_sum* (%*k*. *Inl* (*F1map id* (*tobd_F12 s1 k*) (*tobd_F13 s1 k*) (*s1 k*)))
      (%*k*. *Inr* (*F2map id* (*tobd_F22 s2 k*) (*tobd_F23 s2 k*) (*s2 k*))) (*rv1 s1 s2 kl b1*))

abbreviation *Lab2 s1 s2 b2 kl* ≡
  (*case_sum* (%*k*. *Inl* (*F1map id* (*tobd_F12 s1 k*) (*tobd_F13 s1 k*) (*s1 k*)))
      (%*k*. *Inr* (*F2map id* (*tobd_F22 s2 k*) (*tobd_F23 s2 k*) (*s2 k*))) (*rv2 s1 s2 kl b2*))

definition *beh1 s1 s2 a* = ($\bigcup$ *n*. *Lev1 s1 s2 n a*, *Lab1 s1 s2 a*)
definition *beh2 s1 s2 a* = ($\bigcup$ *n*. *Lev2 s1 s2 n a*, *Lab2 s1 s2 a*)

lemma *length_Lev*:
  ∀ *kl b1 b2*. (*kl* ∈ *Lev1 s1 s2 n b1* ⟶ *length kl* = *n*) ∧
      (*kl* ∈ *Lev2 s1 s2 n b2* ⟶ *length kl* = *n*)
  ⟨*proof*⟩

lemmas *length_Lev1* = *mp*[*OF conjunct1*[*OF spec*[*OF spec* [*OF spec*[*OF length_Lev*]]]]]
lemmas *length_Lev2* = *mp*[*OF conjunct2*[*OF spec*[*OF spec* [*OF spec*[*OF length_Lev*]]]]]

lemma *length_Lev1'*: *kl* ∈ *Lev1 s1 s2 n a* ⟹ *kl* ∈ *Lev1 s1 s2* (*length kl*) *a*
  ⟨*proof*⟩

lemma *length_Lev2'*: *kl* ∈ *Lev2 s1 s2 n a* ⟹ *kl* ∈ *Lev2 s1 s2* (*length kl*) *a*
  ⟨*proof*⟩

lemma *rv_last*:
  ∀ *k b1 b2*.
   ((∃ *b1'*. *rv1 s1 s2* (*kl* @ [*Inl k*]) *b1* = *Inl b1'*) ∧
    (∃ *b1'*. *rv1 s1 s2* (*kl* @ [*Inr k*]) *b1* = *Inr b1'*)) ∧
   ((∃ *b2'*. *rv2 s1 s2* (*kl* @ [*Inl k*]) *b2* = *Inl b2'*) ∧
    (∃ *b2'*. *rv2 s1 s2* (*kl* @ [*Inr k*]) *b2* = *Inr b2'*))
  ⟨*proof*⟩

lemmas *rv_last'* = *spec*[*OF spec*[*OF spec*[*OF rv_last*]]]
lemmas *rv1_Inl_last* = *conjunct1*[*OF conjunct1*[*OF rv_last'*]]
lemmas *rv1_Inr_last* = *conjunct2*[*OF conjunct1*[*OF rv_last'*]]
lemmas *rv2_Inl_last* = *conjunct1*[*OF conjunct2*[*OF rv_last'*]]
lemmas *rv2_Inr_last* = *conjunct2*[*OF conjunct2*[*OF rv_last'*]]

lemma *Fset_Lev*:
  ∀ *kl b1 b2 b1' b2' b1'' b2''*.
   (*kl* ∈ *Lev1 s1 s2 n b1* ⟶
    ((*rv1 s1 s2 kl b1* = *Inl b1'* ⟶
      (*b1''* ∈ *F1set2* (*s1 b1'*) ⟶
        kl @ [*Inl* (*tobd_F12 s1 b1' b1''*)] ∈ *Lev1 s1 s2* (*Suc n*) *b1*) ∧
      (*b2''* ∈ *F1set3* (*s1 b1'*) ⟶
        kl @ [*Inr* (*tobd_F13 s1 b1' b2''*)] ∈ *Lev1 s1 s2* (*Suc n*) *b1*)) ∧
     (*rv1 s1 s2 kl b1* = *Inr b2'* ⟶
      (*b1''* ∈ *F2set2* (*s2 b2'*) ⟶
        kl @ [*Inl* (*tobd_F22 s2 b2' b1''*)] ∈ *Lev1 s1 s2* (*Suc n*) *b1*) ∧
      (*b2''* ∈ *F2set3* (*s2 b2'*) ⟶
        kl @ [*Inr* (*tobd_F23 s2 b2' b2''*)] ∈ *Lev1 s1 s2* (*Suc n*) *b1*)))) ∧
   (*kl* ∈ *Lev2 s1 s2 n b2* ⟶
    ((*rv2 s1 s2 kl b2* = *Inl b1'* ⟶
      (*b1''* ∈ *F1set2* (*s1 b1'*) ⟶
        kl @ [*Inl* (*tobd_F12 s1 b1' b1''*)] ∈ *Lev2 s1 s2* (*Suc n*) *b2*) ∧
      (*b2''* ∈ *F1set3* (*s1 b1'*) ⟶
        kl @ [*Inr* (*tobd_F13 s1 b1' b2''*)] ∈ *Lev2 s1 s2* (*Suc n*) *b2*)) ∧
     (*rv2 s1 s2 kl b2* = *Inr b2'* ⟶
      (*b1''* ∈ *F2set2* (*s2 b2'*) ⟶

kl @ [Inl (tobd_F22 s2 b2′ b1′′)] ∈ Lev2 s1 s2 (Suc n) b2) ∧
(b2′′ ∈ F2set3 (s2 b2′) ⟶
kl @ [Inr (tobd_F23 s2 b2′ b2′′)] ∈ Lev2 s1 s2 (Suc n) b2))))
⟨proof⟩

lemmas *Fset_Lev′* = *spec*[*OF spec*[*OF spec*[*OF spec*[*OF spec*[*OF spec*[*OF spec*[*OF Fset_Lev*]]]]]]]
lemmas *F1set2_Lev1* = *mp*[*OF conjunct1*[*OF mp*[*OF conjunct1*[*OF mp*[*OF conjunct1*[*OF Fset_Lev′*]]]]]]
lemmas *F1set2_Lev2* = *mp*[*OF conjunct1*[*OF mp*[*OF conjunct1*[*OF mp*[*OF conjunct2*[*OF Fset_Lev′*]]]]]]
lemmas *F2set2_Lev1* = *mp*[*OF conjunct1*[*OF mp*[*OF conjunct2*[*OF mp*[*OF conjunct1*[*OF Fset_Lev′*]]]]]]
lemmas *F2set2_Lev2* = *mp*[*OF conjunct1*[*OF mp*[*OF conjunct2*[*OF mp*[*OF conjunct2*[*OF Fset_Lev′*]]]]]]
lemmas *F1set3_Lev1* = *mp*[*OF conjunct2*[*OF mp*[*OF conjunct1*[*OF mp*[*OF conjunct1*[*OF Fset_Lev′*]]]]]]
lemmas *F1set3_Lev2* = *mp*[*OF conjunct2*[*OF mp*[*OF conjunct1*[*OF mp*[*OF conjunct2*[*OF Fset_Lev′*]]]]]]
lemmas *F2set3_Lev1* = *mp*[*OF conjunct2*[*OF mp*[*OF conjunct2*[*OF mp*[*OF conjunct1*[*OF Fset_Lev′*]]]]]]
lemmas *F2set3_Lev2* = *mp*[*OF conjunct2*[*OF mp*[*OF conjunct2*[*OF mp*[*OF conjunct2*[*OF Fset_Lev′*]]]]]]

lemma *Fset_image_Lev*:
∀ kl k b1 b2 b1′ b2′.
(kl ∈ Lev1 s1 s2 n b1 ⟶
(kl @ [Inl k] ∈ Lev1 s1 s2 (Suc n) b1 ⟶
(rv1 s1 s2 kl b1 = Inl b1′ ⟶ k ∈ tobd_F12 s1 b1′ ‘ F1set2 (s1 b1′)) ∧
(rv1 s1 s2 kl b1 = Inr b2′ ⟶ k ∈ tobd_F22 s2 b2′ ‘ F2set2 (s2 b2′))) ∧
(kl @ [Inr k] ∈ Lev1 s1 s2 (Suc n) b1 ⟶
(rv1 s1 s2 kl b1 = Inl b1′ ⟶ k ∈ tobd_F13 s1 b1′ ‘ F1set3 (s1 b1′)) ∧
(rv1 s1 s2 kl b1 = Inr b2′ ⟶ k ∈ tobd_F23 s2 b2′ ‘ F2set3 (s2 b2′)))) ∧
(kl ∈ Lev2 s1 s2 n b2 ⟶
(kl @ [Inl k] ∈ Lev2 s1 s2 (Suc n) b2 ⟶
(rv2 s1 s2 kl b2 = Inl b1′ ⟶ k ∈ tobd_F12 s1 b1′ ‘ F1set2 (s1 b1′)) ∧
(rv2 s1 s2 kl b2 = Inr b2′ ⟶ k ∈ tobd_F22 s2 b2′ ‘ F2set2 (s2 b2′))) ∧
(kl @ [Inr k] ∈ Lev2 s1 s2 (Suc n) b2 ⟶
(rv2 s1 s2 kl b2 = Inl b1′ ⟶ k ∈ tobd_F13 s1 b1′ ‘ F1set3 (s1 b1′)) ∧
(rv2 s1 s2 kl b2 = Inr b2′ ⟶ k ∈ tobd_F23 s2 b2′ ‘ F2set3 (s2 b2′))))
⟨proof⟩

lemmas *Fset_image_Lev′* =
spec[*OF spec*[*OF spec*[*OF spec*[*OF spec*[*OF spec*[*OF Fset_image_Lev*]]]]]]
lemmas *F1set2_image_Lev1* =
mp[*OF conjunct1*[*OF mp*[*OF conjunct1*[*OF mp*[*OF conjunct1*[*OF Fset_image_Lev′*]]]]]]
lemmas *F1set2_image_Lev2* =
mp[*OF conjunct1*[*OF mp*[*OF conjunct1*[*OF mp*[*OF conjunct2*[*OF Fset_image_Lev′*]]]]]]
lemmas *F1set3_image_Lev1* =
mp[*OF conjunct1*[*OF mp*[*OF conjunct2*[*OF mp*[*OF conjunct1*[*OF Fset_image_Lev′*]]]]]]
lemmas *F1set3_image_Lev2* =
mp[*OF conjunct1*[*OF mp*[*OF conjunct2*[*OF mp*[*OF conjunct2*[*OF Fset_image_Lev′*]]]]]]
lemmas *F2set2_image_Lev1* =
mp[*OF conjunct2*[*OF mp*[*OF conjunct1*[*OF mp*[*OF conjunct1*[*OF Fset_image_Lev′*]]]]]]
lemmas *F2set2_image_Lev2* =
mp[*OF conjunct2*[*OF mp*[*OF conjunct1*[*OF mp*[*OF conjunct2*[*OF Fset_image_Lev′*]]]]]]
lemmas *F2set3_image_Lev1* =
mp[*OF conjunct2*[*OF mp*[*OF conjunct2*[*OF mp*[*OF conjunct1*[*OF Fset_image_Lev′*]]]]]]
lemmas *F2set3_image_Lev2* =
mp[*OF conjunct2*[*OF mp*[*OF conjunct2*[*OF mp*[*OF conjunct2*[*OF Fset_image_Lev′*]]]]]]

lemma *mor_beh*:
mor UNIV UNIV s1 s2 carT1 carT2 strT1 strT2 (*beh1 s1 s2*) (*beh2 s1 s2*)
⟨proof⟩

2.6 Quotient Coalgebra

abbreviation *car_final1* **where**
car_final1 ≡ *carT1* // (*lsbis1 carT1 carT2 strT1 strT2*)
abbreviation *car_final2* **where**
car_final2 ≡ *carT2* // (*lsbis2 carT1 carT2 strT1 strT2*)
abbreviation *str_final1* **where**
str_final1 ≡ *univ* (*F1map id*

(*Equiv_Relations.proj* (*lsbis1 carT1 carT2 strT1 strT2*))
(*Equiv_Relations.proj* (*lsbis2 carT1 carT2 strT1 strT2*)) *o strT1*)
abbreviation *str_final2* **where**
str_final2 ≡ *univ* (*F2map id*
(*Equiv_Relations.proj* (*lsbis1 carT1 carT2 strT1 strT2*))
(*Equiv_Relations.proj* (*lsbis2 carT1 carT2 strT1 strT2*)) *o strT2*)

lemma *congruent_strQ1*: *congruent* (*lsbis1 carT1 carT2 strT1 strT2* :: *'a carrier rel*)
(*F1map id* (*Equiv_Relations.proj* (*lsbis1 carT1 carT2 strT1 strT2* :: *'a carrier rel*))
(*Equiv_Relations.proj* (*lsbis2 carT1 carT2 strT1 strT2* :: *'a carrier rel*)) *o strT1*)
⟨*proof*⟩

lemma *congruent_strQ2*: *congruent* (*lsbis2 carT1 carT2 strT1 strT2* :: *'a carrier rel*)
(*F2map id* (*Equiv_Relations.proj* (*lsbis1 carT1 carT2 strT1 strT2* :: *'a carrier rel*))
(*Equiv_Relations.proj* (*lsbis2 carT1 carT2 strT1 strT2* :: *'a carrier rel*)) *o strT2*)
⟨*proof*⟩

lemma *coalg_final*:
coalg car_final1 car_final2 str_final1 str_final2
⟨*proof*⟩

lemma *mor_T_final*:
mor carT1 carT2 strT1 strT2 car_final1 car_final2 str_final1 str_final2
(*Equiv_Relations.proj* (*lsbis1 carT1 carT2 strT1 strT2*))
(*Equiv_Relations.proj* (*lsbis2 carT1 carT2 strT1 strT2*))
⟨*proof*⟩

lemmas *mor_final* = *mor_comp*[*OF mor_beh mor_T_final*]
lemmas *in_car_final1* = *mor_image1'*[*OF mor_final UNIV_I*]
lemmas *in_car_final2* = *mor_image2'*[*OF mor_final UNIV_I*]

typedef (**overloaded**) *'a JF1* = *car_final1* :: *'a carrier set set*
⟨*proof*⟩

typedef (**overloaded**) *'a JF2* = *car_final2* :: *'a carrier set set*
⟨*proof*⟩

definition *dtor1* **where**
dtor1 x = *F1map id Abs_JF1 Abs_JF2* (*str_final1* (*Rep_JF1 x*))
definition *dtor2* **where**
dtor2 x = *F2map id Abs_JF1 Abs_JF2* (*str_final2* (*Rep_JF2 x*))

lemma *mor_Rep_JF*: *mor UNIV UNIV dtor1 dtor2*
car_final1 car_final2 str_final1 str_final2
Rep_JF1 Rep_JF2
⟨*proof*⟩

lemma *mor_Abs_JF*: *mor car_final1 car_final2 str_final1 str_final2*
UNIV UNIV dtor1 dtor2 Abs_JF1 Abs_JF2
⟨*proof*⟩

definition *unfold1* **where**
unfold1 s1 s2 x =
Abs_JF1 ((*Equiv_Relations.proj* (*lsbis1 carT1 carT2 strT1 strT2*) *o beh1 s1 s2*) *x*)
definition *unfold2* **where**
unfold2 s1 s2 x =
Abs_JF2 ((*Equiv_Relations.proj* (*lsbis2 carT1 carT2 strT1 strT2*) *o beh2 s1 s2*) *x*)

lemma *mor_unfold*:

mor UNIV UNIV s1 s2 UNIV UNIV dtor1 dtor2 (*unfold1 s1 s2*) (*unfold2 s1 s2*)
⟨*proof*⟩

lemmas *unfold1* = *sym*[*OF morE1*[*OF mor_unfold UNIV_I*]]
lemmas *unfold2* = *sym*[*OF morE2*[*OF mor_unfold UNIV_I*]]

lemma *JF_cind*: *sbis UNIV UNIV dtor1 dtor2 R1 R2* $\Longrightarrow$ *R1* $\subseteq$ *Id* $\wedge$ *R2* $\subseteq$ *Id*
⟨*proof*⟩

lemmas *JF_cind1* = *conjunct1*[*OF JF_cind*]
lemmas *JF_cind2* = *conjunct2*[*OF JF_cind*]

lemma *unfold_unique_mor*:
mor UNIV UNIV s1 s2 UNIV UNIV dtor1 dtor2 f1 f2 $\Longrightarrow$
f1 = *unfold1 s1 s2* $\wedge$ *f2* = *unfold2 s1 s2*
⟨*proof*⟩

lemmas *unfold_unique* = *unfold_unique_mor*[*OF iffD2*[*OF mor_UNIV*], *OF conjI*]
lemmas *unfold1_dtor* = *sym*[*OF conjunct1*[*OF unfold_unique_mor*[*OF mor_id*]]]
lemmas *unfold2_dtor* = *sym*[*OF conjunct2*[*OF unfold_unique_mor*[*OF mor_id*]]]

lemmas *unfold1_o_dtor1* =
trans[*OF conjunct1*[*OF unfold_unique_mor*[*OF mor_comp*[*OF mor_str mor_unfold*]]] *unfold1_dtor*]
lemmas *unfold2_o_dtor2* =
trans[*OF conjunct2*[*OF unfold_unique_mor*[*OF mor_comp*[*OF mor_str mor_unfold*]]] *unfold2_dtor*]

definition *ctor1* **where** *ctor1* = *unfold1* (*F1map id dtor1 dtor2*) (*F2map id dtor1 dtor2*)
definition *ctor2* **where** *ctor2* = *unfold2* (*F1map id dtor1 dtor2*) (*F2map id dtor1 dtor2*)

lemma *ctor1_o_dtor1*:
ctor1 o dtor1 = *id*
⟨*proof*⟩

lemma *ctor2_o_dtor2*:
ctor2 o dtor2 = *id*
⟨*proof*⟩

lemma *dtor1_o_ctor1*:
dtor1 o ctor1 = *id*
⟨*proof*⟩

lemma *dtor2_o_ctor2*:
dtor2 o ctor2 = *id*
⟨*proof*⟩

lemmas *dtor1_ctor1* = *pointfree_idE*[*OF dtor1_o_ctor1*]
lemmas *dtor2_ctor2* = *pointfree_idE*[*OF dtor2_o_ctor2*]
lemmas *ctor1_dtor1* = *pointfree_idE*[*OF ctor1_o_dtor1*]
lemmas *ctor2_dtor2* = *pointfree_idE*[*OF ctor2_o_dtor2*]

lemmas *bij_dtor1* = *o_bij*[*OF ctor1_o_dtor1 dtor1_o_ctor1*]
lemmas *inj_dtor1* = *bij_is_inj*[*OF bij_dtor1*]
lemmas *surj_dtor1* = *bij_is_surj*[*OF bij_dtor1*]
lemmas *dtor1_nchotomy* = *surjD*[*OF surj_dtor1*]
lemmas *dtor1_diff* = *inj_eq*[*OF inj_dtor1*]
lemmas *dtor1_cases* = *exE*[*OF dtor1_nchotomy*]
lemmas *bij_dtor2* = *o_bij*[*OF ctor2_o_dtor2 dtor2_o_ctor2*]
lemmas *inj_dtor2* = *bij_is_inj*[*OF bij_dtor2*]
lemmas *surj_dtor2* = *bij_is_surj*[*OF bij_dtor2*]
lemmas *dtor2_nchotomy* = *surjD*[*OF surj_dtor2*]
lemmas *dtor2_diff* = *inj_eq*[*OF inj_dtor2*]
lemmas *dtor2_cases* = *exE*[*OF dtor2_nchotomy*]

lemmas *bij_ctor1* = *o_bij*[*OF dtor1_o_ctor1 ctor1_o_dtor1*]
lemmas *inj_ctor1* = *bij_is_inj*[*OF bij_ctor1*]
lemmas *surj_ctor1* = *bij_is_surj*[*OF bij_ctor1*]
lemmas *ctor1_nchotomy* = *surjD*[*OF surj_ctor1*]
lemmas *ctor1_diff* = *inj_eq*[*OF inj_ctor1*]
lemmas *ctor1_cases* = *exE*[*OF ctor1_nchotomy*]
lemmas *bij_ctor2* = *o_bij*[*OF dtor2_o_ctor2 ctor2_o_dtor2*]
lemmas *inj_ctor2* = *bij_is_inj*[*OF bij_ctor2*]
lemmas *surj_ctor2* = *bij_is_surj*[*OF bij_ctor2*]
lemmas *ctor2_nchotomy* = *surjD*[*OF surj_ctor2*]
lemmas *ctor2_diff* = *inj_eq*[*OF inj_ctor2*]
lemmas *ctor2_cases* = *exE*[*OF ctor2_nchotomy*]

lemmas *ctor1_unfold1* = *iffD1*[*OF dtor1_diff trans*[*OF unfold1 sym*[*OF dtor1_ctor1*]]]
lemmas *ctor2_unfold2* = *iffD1*[*OF dtor2_diff trans*[*OF unfold2 sym*[*OF dtor2_ctor2*]]]

definition *corec1* **where** *corec1 s1 s2* =
  unfold1 (*case_sum* (*F1map id Inl Inl o dtor1*) *s1*)
    (*case_sum* (*F2map id Inl Inl o dtor2*) *s2*) *o Inr*
definition *corec2* **where** *corec2 s1 s2* =
  unfold2 (*case_sum* (*F1map id Inl Inl o dtor1*) *s1*)
    (*case_sum* (*F2map id Inl Inl o dtor2*) *s2*) *o Inr*

lemma *dtor1_o_unfold1*: *dtor1 o unfold1 s1 s2* = *F1map id* (*unfold1 s1 s2*) (*unfold2 s1 s2*) *o s1*
  ⟨*proof*⟩
lemma *dtor2_o_unfold2*: *dtor2 o unfold2 s1 s2* = *F2map id* (*unfold1 s1 s2*) (*unfold2 s1 s2*) *o s2*
  ⟨*proof*⟩

lemma *corec1_Inl_sum*:
  unfold1 (*case_sum* (*F1map id Inl Inl* ∘ *dtor1*) *s1*) (*case_sum* (*F2map id Inl Inl* ∘ *dtor2*) *s2*) ∘ *Inl* = *id*
  ⟨*proof*⟩

lemma *corec2_Inl_sum*:
  unfold2 (*case_sum* (*F1map id Inl Inl* ∘ *dtor1*) *s1*) (*case_sum* (*F2map id Inl Inl* ∘ *dtor2*) *s2*) ∘ *Inl* = *id*
  ⟨*proof*⟩

lemma *case_sum_expand_Inr*: *f o Inl* = *g* ⟹ *case_sum g* (*f o Inr*) = *f*
  ⟨*proof*⟩

theorem *corec1*:
  dtor1 (*corec1 s1 s2 a*) =
    F1map id (*case_sum id* (*corec1 s1 s2*)) (*case_sum id* (*corec2 s1 s2*)) (*s1 a*)
  ⟨*proof*⟩

theorem *corec2*:
  dtor2 (*corec2 s1 s2 a*) =
    F2map id (*case_sum id* (*corec1 s1 s2*)) (*case_sum id* (*corec2 s1 s2*)) (*s2 a*)
  ⟨*proof*⟩

lemma *corec_unique*:
  F1map id (*case_sum id f1*) (*case_sum id f2*) ∘ *s1* = *dtor1* ∘ *f1* ⟹
   F2map id (*case_sum id f1*) (*case_sum id f2*) ∘ *s2* = *dtor2* ∘ *f2* ⟹
   f1 = *corec1 s1 s2* ∧ *f2* = *corec2 s1 s2*
  ⟨*proof*⟩

2.7 Coinduction

lemma *Frel_coind*:
  ⟦∀ *a b*. *phi1 a b* ⟶ *F1rel* (=) *phi1 phi2* (*dtor1 a*) (*dtor1 b*);
   ∀ *a b*. *phi2 a b* ⟶ *F2rel* (=) *phi1 phi2* (*dtor2 a*) (*dtor2 b*)⟧ ⟹
  (*phi1 a1 b1* ⟶ *a1* = *b1*) ∧ (*phi2 a2 b2* ⟶ *a2* = *b2*)

⟨*proof*⟩

2.8 The Result as an BNF

abbreviation *JF1map* **where**
  $JF1map\ u \equiv unfold1\ (F1map\ u\ id\ id\ o\ dtor1)\ (F2map\ u\ id\ id\ o\ dtor2)$
abbreviation *JF2map* **where**
  $JF2map\ u \equiv unfold2\ (F1map\ u\ id\ id\ o\ dtor1)\ (F2map\ u\ id\ id\ o\ dtor2)$

lemma *JF1map*: $dtor1\ o\ JF1map\ u = F1map\ u\ (JF1map\ u)\ (JF2map\ u)\ o\ dtor1$
  ⟨*proof*⟩

lemma *JF2map*: $dtor2\ o\ JF2map\ u = F2map\ u\ (JF1map\ u)\ (JF2map\ u)\ o\ dtor2$
  ⟨*proof*⟩

lemmas $JF1map\_simps = o\_eq\_dest[OF\ JF1map]$
lemmas $JF2map\_simps = o\_eq\_dest[OF\ JF2map]$

theorem *JF1map_id*: $JF1map\ id = id$
  ⟨*proof*⟩

theorem *JF2map_id*: $JF2map\ id = id$
  ⟨*proof*⟩

lemma *JFmap_unique*:
  $[\![dtor1\ o\ u = F1map\ f\ u\ v\ o\ dtor1;\ dtor2\ o\ v = F2map\ f\ u\ v\ o\ dtor2]\!] \Longrightarrow$
    $u = JF1map\ f \wedge v = JF2map\ f$
  ⟨*proof*⟩

theorem *JF1map_comp*: $JF1map\ (g\ o\ f) = JF1map\ g\ o\ JF1map\ f$
  ⟨*proof*⟩

theorem *JF2map_comp*: $JF2map\ (g\ o\ f) = JF2map\ g\ o\ JF2map\ f$
  ⟨*proof*⟩

definition *JFcol* **where**
  $JFcol = rec\_nat\ (\%a.\ \{\},\ \%b.\ \{\})$
    $(\%n\ rec.$
    $(\%a.\ F1set1\ (dtor1\ a) \cup$
      $((\bigcup a' \in F1set2\ (dtor1\ a).\ fst\ rec\ a') \cup$
      $(\bigcup a' \in F1set3\ (dtor1\ a).\ snd\ rec\ a')),$
    $\%b.\ F2set1\ (dtor2\ b) \cup$
      $((\bigcup b' \in F2set2\ (dtor2\ b).\ fst\ rec\ b') \cup$
      $(\bigcup b' \in F2set3\ (dtor2\ b).\ snd\ rec\ b'))))$

abbreviation *JF1col* **where** $JF1col\ n \equiv fst\ (JFcol\ n)$
abbreviation *JF2col* **where** $JF2col\ n \equiv snd\ (JFcol\ n)$

lemmas $JF1col\_0 = fun\_cong[OF\ fstI[OF\ rec\_nat\_0\_imp[OF\ JFcol\_def]]]$
lemmas $JF2col\_0 = fun\_cong[OF\ sndI[OF\ rec\_nat\_0\_imp[OF\ JFcol\_def]]]$
lemmas $JF1col\_Suc = fun\_cong[OF\ fstI[OF\ rec\_nat\_Suc\_imp[OF\ JFcol\_def]]]$
lemmas $JF2col\_Suc = fun\_cong[OF\ sndI[OF\ rec\_nat\_Suc\_imp[OF\ JFcol\_def]]]$

lemma *JFcol_minimal*:
  $[\![\bigwedge a.\ F1set1\ (dtor1\ a) \subseteq K1\ a;$
    $\bigwedge b.\ F2set1\ (dtor2\ b) \subseteq K2\ b;$
    $\bigwedge a\ a'.\ a' \in F1set2\ (dtor1\ a) \Longrightarrow K1\ a' \subseteq K1\ a;$
    $\bigwedge a\ b'.\ b' \in F1set3\ (dtor1\ a) \Longrightarrow K2\ b' \subseteq K1\ a;$
    $\bigwedge b\ a'.\ a' \in F2set2\ (dtor2\ b) \Longrightarrow K1\ a' \subseteq K2\ b;$
    $\bigwedge b\ b'.\ b' \in F2set3\ (dtor2\ b) \Longrightarrow K2\ b' \subseteq K2\ b]\!] \Longrightarrow$
  $\forall a\ b.\ JF1col\ n\ a \subseteq K1\ a \wedge JF2col\ n\ b \subseteq K2\ b$
  ⟨*proof*⟩

lemma *JFset_minimal*:
$[\![\bigwedge a.\ F1set1\ (dtor1\ a) \subseteq K1\ a;$
$\bigwedge b.\ F2set1\ (dtor2\ b) \subseteq K2\ b;$
$\bigwedge a\ a'.\ a' \in F1set2\ (dtor1\ a) \Longrightarrow K1\ a' \subseteq K1\ a;$
$\bigwedge a\ b'.\ b' \in F1set3\ (dtor1\ a) \Longrightarrow K2\ b' \subseteq K1\ a;$
$\bigwedge b\ a'.\ a' \in F2set2\ (dtor2\ b) \Longrightarrow K1\ a' \subseteq K2\ b;$
$\bigwedge b\ b'.\ b' \in F2set3\ (dtor2\ b) \Longrightarrow K2\ b' \subseteq K2\ b]\!] \Longrightarrow$
$(\bigcup n.\ JF1col\ n\ a) \subseteq K1\ a \wedge (\bigcup n.\ JF2col\ n\ b) \subseteq K2\ b$
$\langle proof \rangle$

abbreviation *JF1set* **where** $JF1set\ a \equiv (\bigcup n.\ JF1col\ n\ a)$
abbreviation *JF2set* **where** $JF2set\ a \equiv (\bigcup n.\ JF2col\ n\ a)$

lemma *F1set1_incl_JF1set*:
$F1set1\ (dtor1\ a) \subseteq JF1set\ a$
$\langle proof \rangle$

lemma *F2set1_incl_JF2set*:
$F2set1\ (dtor2\ a) \subseteq JF2set\ a$
$\langle proof \rangle$

lemma *F1set2_JF1set_incl_JF1set*:
$a' \in F1set2\ (dtor1\ a) \Longrightarrow JF1set\ a' \subseteq JF1set\ a$
$\langle proof \rangle$

lemma *F1set3_JF2set_incl_JF1set*:
$a' \in F1set3\ (dtor1\ a) \Longrightarrow JF2set\ a' \subseteq JF1set\ a$
$\langle proof \rangle$

lemma *F2set2_JF1set_incl_JF2set*:
$a' \in F2set2\ (dtor2\ a) \Longrightarrow JF1set\ a' \subseteq JF2set\ a$
$\langle proof \rangle$

lemma *F2set3_JF2set_incl_JF2set*:
$a' \in F2set3\ (dtor2\ a) \Longrightarrow JF2set\ a' \subseteq JF2set\ a$
$\langle proof \rangle$

lemmas *F1set1_JF1set* = *subsetD*[*OF F1set1_incl_JF1set*]
lemmas *F2set1_JF2set* = *subsetD*[*OF F2set1_incl_JF2set*]
lemmas *F1set2_JF1set_JF1set* = *subsetD*[*OF F1set2_JF1set_incl_JF1set*]
lemmas *F1set3_JF2set_JF1set* = *subsetD*[*OF F1set3_JF2set_incl_JF1set*]
lemmas *F2set2_JF1set_JF2set* = *subsetD*[*OF F2set2_JF1set_incl_JF2set*]
lemmas *F2set3_JF2set_JF2set* = *subsetD*[*OF F2set3_JF2set_incl_JF2set*]

lemma *JFset_le*:
fixes $a :: 'a\ JF1$ **and** $b :: 'a\ JF2$
shows
$JF1set\ a \subseteq F1set1\ (dtor1\ a) \cup (\bigcup\ (JF1set\ `\ F1set2\ (dtor1\ a)) \cup \bigcup\ (JF2set\ `\ F1set3\ (dtor1\ a))) \wedge$
$JF2set\ b \subseteq F2set1\ (dtor2\ b) \cup (\bigcup\ (JF1set\ `\ F2set2\ (dtor2\ b)) \cup \bigcup\ (JF2set\ `\ F2set3\ (dtor2\ b)))$
$\langle proof \rangle$

theorem *JF1set_simps*:
$JF1set\ a = F1set1\ (dtor1\ a) \cup$
$((\bigcup\ b \in F1set2\ (dtor1\ a).\ JF1set\ b) \cup$
$(\bigcup\ b \in F1set3\ (dtor1\ a).\ JF2set\ b))$
$\langle proof \rangle$

theorem *JF2set_simps*:
$JF2set\ a = F2set1\ (dtor2\ a) \cup$
$((\bigcup\ b \in F2set2\ (dtor2\ a).\ JF1set\ b) \cup$
$(\bigcup\ b \in F2set3\ (dtor2\ a).\ JF2set\ b))$
$\langle proof \rangle$

lemma *JFcol_natural*:
  $\forall b1\ b2.\ u\ `\ (JF1col\ n\ b1) = JF1col\ n\ (JF1map\ u\ b1) \land$
        $u\ `\ (JF2col\ n\ b2) = JF2col\ n\ (JF2map\ u\ b2)$
  ⟨*proof*⟩

theorem *JF1set_natural*: $JF1set\ o\ (JF1map\ u) = image\ u\ o\ JF1set$
  ⟨*proof*⟩

theorem *JF2set_natural*: $JF2set\ o\ (JF2map\ u) = image\ u\ o\ JF2set$
  ⟨*proof*⟩

theorem *JFmap_cong0*:
  $((\forall p \in JF1set\ a.\ u\ p = v\ p) \longrightarrow JF1map\ u\ a = JF1map\ v\ a) \land$
  $((\forall p \in JF2set\ b.\ u\ p = v\ p) \longrightarrow JF2map\ u\ b = JF2map\ v\ b)$
  ⟨*proof*⟩

lemmas *JF1map_cong0* = *mp*[*OF conjunct1*[*OF JFmap_cong0*]]
lemmas *JF2map_cong0* = *mp*[*OF conjunct2*[*OF JFmap_cong0*]]

lemma *JFcol_bd*: $\forall (j1 :: 'a\ JF1)\ (j2 :: 'a\ JF2).\ |JF1col\ n\ j1| <o\ bd\_F \land |JF2col\ n\ j2| <o\ bd\_F$
  ⟨*proof*⟩

theorem *JF1set_bd*: $|JF1set\ j| <o\ bd\_F$
  ⟨*proof*⟩

theorem *JF2set_bd*: $|JF2set\ j| <o\ bd\_F$
  ⟨*proof*⟩

abbreviation *JF2wit* ≡ *ctor2 wit_F2*

theorem *JF2wit*: $\bigwedge x.\ x \in JF2set\ JF2wit \Longrightarrow False$
  ⟨*proof*⟩

abbreviation *JF1wit* ≡ (%*a. ctor1* (*wit2_F1 a JF2wit*))

theorem *JF1wit*: $\bigwedge x.\ x \in JF1set\ (JF1wit\ a) \Longrightarrow x = a$
  ⟨*proof*⟩

context
  fixes $phi1 :: 'a \Rightarrow 'a\ JF1 \Rightarrow bool$ **and** $phi2 :: 'a \Rightarrow 'a\ JF2 \Rightarrow bool$
begin

lemmas *JFset_induct* =
  JFset_minimal[*of* %*b1*. {*a* ∈ *JF1set b1* . *phi1 a b1*} %*b2*. {*a* ∈ *JF2set b2* . *phi2 a b2*},
    unfolded subset_Collect_iff[*OF F1set1_incl_JF1set*] *subset_Collect_iff*[*OF F2set1_incl_JF2set*]
    subset_Collect_iff[*OF subset_refl*],
    OF ballI ballI
    subset_CollectI[*OF F1set2_JF1set_incl_JF1set*]
    subset_CollectI[*OF F1set3_JF2set_incl_JF1set*]
    subset_CollectI[*OF F2set2_JF1set_incl_JF2set*]
    subset_CollectI[*OF F2set3_JF2set_incl_JF2set*]]

end

⟨*ML*⟩

abbreviation *JF1in* **where** *JF1in B* ≡ {*a. JF1set a* ⊆ *B*}
abbreviation *JF2in* **where** *JF2in B* ≡ {*a. JF2set a* ⊆ *B*}

definition *JF1rel* **where**
$JF1rel\ R = (BNF\_Def.Grp\ (JF1in\ (Collect\ (case\_prod\ R)))\ (JF1map\ fst))\hat{}{-}{-}1\ OO$
$(BNF\_Def.Grp\ (JF1in\ (Collect\ (case\_prod\ R)))\ (JF1map\ snd))$

definition *JF2rel* **where**
$JF2rel\ R = (BNF\_Def.Grp\ (JF2in\ (Collect\ (case\_prod\ R)))\ (JF2map\ fst))\hat{}{-}{-}1\ OO$
$(BNF\_Def.Grp\ (JF2in\ (Collect\ (case\_prod\ R)))\ (JF2map\ snd))$

lemma *in_JF1rel*:
$JF1rel\ R\ x\ y \longleftrightarrow (\exists\ z.\ z \in JF1in\ (Collect\ (case\_prod\ R)) \land JF1map\ fst\ z = x \land JF1map\ snd\ z = y)$
$\langle proof \rangle$

lemma *in_JF2rel*:
$JF2rel\ R\ x\ y \longleftrightarrow (\exists\ z.\ z \in JF2in\ (Collect\ (case\_prod\ R)) \land JF2map\ fst\ z = x \land JF2map\ snd\ z = y)$
$\langle proof \rangle$

lemma *J_rel_coind_ind*:
$[\![\forall x\ y.\ R2\ x\ y \longrightarrow (f\ x\ y \in F1in\ (Collect\ (case\_prod\ R1))\ (Collect\ (case\_prod\ R2))\ (Collect\ (case\_prod\ R3)) \land$
$F1map\ fst\ fst\ fst\ (f\ x\ y) = dtor1\ x \land$
$F1map\ snd\ snd\ snd\ (f\ x\ y) = dtor1\ y);$
$\forall x\ y.\ R3\ x\ y \longrightarrow (g\ x\ y \in F2in\ (Collect\ (case\_prod\ R1))\ (Collect\ (case\_prod\ R2))\ (Collect\ (case\_prod\ R3)) \land$
$F2map\ fst\ fst\ fst\ (g\ x\ y) = dtor2\ x \land$
$F2map\ snd\ snd\ snd\ (g\ x\ y) = dtor2\ y)]\!] \Longrightarrow$
$(\forall a{\in}JF1set\ z1.\ \forall x\ y.\ R2\ x\ y \land z1 = unfold1\ (case\_prod\ f)\ (case\_prod\ g)\ (x,\ y) \longrightarrow R1\ (fst\ a)\ (snd\ a)) \land$
$(\forall a{\in}JF2set\ z2.\ \forall x\ y.\ R3\ x\ y \land z2 = unfold2\ (case\_prod\ f)\ (case\_prod\ g)\ (x,\ y) \longrightarrow R1\ (fst\ a)\ (snd\ a))$
$\langle proof \rangle$

lemma *J_rel_coind_coind1*:
$[\![\forall x\ y.\ R2\ x\ y \longrightarrow (f\ x\ y \in F1in\ (Collect\ (case\_prod\ R1))\ (Collect\ (case\_prod\ R2))\ (Collect\ (case\_prod\ R3)) \land$
$F1map\ fst\ fst\ fst\ (f\ x\ y) = dtor1\ x \land$
$F1map\ snd\ snd\ snd\ (f\ x\ y) = dtor1\ y);$
$\forall x\ y.\ R3\ x\ y \longrightarrow (g\ x\ y \in F2in\ (Collect\ (case\_prod\ R1))\ (Collect\ (case\_prod\ R2))\ (Collect\ (case\_prod\ R3)) \land$
$F2map\ fst\ fst\ fst\ (g\ x\ y) = dtor2\ x \land$
$F2map\ snd\ snd\ snd\ (g\ x\ y) = dtor2\ y)]\!] \Longrightarrow$
$((\exists y.\ R2\ x1\ y \land x1' = JF1map\ fst\ (unfold1\ (case\_prod\ f)\ (case\_prod\ g)\ (x1,\ y))) \longrightarrow x1' = x1) \land$
$((\exists y.\ R3\ x2\ y \land x2' = JF2map\ fst\ (unfold2\ (case\_prod\ f)\ (case\_prod\ g)\ (x2,\ y))) \longrightarrow x2' = x2)$
$\langle proof \rangle$

lemma *J_rel_coind_coind2*:
$[\![\forall x\ y.\ R2\ x\ y \longrightarrow (f\ x\ y \in F1in\ (Collect\ (case\_prod\ R1))\ (Collect\ (case\_prod\ R2))\ (Collect\ (case\_prod\ R3)) \land$
$F1map\ fst\ fst\ fst\ (f\ x\ y) = dtor1\ x \land$
$F1map\ snd\ snd\ snd\ (f\ x\ y) = dtor1\ y);$
$\forall x\ y.\ R3\ x\ y \longrightarrow (g\ x\ y \in F2in\ (Collect\ (case\_prod\ R1))\ (Collect\ (case\_prod\ R2))\ (Collect\ (case\_prod\ R3)) \land$
$F2map\ fst\ fst\ fst\ (g\ x\ y) = dtor2\ x \land$
$F2map\ snd\ snd\ snd\ (g\ x\ y) = dtor2\ y)]\!] \Longrightarrow$
$((\exists x.\ R2\ x\ y1 \land y1' = JF1map\ snd\ (unfold1\ (case\_prod\ f)\ (case\_prod\ g)\ (x,\ y1))) \longrightarrow y1' = y1) \land$
$((\exists x.\ R3\ x\ y2 \land y2' = JF2map\ snd\ (unfold2\ (case\_prod\ f)\ (case\_prod\ g)\ (x,\ y2))) \longrightarrow y2' = y2)$
$\langle proof \rangle$

lemma *J_rel_coind*:
assumes $CIH1$: $\forall x2\ y2.\ R2\ x2\ y2 \longrightarrow F1rel\ R1\ R2\ R3\ (dtor1\ x2)\ (dtor1\ y2)$
and $CIH2$: $\forall x3\ y3.\ R3\ x3\ y3 \longrightarrow F2rel\ R1\ R2\ R3\ (dtor2\ x3)\ (dtor2\ y3)$
shows $R2 \leq JF1rel\ R1 \land R3 \leq JF2rel\ R1$
$\langle proof \rangle$

lemma *JF1rel_F1rel*: $JF1rel\ R\ a\ b \longleftrightarrow F1rel\ R\ (JF1rel\ R)\ (JF2rel\ R)\ (dtor1\ a)\ (dtor1\ b)$
$\langle proof \rangle$

lemma *JF2rel_F2rel*: $JF2rel\ R\ a\ b \longleftrightarrow F2rel\ R\ (JF1rel\ R)\ (JF2rel\ R)\ (dtor2\ a)\ (dtor2\ b)$
$\langle proof \rangle$

lemma *JFrel_Comp_le*:
JF1rel R1 OO JF1rel R2 ≤ *JF1rel* (*R1 OO R2*) ∧ *JF2rel R1 OO JF2rel R2* ≤ *JF2rel* (*R1 OO R2*)
⟨*proof*⟩

context includes *lifting_syntax*
begin

lemma *unfold_transfer*:
((*S* ===> *F1rel R S T*) ===> (*T* ===> *F2rel R S T*) ===> *S* ===> *JF1rel R*) *unfold1 unfold1* ∧
((*S* ===> *F1rel R S T*) ===> (*T* ===> *F2rel R S T*) ===> *T* ===> *JF2rel R*) *unfold2 unfold2*
⟨*proof*⟩

end

⟨*ML*⟩

bnf *'a JF1*
map: *JF1map*
sets: *JF1set*
bd: *bd_F*
wits: *JF1wit*
rel: *JF1rel*
⟨*proof*⟩

bnf *'a JF2*
map: *JF2map*
sets: *JF2set*
bd: *bd_F*
wits: *JF2wit*
rel: *JF2rel*
⟨*proof*⟩

3 Normalized Composition of BNFs

Expected normal form: outer m-ary BNF is composed with m inner n-ary BNFs.

unbundle *cardinal_syntax*

declare [[*bnf_internals*]]
bnf-axiomatization (*dead 'p1*, *F1set1*: *'a1*, *F1set2*: *'a2*) *F1*
[*wits*: (*'p1*, *'a1*, *'a2*) *F1*]
for *map*: *F1map rel*: *F1rel*
bnf-axiomatization (*dead 'p2*, *F2set1*: *'a1*, *F2set2*: *'a2*) *F2*
[*wits*: *'a1* ⇒ (*'p2*, *'a1*, *'a2*) *F2 'a2* ⇒ (*'p2*, *'a1*, *'a2*) *F2*]
for *map*: *F2map rel*: *F2rel*
bnf-axiomatization (*dead 'p3*, *F3set1*: *'a1*, *F3set2*: *'a2*) *F3*
[*wits*: *'a1* ⇒ *'a2* ⇒ (*'p3*, *'a1*, *'a2*) *F3*]
for *map*: *F3map rel*: *F3rel*
bnf-axiomatization (*dead 'p*, *Gset1*: *'b1*, *Gset2*: *'b2*, *Gset3*: *'b3*) *G*
[*wits*: *'b1* ⇒ *'b3* ⇒ (*'p*, *'b1*, *'b2*, *'b3*) *G 'b2* ⇒ *'b3* ⇒ (*'p*, *'b1*, *'b2*, *'b3*) *G*]
for *map*: *Gmap rel*: *Grel*
type-synonym (*'p1*, *'p2*, *'p3*, *'p*, *'a1*, *'a2*) *H* =
(*'p*, (*'p1*, *'a1*, *'a2*) *F1*, (*'p2*, *'a1*, *'a2*) *F2*, (*'p3*, *'a1*, *'a2*) *F3*) *G*
type-synonym (*'p1*, *'p2*, *'p3*, *'p*) *Hbd_type* =
(*'p1 bd_type_F1* + *'p2 bd_type_F2* + *'p3 bd_type_F3*) × *'p bd_type_G*

abbreviation *F1in* **where** *F1in A1 A2* ≡ {*x*. *F1set1 x* ⊆ *A1* ∧ *F1set2 x* ⊆ *A2*}
abbreviation *F2in* **where** *F2in A1 A2* ≡ {*x*. *F2set1 x* ⊆ *A1* ∧ *F2set2 x* ⊆ *A2*}
abbreviation *F3in* **where** *F3in A1 A2* ≡ {*x*. *F3set1 x* ⊆ *A1* ∧ *F3set2 x* ⊆ *A2*}
abbreviation *Gin* **where** *Gin A1 A2 A3* ≡ {*x*. *Gset1 x* ⊆ *A1* ∧ *Gset2 x* ⊆ *A2* ∧ *Gset3 x* ⊆ *A3*}

abbreviation *Gset* **where**
$Gset \equiv BNF\_Def.collect \ \{Gset1,\ Gset2,\ Gset3\}$

abbreviation $Hmap :: ('a1 \Rightarrow 'b1) \Rightarrow ('a2 \Rightarrow 'b2) \Rightarrow$
$('p1,\ 'p2,\ 'p3,\ 'p,\ 'a1,\ 'a2)\ H \Rightarrow ('p1,\ 'p2,\ 'p3,\ 'p,\ 'b1,\ 'b2)\ H$ **where**
$Hmap\ f\ g \equiv Gmap\ (F1map\ f\ g)\ (F2map\ f\ g)\ (F3map\ f\ g)$

abbreviation $Hset1 :: ('p1,\ 'p2,\ 'p3,\ 'p,\ 'a1,\ 'a2)\ H \Rightarrow 'a1\ set$ **where**
$Hset1 \equiv Union\ o\ Gset\ o\ Gmap\ F1set1\ F2set1\ F3set1$

abbreviation $Hset2 :: ('p1,\ 'p2,\ 'p3,\ 'p,\ 'a1,\ 'a2)\ H \Rightarrow 'a2\ set$ **where**
$Hset2 \equiv Union\ o\ Gset\ o\ Gmap\ F1set2\ F2set2\ F3set2$

lemma *Hset1_alt*:
$Hset1 = Union\ o\ BNF\_Def.collect\ \{image\ F1set1\ o\ Gset1,\ image\ F2set1\ o\ Gset2,\ image\ F3set1\ o\ Gset3\}$
$\langle proof \rangle$

lemma *Hset2_alt*:
$Hset2 = Union\ o\ BNF\_Def.collect\ \{image\ F1set2\ o\ Gset1,\ image\ F2set2\ o\ Gset2,\ image\ F3set2\ o\ Gset3\}$
$\langle proof \rangle$

abbreviation *Hbd* **where**
$Hbd \equiv (bd\_F1\ +c\ bd\_F2\ +c\ bd\_F3)\ *c\ bd\_G$

theorem *Hmap_id*: $Hmap\ id\ id = id$
$\langle proof \rangle$

theorem *Hmap_comp*: $Hmap\ (f1\ o\ g1)\ (f2\ o\ g2) = Hmap\ f1\ f2\ o\ Hmap\ g1\ g2$
$\langle proof \rangle$

theorem *Hmap_cong*: $[\![\bigwedge z.\ z \in Hset1\ x \Longrightarrow f1\ z = g1\ z;\ \bigwedge z.\ z \in Hset2\ x \Longrightarrow f2\ z = g2\ z]\!] \Longrightarrow$
$Hmap\ f1\ f2\ x = Hmap\ g1\ g2\ x$
$\langle proof \rangle$

theorem *Hset1_natural*: $Hset1\ o\ Hmap\ f1\ f2 = image\ f1\ o\ Hset1$
$\langle proof \rangle$

theorem *Hset2_natural*: $Hset2\ o\ Hmap\ f1\ f2 = image\ f2\ o\ Hset2$
$\langle proof \rangle$

theorem *Hbd_card_order*: $card\_order\ Hbd$
$\langle proof \rangle$

theorem *Hbd_cinfinite*: $cinfinite\ Hbd$
$\langle proof \rangle$

theorem *Hbd_regularCard*: $regularCard\ Hbd$
$\langle proof \rangle$

theorem *Hset1_bd*: $|Hset1\ (x :: ('p1,\ 'p2,\ 'p3,\ 'p,\ 'a1,\ 'a2)\ H\ )| <o$
$(Hbd :: ('p1,\ 'p2,\ 'p3,\ 'p)\ Hbd\_type\ rel)$
$\langle proof \rangle$

theorem *Hset2_bd*: $|Hset2\ (x :: ('p1,\ 'p2,\ 'p3,\ 'p,\ 'a1,\ 'a2)\ H\ )| <o$
$(Hbd :: ('p1,\ 'p2,\ 'p3,\ 'p)\ Hbd\_type\ rel)$
$\langle proof \rangle$

abbreviation *Hin* **where** $Hin\ A1\ A2 \equiv \{x.\ Hset1\ x \subseteq A1 \wedge Hset2\ x \subseteq A2\}$

lemma *Hin_alt*: $Hin\ A1\ A2 = Gin\ (F1in\ A1\ A2)\ (F2in\ A1\ A2)\ (F3in\ A1\ A2)$
$\langle proof \rangle$

definition *Hwit1* **where** $Hwit1\ b\ c = wit1\_G\ wit\_F1\ (wit\_F3\ b\ c)$

definition *Hwit21* **where** $Hwit21\ b\ c = wit2\_G\ (wit1\_F2\ b)\ (wit\_F3\ b\ c)$
definition *Hwit22* **where** $Hwit22\ b\ c = wit2\_G\ (wit2\_F2\ c)\ (wit\_F3\ b\ c)$

lemma *Hwit1*:
$\bigwedge x.\ x \in Hset1\ (Hwit1\ b\ c) \Longrightarrow x = b$
$\bigwedge x.\ x \in Hset2\ (Hwit1\ b\ c) \Longrightarrow x = c$
⟨*proof*⟩

lemma *Hwit21*:
$\bigwedge x.\ x \in Hset1\ (Hwit21\ b\ c) \Longrightarrow x = b$
$\bigwedge x.\ x \in Hset2\ (Hwit21\ b\ c) \Longrightarrow x = c$
⟨*proof*⟩

lemma *Hwit22*:
$\bigwedge x.\ x \in Hset1\ (Hwit22\ b\ c) \Longrightarrow x = b$
$\bigwedge x.\ x \in Hset2\ (Hwit22\ b\ c) \Longrightarrow x = c$
⟨*proof*⟩

lemma *Grel_cong*: $[\![R1 = S1;\ R2 = S2;\ R3 = S3]\!] \Longrightarrow Grel\ R1\ R2\ R3 = Grel\ S1\ S2\ S3$
⟨*proof*⟩

definition *Hrel* **where**
Hrel R1 R2 = (BNF_Def.Grp (Hin (Collect (case_prod R1)) (Collect (case_prod R2))) (Hmap fst fst))^−−1 OO (BNF_Def.Grp (Hin (Collect (case_prod R1)) (Collect (case_prod R2))) (Hmap snd snd))

lemmas *Hrel_unfold = trans[OF Hrel_def trans[OF OO_Grp_cong[OF Hin_alt] trans[OF arg_cong2[of _ _ _ _ relcompp, OF trans[OF arg_cong[of _ _ conversep, OF sym[OF G.rel_Grp]] G.rel_conversep[symmetric]] sym[OF G.rel_Grp]] trans[OF G.rel_compp[symmetric] Grel_cong[OF sym[OF F1.rel_compp_Grp] sym[OF F2.rel_compp_Grp] sym[OF F3.rel_compp_Grp]]]]]]*

bnf *H*: *('p1, 'p2, 'p3, 'p, 'a1, 'a2) H*
map: *Hmap*
sets: *Hset1 Hset2*
bd: *Hbd :: ('p1, 'p2, 'p3, 'p) Hbd_type rel*
rel: *Hrel*
⟨*proof*⟩

4 Removing Live Variables

unbundle *cardinal_syntax*

declare [[*bnf_internals*]]
bnf-axiomatization (*dead 'p, Fset1*: *'a1, Fset2*: *'a2, Fset3*: *'a3*) *F* **for** *map*: *Fmap rel*: *Frel*

abbreviation *F1map* :: $('a2 \Rightarrow 'b2) \Rightarrow ('a3 \Rightarrow 'b3) \Rightarrow ('p, 'a1, 'a2, 'a3)\ F \Rightarrow ('p, 'a1, 'b2, 'b3)\ F$ **where**
F1map ≡ *Fmap id*
abbreviation *F2map* :: $('a3 \Rightarrow 'b3) \Rightarrow ('p, 'a1, 'a2, 'a3)\ F \Rightarrow ('p, 'a1, 'a2, 'b3)\ F$ **where**
F2map ≡ *Fmap id id*

abbreviation *F1set1* ≡ *Fset2*
abbreviation *F1set2* ≡ *Fset3*
abbreviation *F2set* ≡ *Fset3*

theorem *F1map_id*: *F1map id id = id*
⟨*proof*⟩

theorem *F2map_id*: *F2map id = id*

⟨proof⟩

theorem *F1map_comp*: *F1map (f1 o g1) (f2 o g2) = F1map f1 f2 o F1map g1 g2*
⟨proof⟩

theorem *F2map_comp*: *F2map (f o g) = F2map f o F2map g*
⟨proof⟩

theorem *F1map_cong*: $[\![\bigwedge z.\ z \in F1set1\ x \Longrightarrow f1\ z = g1\ z; \bigwedge z.\ z \in F1set2\ x \Longrightarrow f2\ z = g2\ z]\!]$
$\Longrightarrow F1map\ f1\ f2\ x = F1map\ g1\ g2\ x$
⟨proof⟩

theorem *F2map_cong*: $[\![\bigwedge z.\ z \in F2set\ x \Longrightarrow f\ z = g\ z]\!] \Longrightarrow F2map\ f\ x = F2map\ g\ x$
⟨proof⟩

theorem *F1set1_natural*: *F1set1 o F1map f1 f2 = image f1 o F1set1*
⟨proof⟩

theorem *F1set2_natural*: *F1set2 o F1map f1 f2 = image f2 o F1set2*
⟨proof⟩

theorem *F2set_natural*: *F2set o F2map f = image f o F2set*
⟨proof⟩

abbreviation *Fin* :: $'a1\ set \Rightarrow\ 'a2\ set \Rightarrow\ 'a3\ set \Rightarrow (('p,\ 'a1,\ 'a2,\ 'a3)\ F)\ set$ **where**
$Fin\ A1\ A2\ A3 \equiv \{x.\ Fset1\ x \subseteq A1 \wedge Fset2\ x \subseteq A2 \wedge Fset3\ x \subseteq A3\}$

abbreviation *F1in* :: $'a2\ set \Rightarrow\ 'a3\ set \Rightarrow (('p,\ 'a1,\ 'a2,\ 'a3)\ F)\ set$ **where**
$F1in\ A1\ A2 \equiv \{x.\ F1set1\ x \subseteq A1 \wedge F1set2\ x \subseteq A2\}$

lemma *F1in_alt*: *F1in A2 A3 = Fin UNIV A2 A3*
⟨proof⟩

abbreviation *F2in* :: $'a3\ set \Rightarrow (('p,\ 'a1,\ 'a2,\ 'a3)\ F)\ set$ **where**
$F2in\ A \equiv \{x.\ F2set\ x \subseteq A\}$

lemma *F2in_alt*: *F2in A3 = Fin UNIV UNIV A3*
⟨proof⟩

lemma *Frel_cong*: $[\![R1 = S1; R2 = S2; R3 = S3]\!] \Longrightarrow Frel\ R1\ R2\ R3 = Frel\ S1\ S2\ S3$
⟨proof⟩

definition *F1rel* **where**
*F1rel R1 R2 = (BNF_Def.Grp (F1in (Collect (case_prod R1)) (Collect (case_prod R2))) (F1map fst fst))^−−1
OO
(BNF_Def.Grp (F1in (Collect (case_prod R1)) (Collect (case_prod R2))) (F1map snd snd))*

lemmas *F1rel_unfold = trans[OF F1rel_def trans[OF OO_Grp_cong[OF F1in_alt]
trans[OF arg_cong2[of _ _ _ _ relcompp, OF trans[OF arg_cong[of _ _ conversep, OF sym[OF F.rel_Grp]]
F.rel_conversep[symmetric]] sym[OF F.rel_Grp]]
trans[OF F.rel_compp[symmetric] Frel_cong[OF trans[OF Grp_UNIV_id[OF refl] eq_alt[symmetric]] Grp_fst_snd
Grp_fst_snd]]]]]*

definition *F2rel* **where**
*F2rel R1 = (BNF_Def.Grp (F2in (Collect (case_prod R1))) (F2map fst))^−−1 OO
(BNF_Def.Grp (F2in (Collect (case_prod R1))) (F2map snd))*

lemmas *F2rel_unfold = trans[OF F2rel_def trans[OF OO_Grp_cong[OF F2in_alt]
trans[OF arg_cong2[of _ _ _ _ relcompp, OF trans[OF arg_cong[of _ _ conversep, OF sym[OF F.rel_Grp]]
F.rel_conversep[symmetric]] sym[OF F.rel_Grp]]
trans[OF F.rel_compp[symmetric] Frel_cong[OF trans[OF Grp_UNIV_id[OF refl] eq_alt[symmetric]] trans[OF
Grp_UNIV_id[OF refl] eq_alt[symmetric]] Grp_fst_snd]]]]]*

bnf *F1*: $('p, 'a1, 'a2, 'a3)$ *F*
  map: *F1map*
  sets: *F1set1 F1set2*
  bd: *bd_F* :: $('p$ *bd_type_F*$)$ *rel*
  rel: *F1rel*
    ⟨*proof*⟩

bnf *F2*: $('p, 'a1, 'a2, 'a3)$ *F*
  map: *F2map*
  sets: *F2set*
  bd: *bd_F* :: $('p$ *bd_type_F*$)$ *rel*
  rel: *F2rel*
    ⟨*proof*⟩

5 Adding New Live Variables

unbundle *cardinal_syntax*

declare [[*bnf_internals*]]
bnf-axiomatization (*dead* $'p$, *Fset1*: $'a1$, *Fset2*: $'a2$) *F*
  [*wits*: $'a1 \Rightarrow 'a2 \Rightarrow ('p, 'a1, 'a2)$ *F*]
  for *map*: *Fmap* *rel*: *Frel*
type-synonym $('p, 'a1, 'a2, 'a3, 'a4)$ F' = $('p, 'a3, 'a4)$ *F*

abbreviation $F'map$ :: $('a1 \Rightarrow 'b1) \Rightarrow ('a2 \Rightarrow 'b2) \Rightarrow ('a3 \Rightarrow 'b3) \Rightarrow ('a4 \Rightarrow 'b4) \Rightarrow ('p, 'a1, 'a2, 'a3, 'a4)\ F'$
$\Rightarrow ('p, 'b1, 'b2, 'b3, 'b4)\ F'$ **where**
  $F'map\ f1\ f2\ f3\ f4 \equiv Fmap\ f3\ f4$

abbreviation $F'set1$ :: $('p, 'a1, 'a2, 'a3, 'a4)\ F' \Rightarrow 'a1\ set$ **where**
  $F'set1 \equiv \lambda\_.\ \{\}$

abbreviation $F'set2$ :: $('p, 'a1, 'a2, 'a3, 'a4)\ F' \Rightarrow 'a2\ set$ **where**
  $F'set2 \equiv \lambda\_.\ \{\}$

abbreviation $F'set3$ :: $('p, 'a1, 'a2, 'a3, 'a4)\ F' \Rightarrow 'a3\ set$ **where**
  $F'set3 \equiv Fset1$

abbreviation $F'set4$ :: $('p, 'a1, 'a2, 'a3, 'a4)\ F' \Rightarrow 'a4\ set$ **where**
  $F'set4 \equiv Fset2$

abbreviation $F'bd$ **where**
  $F'bd \equiv bd\_F$

theorem *F'map_id*: $F'map\ id\ id\ id\ id = id$
  ⟨*proof*⟩

theorem *F'map_comp*:
  $F'map\ (f1\ o\ g1)\ (f2\ o\ g2)\ (f3\ o\ g3)\ (f4\ o\ g4) = F'map\ f1\ f2\ f3\ f4\ o\ F'map\ g1\ g2\ g3\ g4$
  ⟨*proof*⟩

theorem *F'map_cong*:
  $[\![\bigwedge z.\ z \in F'set1\ x \Longrightarrow f1\ z = g1\ z;\ \bigwedge z.\ z \in F'set2\ x \Longrightarrow f2\ z = g2\ z;$
  $\bigwedge z.\ z \in F'set3\ x \Longrightarrow f3\ z = g3\ z;\ \bigwedge z.\ z \in F'set4\ x \Longrightarrow f4\ z = g4\ z]\!]$
  $\Longrightarrow F'map\ f1\ f2\ f3\ f4\ x = F'map\ g1\ g2\ g3\ g4\ x$
  ⟨*proof*⟩

theorem *F'set1_natural*: $F'set1\ o\ F'map\ f1\ f2\ f3\ f4 = image\ f1\ o\ F'set1$
  ⟨*proof*⟩

theorem *F'set2_natural*: $F'set2\ o\ F'map\ f1\ f2\ f3\ f4 = image\ f2\ o\ F'set2$
  ⟨*proof*⟩

theorem $F'set3\_natural$: $F'set3\ o\ F'map\ f1\ f2\ f3\ f4 = image\ f3\ o\ F'set3$
⟨*proof*⟩

theorem $F'set4\_natural$: $F'set4\ o\ F'map\ f1\ f2\ f3\ f4 = image\ f4\ o\ F'set4$
⟨*proof*⟩

theorem $F'bd\_card\_order$: $card\_order\ bd\_F$
⟨*proof*⟩

theorem $F'bd\_cinfinite$: $cinfinite\ bd\_F$
⟨*proof*⟩

theorem $F'bd\_regularCard$: $regularCard\ bd\_F$
⟨*proof*⟩

theorem $F'set1\_bd$: $|F'set1\ x| <o\ F'bd$
⟨*proof*⟩

theorem $F'set2\_bd$: $|F'set2\ x| <o\ F'bd$
⟨*proof*⟩

theorem $F'set3\_bd$: $|F'set3\ (x :: ('c, 'a, 'd)\ F)| <o\ (F'bd :: 'c\ bd\_type\_F\ rel)$
⟨*proof*⟩

theorem $F'set4\_bd$: $|F'set4\ (x :: ('c, 'a, 'd)\ F)| <o\ (F'bd :: 'c\ bd\_type\_F\ rel)$
⟨*proof*⟩

abbreviation $F'in :: 'a1\ set \Rightarrow 'a2\ set \Rightarrow 'a3\ set \Rightarrow 'a4\ set \Rightarrow (('p, 'a1, 'a2, 'a3, 'a4)\ F')\ set$ **where**
$F'in\ A1\ A2\ A3\ A4 \equiv \{x.\ F'set1\ x \subseteq A1 \wedge F'set2\ x \subseteq A2 \wedge F'set3\ x \subseteq A3 \wedge F'set4\ x \subseteq A4\}$

definition $F'rel$ **where**
$F'rel\ R1\ R2\ R3\ R4 = (BNF\_Def.Grp\ (F'in\ (Collect\ (case\_prod\ R1))\ (Collect\ (case\_prod\ R2))\ (Collect\ (case\_prod\ R3))\ (Collect\ (case\_prod\ R4)))\ (F'map\ fst\ fst\ fst\ fst))\hat{}{--1}\ OO$
$(BNF\_Def.Grp\ (F'in\ (Collect\ (case\_prod\ R1))\ (Collect\ (case\_prod\ R2))\ (Collect\ (case\_prod\ R3))\ (Collect\ (case\_prod\ R4)))\ (F'map\ snd\ snd\ snd\ snd))$

lemmas $F'rel\_unfold = trans[OF\ F'rel\_def[unfolded\ eqTrueI[OF\ empty\_subsetI]\ simp\_thms(22)]$
$trans[OF\ OO\_Grp\_cong[OF\ refl]\ sym[OF\ F.rel\_compp\_Grp]]]$

bnf F': $('p, 'a1, 'a2, 'a3, 'a4)\ F'$
map: $F'map$
sets: $F'set1\ F'set2\ F'set3\ F'set4$
bd: $F'bd :: 'p\ bd\_type\_F\ rel$
wits: wit_F
rel: $F'rel$
plugins del: *lifting transfer*
⟨*proof*⟩

6 Changing the Order of Live Variables

unbundle *cardinal_syntax*

declare [[*bnf_internals*]]
bnf-axiomatization $(dead\ 'p,\ Fset1: 'a1,\ Fset2: 'a2,\ Fset3: 'a3)\ F$ **for** *map*: *Fmap* *rel*: *Frel*
type-synonym $('p, 'a1, 'a2, 'a3)\ F' = ('p, 'a3, 'a1, 'a2)\ F$

abbreviation $Fin :: 'a1\ set \Rightarrow 'a2\ set \Rightarrow 'a3\ set \Rightarrow (('p, 'a1, 'a2, 'a3)\ F)\ set$ **where**
$Fin\ A1\ A2\ A3 \equiv \{x.\ Fset1\ x \subseteq A1 \wedge Fset2\ x \subseteq A2 \wedge Fset3\ x \subseteq A3\}$

abbreviation $F'map :: ('a1 \Rightarrow 'b1) \Rightarrow ('a2 \Rightarrow 'b2) \Rightarrow ('a3 \Rightarrow 'b3) \Rightarrow ('p, 'a1, 'a2, 'a3)\ F' \Rightarrow ('p, 'b1, 'b2, 'b3)\ F'$ **where**
$F'map\ f\ g\ h \equiv Fmap\ h\ f\ g$

abbreviation $F'set1$ :: $('p, 'a1, 'a2, 'a3)\ F' \Rightarrow 'a1\ set$ **where**
$F'set1 \equiv Fset2$

abbreviation $F'set2$ :: $('p, 'a1, 'a2, 'a3)\ F' \Rightarrow 'a2\ set$ **where**
$F'set2 \equiv Fset3$

abbreviation $F'set3$ :: $('p, 'a1, 'a2, 'a3)\ F' \Rightarrow 'a3\ set$ **where**
$F'set3 \equiv Fset1$

abbreviation $F'bd$ **where**
$F'bd \equiv bd\_F$

theorem $F'map\_id$: $F'map\ id\ id\ id = id$
⟨proof⟩

theorem $F'map\_comp$: $F'map\ (f1\ o\ g1)\ (f2\ o\ g2)\ (f3\ o\ g3) = F'map\ f1\ f2\ f3\ o\ F'map\ g1\ g2\ g3$
⟨proof⟩

theorem $F'map\_cong$: $[\![\bigwedge z.\ z \in F'set1\ x \Longrightarrow f1\ z = g1\ z;\ \bigwedge z.\ z \in F'set2\ x \Longrightarrow f2\ z = g2\ z;\ \bigwedge z.\ z \in F'set3\ x \Longrightarrow f3\ z = g3\ z]\!]$
$\Longrightarrow F'map\ f1\ f2\ f3\ x = F'map\ g1\ g2\ g3\ x$
⟨proof⟩

theorem $F'set1\_natural$: $F'set1\ o\ F'map\ f1\ f2\ f3 = image\ f1\ o\ F'set1$
⟨proof⟩

theorem $F'set2\_natural$: $F'set2\ o\ F'map\ f1\ f2\ f3 = image\ f2\ o\ F'set2$
⟨proof⟩

theorem $F'set3\_natural$: $F'set3\ o\ F'map\ f1\ f2\ f3 = image\ f3\ o\ F'set3$
⟨proof⟩

theorem $F'bd\_card\_order$: $card\_order\ F'bd$
⟨proof⟩

theorem $F'bd\_cinfinite$: $cinfinite\ F'bd$
⟨proof⟩

theorem $F'bd\_regularCard$: $regularCard\ F'bd$
⟨proof⟩

theorem $F'set1\_bd$: $|F'set1\ (x :: ('c, 'a, 'b, 'd)\ F)| <o\ (F'bd :: 'c\ bd\_type\_F\ rel)$
⟨proof⟩

theorem $F'set2\_bd$: $|F'set2\ (x :: ('c, 'a, 'b, 'd)\ F)| <o\ (F'bd :: 'c\ bd\_type\_F\ rel)$
⟨proof⟩

theorem $F'set3\_bd$: $|F'set3\ (x :: ('c, 'a, 'b, 'd)\ F)| <o\ (F'bd :: 'c\ bd\_type\_F\ rel)$
⟨proof⟩

abbreviation $F'in$ :: $'a1\ set \Rightarrow 'a2\ set \Rightarrow 'a3\ set \Rightarrow (('p, 'a1, 'a2, 'a3)\ F')\ set$ **where**
$F'in\ A1\ A2\ A3 \equiv \{x.\ F'set1\ x \subseteq A1 \wedge F'set2\ x \subseteq A2 \wedge F'set3\ x \subseteq A3\}$

lemma $F'in\_alt$: $F'in\ A1\ A2\ A3 = Fin\ A3\ A1\ A2$
⟨proof⟩

definition $F'rel$ **where**
$F'rel\ R1\ R2\ R3 = (BNF\_Def.Grp\ (F'in\ (Collect\ (case\_prod\ R1))\ (Collect\ (case\_prod\ R2))\ (Collect\ (case\_prod\ R3)))\ (F'map\ fst\ fst\ fst))\widehat{\ }{-}{-}1\ OO$
$(BNF\_Def.Grp\ (F'in\ (Collect\ (case\_prod\ R1))\ (Collect\ (case\_prod\ R2))\ (Collect\ (case\_prod\ R3)))\ (F'map\ snd\ snd\ snd))$

lemmas *F'rel_unfold* = *trans*[*OF F'rel_def trans*[*OF OO_Grp_cong*[*OF F'in_alt*] *sym*[*OF F.rel_compp_Grp*]]]

bnf *F'*: (*'p*, *'a1*, *'a2*, *'a3*) *F'*
  map: *F'map*
  sets: *F'set1 F'set2 F'set3*
  bd: *F'bd* :: *'p bd_type_F rel*
  rel: *F'rel*
    ⟨*proof*⟩

7 Mutual View on Nested Datatypes

notation *BNF_Def.convol* (‹<_, _>›)

declare [[*bnf_internals*]]

declare [[*typedef_overloaded*]]

bnf-axiomatization (*'a*, *'b*) *F0* [*wits*: *'a* ⇒ (*'a*, *'b*) *F0*]
bnf-axiomatization (*'a*, *'b*) *G0* [*wits*: *'a* ⇒ (*'a*, *'b*) *G0*]

7.1 Nested Definition

datatype *'a F* = *CF* (*'a*, *'a F*) *F0*
datatype *'a G* = *CG* (*'a*, (*'a G*) *F*) *G0*

type-synonym (*'b*, *'c*) *F_pre_F* = (*'c*, *'b*) *F0*
type-synonym (*'c*, *'a*) *G_pre_G* = (*'a*, *'c F*) *G0*

term *ctor_fold_F* :: ((*'b*, *'c*) *F_pre_F* ⇒ *'b*) ⇒ *'c F* ⇒ *'b*
term *ctor_fold_G* :: ((*'c*, *'a*) *G_pre_G* ⇒ *'c*) ⇒ *'a G* ⇒ *'c*
term *ctor_rec_F* :: ((*'c F* × *'b*, *'c*) *F_pre_F* ⇒ *'b*) ⇒ *'c F* ⇒ *'b*
term *ctor_rec_G* :: ((*'a G* × *'c*, *'a*) *G_pre_G* ⇒ *'c*) ⇒ *'a G* ⇒ *'c*
thm *F.ctor_rel_induct*
thm *G.ctor_rel_induct*[*unfolded rel_pre_G_def id_apply*]

7.2 Isomorphic Mutual Definition

datatype *'a* G_M = *CG* (*'a*, *'a* GF_M) *G0*
  and *'a* GF_M = *CF* (*'a* G_M, *'a* GF_M) *F0*

type-synonym (*'b*, *'c*) GF_M*_pre_*GF_M = (*'c*, *'b*) *F0*
type-synonym (*'c*, *'a*) G_M*_pre_*G_M = (*'a*, *'c*) *G0*

term *ctor_fold_*G_M :: ((*'c*, *'a*) G_M*_pre_*G_M ⇒ *'b*) ⇒ ((*'c*, *'b*) GF_M*_pre_*GF_M ⇒ *'c*) ⇒ *'a* G_M ⇒ *'b*
term *ctor_fold_*GF_M :: ((*'c*, *'a*) G_M*_pre_*G_M ⇒ *'b*) ⇒ ((*'c*, *'b*) GF_M*_pre_*GF_M ⇒ *'c*) ⇒ *'a* GF_M ⇒ *'c*
term *ctor_rec_*G_M :: ((*'a* GF_M × *'c*, *'a*) G_M*_pre_*G_M ⇒ *'b*) ⇒ ((*'a* GF_M × *'c*, *'a* G_M × *'b*) GF_M*_pre_*GF_M ⇒ *'c*) ⇒ *'a* G_M ⇒ *'b*
term *ctor_rec_*GF_M :: ((*'a* GF_M × *'c*, *'a*) G_M*_pre_*G_M ⇒ *'b*) ⇒ ((*'a* GF_M × *'c*, *'a* G_M × *'b*) GF_M*_pre_*GF_M ⇒ *'c*) ⇒ *'a* GF_M ⇒ *'c*
thm G_M*_*GF_M*.ctor_rel_induct*[*unfolded rel_pre_*G_M*_def rel_pre_*GF_M*_def*]

7.3 Mutualization

7.3.1 Iterators

definition *n2m_ctor_fold_G* :: ((*'c*, *'a*) G_M*_pre_*G_M ⇒ *'b*) ⇒ ((*'c*, *'b*) GF_M*_pre_*GF_M ⇒ *'c*) ⇒ *'a G* ⇒ *'b*
  where *n2m_ctor_fold_G s1 s2* = *ctor_fold_G* (*s1 o*
   *map_pre_*G_M *id* (*id* :: *unit* ⇒ *unit*) (*ctor_fold_F* (*s2 o BNF_Composition.id_bnf o BNF_Composition.id_bnf*)) *o BNF_Composition.id_bnf o BNF_Composition.id_bnf*)
definition *n2m_ctor_fold_G_F* :: ((*'c*, *'a*) G_M*_pre_*G_M ⇒ *'b*) ⇒ ((*'c*, *'b*) GF_M*_pre_*GF_M ⇒ *'c*) ⇒ *'a G F* ⇒ *'c*

where *n2m_ctor_fold_G_F s1 s2* = *ctor_fold_F* (*s2 o map_pre_GF*$_M$ (*id* :: *unit* ⇒ *unit*) (*n2m_ctor_fold_G s1 s2*) *id o BNF_Composition.id_bnf o BNF_Composition.id_bnf*)

lemma *G_ctor_o_fold*: *ctor_fold_G s o ctor_G* = *s o map_pre_G id* (*ctor_fold_G s*)
⟨*proof*⟩
lemma *F_ctor_o_fold*: *ctor_fold_F s o ctor_F* = *s o map_pre_F id* (*ctor_fold_F s*)
⟨*proof*⟩

lemma *G_ctor_o_rec*: *ctor_rec_G s o ctor_G* = *s o map_pre_G id* (*BNF_Def.convol id* (*ctor_rec_G s*))
⟨*proof*⟩
lemma *F_ctor_o_rec*: *ctor_rec_F s o ctor_F* = *s o map_pre_F id* (*BNF_Def.convol id* (*ctor_rec_F s*))
⟨*proof*⟩

lemma *n2m_ctor_fold_G*:
n2m_ctor_fold_G s1 s2 o ctor_G = *s1 o map_pre_G*$_M$ *id id* (*n2m_ctor_fold_G_F s1 s2*) *o BNF_Composition.id_bnf o BNF_Composition.id_bnf*
⟨*proof*⟩

lemma *n2m_ctor_fold_G_F*:
n2m_ctor_fold_G_F s1 s2 o ctor_F = *s2 o map_pre_GF*$_M$ *id* (*n2m_ctor_fold_G s1 s2*) (*n2m_ctor_fold_G_F s1 s2*) *o BNF_Composition.id_bnf o BNF_Composition.id_bnf*
⟨*proof*⟩

7.3.2 Recursors

definition *n2m_ctor_rec_G* ::
(('*a G F* × '*c*, '*a*) *G*$_M$*_pre_G*$_M$ ⇒ '*b*) ⇒ (('*a G F* × '*c*, '*a G* × '*b*) *GF*$_M$*_pre_GF*$_M$ ⇒ '*c*) ⇒ '*a G* ⇒ '*b*
where *n2m_ctor_rec_G s1 s2* =
ctor_rec_G (*s1 o*
map_pre_G$_M$ *id* (*id* :: *unit* ⇒ *unit*)
(*BNF_Def.convol* (*map_F fst*) (*ctor_rec_F* (*s2 o map_pre_GF*$_M$ (*id* :: *unit* ⇒ *unit*) *id* (*map_prod* (*map_F fst*) *id*) *o BNF_Composition.id_bnf o BNF_Composition.id_bnf*))) *o*
BNF_Composition.id_bnf o BNF_Composition.id_bnf)

definition *n2m_ctor_rec_G_F* ::
(('*a G F* × '*c*, '*a*) *G*$_M$*_pre_G*$_M$ ⇒ '*b*) ⇒ (('*a G F* × '*c*, '*a G* × '*b*) *GF*$_M$*_pre_GF*$_M$ ⇒ '*c*) ⇒ '*a G F* ⇒ '*c*
where *n2m_ctor_rec_G_F s1 s2* = *ctor_rec_F* (*s2 o map_pre_GF*$_M$ (*id* :: *unit* ⇒ *unit*) (*BNF_Def.convol id* (*n2m_ctor_rec_G s1 s2*)) *id o BNF_Composition.id_bnf o BNF_Composition.id_bnf*)

lemma *n2m_ctor_rec_G*:
n2m_ctor_rec_G s1 s2 o ctor_G = *s1 o map_pre_G*$_M$ *id id* (*BNF_Def.convol id* (*n2m_ctor_rec_G_F s1 s2*)) *o BNF_Composition.id_bnf o BNF_Composition.id_bnf*
⟨*proof*⟩

lemma *n2m_ctor_rec_G_F*:
n2m_ctor_rec_G_F s1 s2 o ctor_F = *s2 o map_pre_GF*$_M$ *id* (*BNF_Def.convol id* (*n2m_ctor_rec_G s1 s2*)) (*BNF_Def.convol id* (*n2m_ctor_rec_G_F s1 s2*)) *o BNF_Composition.id_bnf o BNF_Composition.id_bnf*
⟨*proof*⟩

7.3.3 Induction

lemma *n2m_rel_induct_G_G_F*:
assumes *IH1*: ∀ *x y*. *BNF_Def.vimage2p* (*BNF_Composition.id_bnf o BNF_Composition.id_bnf*) (*BNF_Composition.id_bnf o BNF_Composition.id_bnf*) (*rel_pre_G*$_M$ *P R S*) *x y* ⟶ *R* (*ctor_G x*) (*ctor_G y*)
and *IH2*: ∀ *x y*. *BNF_Def.vimage2p* (*BNF_Composition.id_bnf o BNF_Composition.id_bnf*) (*BNF_Composition.id_bnf o BNF_Composition.id_bnf*) (*rel_pre_GF*$_M$ *P R S*) *x y* ⟶ *S* (*ctor_F x*) (*ctor_F y*)
shows *rel_G P* ≤ *R* ∧ *rel_F* (*rel_G P*) ≤ *S*
⟨*proof*⟩

lemmas *n2m_ctor_induct_G_G_F* = *spec*[*OF spec* [*OF*
n2m_rel_induct_G_G_F[*of* (=) *BNF_Def.Grp* (*Collect R*) *id BNF_Def.Grp* (*Collect S*) *id* **for** *R S*,
unfolded G.rel_eq F.rel_eq eq_le_Grp_id_iff all_simps(*1*,*2*)[*symmetric*]]],
unfolded eq_alt pre_G$_M$*.rel_Grp pre_GF*$_M$*.rel_Grp pre_G*$_M$*.map_id0 pre_GF*$_M$*.map_id0*,
unfolded vimage2p_comp vimage2p_id comp_apply comp_id Grp_id_mono_subst

type_copy_vimage2p_Grp_Rep[*OF BNF_Composition.type_definition_id_bnf_UNIV*]
type_copy_Abs_o_Rep[*OF BNF_Composition.type_definition_id_bnf_UNIV*]
eqTrueI[*OF subset_UNIV*] *simp_thms*(*22*)
atomize_conjL[*symmetric*] *atomize_all*[*symmetric*] *atomize_imp*[*symmetric*],
unfolded subset_iff mem_Collect_eq]

8 Mutual View on Nested Coatatypes

bnf-axiomatization (*'a*, *'b*) *coF0*
bnf-axiomatization (*'a*, *'b*) *coG0*

8.1 Nested definition

codatatype *'a coF* = *CcoF* (*'a*, *'a coF*) *coF0*
codatatype *'a coG* = *CcoG* (*'a*, (*'a coG*) *coF*) *coG0*

type-synonym (*'b*, *'c*) *coF_pre_coF* = (*'c*, *'b*) *coF0*
type-synonym (*'c*, *'a*) *coG_pre_coG* = (*'a*, *'c coF*) *coG0*

term *dtor_unfold_coF* :: (*'b* ⇒ (*'b*, *'c*) *coF_pre_coF*) ⇒ *'b* ⇒ *'c coF*
term *dtor_unfold_coG* :: (*'c* ⇒ (*'c*, *'a*) *coG_pre_coG*) ⇒ *'c* ⇒ *'a coG*
term *dtor_corec_coF* :: (*'b* ⇒ (*'c coF* + *'b*, *'c*) *coF_pre_coF*) ⇒ *'b* ⇒ *'c coF*
term *dtor_corec_coG* :: (*'c* ⇒ (*'a coG* + *'c*, *'a*) *coG_pre_coG*) ⇒ *'c* ⇒ *'a coG*
thm *coF.dtor_rel_coinduct*
thm *coG.dtor_rel_coinduct*[*unfolded rel_pre_coG_def id_apply*]

8.2 Isomorphic Mutual Definition

codatatype *'a* coG_M = *CcoG* (*'a*, *'a* $coGcoF_M$) *coG0*
  and *'a* $coGcoF_M$ = *CcoF* (*'a* coG_M, *'a* $coGcoF_M$) *coF0*

type-synonym (*'b*, *'c*) $coGcoF_M$*_pre_*$coGcoF_M$ = (*'c*, *'b*) *coF0*
type-synonym (*'c*, *'a*) coG_M*_pre_*coG_M = (*'a*, *'c*) *coG0*

term *dtor_unfold_*coG_M :: (*'b* ⇒ (*'c*, *'a*) coG_M*_pre_*coG_M) ⇒ (*'c* ⇒ (*'c*, *'b*) $coGcoF_M$*_pre_*$coGcoF_M$) ⇒ *'b* ⇒ *'a* coG_M
term *dtor_unfold_*$coGcoF_M$ :: (*'b* ⇒ (*'c*, *'a*) coG_M*_pre_*coG_M) ⇒ (*'c* ⇒ (*'c*, *'b*) $coGcoF_M$*_pre_*$coGcoF_M$) ⇒ *'c* ⇒ *'a* $coGcoF_M$
term *dtor_corec_*coG_M :: (*'b* ⇒ (*'a* $coGcoF_M$ + *'c*, *'a*) coG_M*_pre_*coG_M) ⇒ (*'c* ⇒ (*'a* $coGcoF_M$ + *'c*, *'a* coG_M + *'b*) $coGcoF_M$*_pre_*$coGcoF_M$) ⇒ *'b* ⇒ *'a* coG_M
term *dtor_corec_*$coGcoF_M$ :: (*'b* ⇒ (*'a* $coGcoF_M$ + *'c*, *'a*) coG_M*_pre_*coG_M) ⇒ (*'c* ⇒ (*'a* $coGcoF_M$ + *'c*, *'a* coG_M + *'b*) $coGcoF_M$*_pre_*$coGcoF_M$) ⇒ *'c* ⇒ *'a* $coGcoF_M$
thm coG_M*_*$coGcoF_M$*.dtor_rel_coinduct*[*unfolded rel_pre_*coG_M*_def rel_pre_*$coGcoF_M$*_def*]

8.3 Mutualization

8.3.1 Coiterators

definition *n2m_dtor_unfold_coG* :: (*'b* ⇒ (*'c*, *'a*) coG_M*_pre_*coG_M) ⇒ (*'c* ⇒ (*'c*, *'b*) $coGcoF_M$*_pre_*$coGcoF_M$) ⇒ *'b* ⇒ *'a coG*
  where *n2m_dtor_unfold_coG s1 s2* = *dtor_unfold_coG* (*BNF_Composition.id_bnf o BNF_Composition.id_bnf o*
    *map_pre_*coG_M *id* (*id* :: *unit* ⇒ *unit*) (*dtor_unfold_coF* (*BNF_Composition.id_bnf o BNF_Composition.id_bnf o s2*)) *o s1*)
definition *n2m_dtor_unfold_coG_coF* :: (*'b* ⇒ (*'c*, *'a*) coG_M*_pre_*coG_M) ⇒ (*'c* ⇒ (*'c*, *'b*) $coGcoF_M$*_pre_*$coGcoF_M$) ⇒ *'c* ⇒ *'a coG coF*
  where *n2m_dtor_unfold_coG_coF s1 s2* = *dtor_unfold_coF* (*BNF_Composition.id_bnf o BNF_Composition.id_bnf o map_pre_*$coGcoF_M$ (*id* :: *unit* ⇒ *unit*) (*n2m_dtor_unfold_coG s1 s2*) *id o s2*)

lemma *coG_dtor_o_unfold*: *dtor_coG o dtor_unfold_coG s* = *map_pre_coG id* (*dtor_unfold_coG s*) *o s*
  ⟨*proof*⟩
lemma *coF_dtor_o_unfold*: *dtor_coF o dtor_unfold_coF s* = *map_pre_coF id* (*dtor_unfold_coF s*) *o s*
  ⟨*proof*⟩

lemma *coG_dtor_o_corec*: *dtor_coG o dtor_corec_coG s* = *map_pre_coG id* (*case_sum id* (*dtor_corec_coG s*)) *o s*
⟨*proof*⟩
lemma *coF_dtor_o_corec*: *dtor_coF o dtor_corec_coF s* = *map_pre_coF id* (*case_sum id* (*dtor_corec_coF s*)) *o s*
⟨*proof*⟩

lemma *n2m_dtor_unfold_coG*:
dtor_coG o n2m_dtor_unfold_coG s1 s2 = *BNF_Composition.id_bnf o BNF_Composition.id_bnf o* $map\_pre\_coG_M$ *id id* (*n2m_dtor_unfold_coG_coF s1 s2*) *o s1*
⟨*proof*⟩

lemma *n2m_dtor_unfold_coG_coF*:
dtor_coF o n2m_dtor_unfold_coG_coF s1 s2 = *BNF_Composition.id_bnf o BNF_Composition.id_bnf o* $map\_pre\_coGcoF_M$ *id* (*n2m_dtor_unfold_coG s1 s2*) (*n2m_dtor_unfold_coG_coF s1 s2*) *o s2*
⟨*proof*⟩

8.3.2 Corecursors

definition *n2m_dtor_corec_coG* ::
$('b \Rightarrow ('a\ coG\ coF + 'c, 'a)\ coG_M\_pre\_coG_M) \Rightarrow ('c \Rightarrow ('a\ coG\ coF + 'c, 'a\ coG + 'b)\ coGcoF_M\_pre\_coGcoF_M) \Rightarrow 'b \Rightarrow 'a\ coG$
where *n2m_dtor_corec_coG s1 s2* =
dtor_corec_coG (*BNF_Composition.id_bnf o BNF_Composition.id_bnf o*
$map\_pre\_coG_M$ *id* (*id* :: *unit* ⇒ *unit*)
(*case_sum* (*map_coF Inl*) (*dtor_corec_coF* (*BNF_Composition.id_bnf o BNF_Composition.id_bnf o* $map\_pre\_coGcoF_M$ (*id* :: *unit* ⇒ *unit*) *id* (*map_sum* (*map_coF Inl*) *id*) *o s2*))) *o*
s1)

definition *n2m_dtor_corec_coG_coF* ::
$('b \Rightarrow ('a\ coG\ coF + 'c, 'a)\ coG_M\_pre\_coG_M) \Rightarrow ('c \Rightarrow ('a\ coG\ coF + 'c, 'a\ coG + 'b)\ coGcoF_M\_pre\_coGcoF_M) \Rightarrow 'c \Rightarrow 'a\ coG\ coF$
where *n2m_dtor_corec_coG_coF s1 s2* = *dtor_corec_coF* (*BNF_Composition.id_bnf o BNF_Composition.id_bnf o* $map\_pre\_coGcoF_M$ (*id* :: *unit* ⇒ *unit*) (*case_sum id* (*n2m_dtor_corec_coG s1 s2*)) *id o s2*)

lemma *n2m_dtor_corec_coG*:
dtor_coG o n2m_dtor_corec_coG s1 s2 = *BNF_Composition.id_bnf o BNF_Composition.id_bnf o* $map\_pre\_coG_M$ *id id* (*case_sum id* (*n2m_dtor_corec_coG_coF s1 s2*)) *o s1*
⟨*proof*⟩

lemma *n2m_dtor_corec_coG_coF*:
dtor_coF o n2m_dtor_corec_coG_coF s1 s2 = *BNF_Composition.id_bnf o BNF_Composition.id_bnf o* $map\_pre\_coGcoF_M$ *id* (*case_sum id* (*n2m_dtor_corec_coG s1 s2*)) (*case_sum id* (*n2m_dtor_corec_coG_coF s1 s2*)) *o s2*
⟨*proof*⟩

8.3.3 Coinduction

lemma *n2m_rel_coinduct_coG_coG_coF*:
assumes *CIH1*: ∀ *x y*. *R x y* ⟶ *BNF_Def.vimage2p* (*BNF_Composition.id_bnf o BNF_Composition.id_bnf*) (*BNF_Composition.id_bnf o BNF_Composition.id_bnf*) ($rel\_pre\_coG_M$ *P R S*) (*dtor_coG x*) (*dtor_coG y*)
and *CIH2*: ∀ *x y*. *S x y* ⟶ *BNF_Def.vimage2p* (*BNF_Composition.id_bnf o BNF_Composition.id_bnf*) (*BNF_Composition.id_bnf o BNF_Composition.id_bnf*) ($rel\_pre\_coGcoF_M$ *P R S*) (*dtor_coF x*) (*dtor_coF y*)
shows *R* ≤ *rel_coG P* ∧ *S* ≤ *rel_coF* (*rel_coG P*)
⟨*proof*⟩

lemmas *n2m_ctor_induct_coG_coG_coF* = *spec*[*OF spec*[*OF spec*[*OF spec*[*OF*
n2m_rel_coinduct_coG_coG_coF[*of _* (=),
unfolded coG.rel_eq coF.rel_eq le_fun_def le_bool_def all_simps(*1*,*2*)[*symmetric*]]]]]]