

Bounded Natural Functors with Covariance and Contravariance

Andreas Lochbihler and Joshua Schneider

December 14, 2021

Abstract

Bounded natural functors (BNFs) provide a modular framework for the construction of (co)datatypes in higher-order logic. Their functorial operations, the mapper and relator, are restricted to a subset of the parameters, namely those where recursion can take place. For certain applications, such as free theorems, data refinement, quotients, and generalised rewriting, it is desirable that these operations do not ignore the other parameters. In this article, we formalise the generalisation BNF_{CC} [2] that extends the mapper and relator to covariant and contravariant parameters. We show that (i) BNF_{CC} s are closed under functor composition and least and greatest fixpoints, (ii) subtypes inherit the BNF_{CC} structure under conditions that generalise those for the BNF case, and (iii) BNF_{CC} s preserve quotients under mild conditions. These proofs are carried out for abstract BNF_{CC} s similar to the AFP entry BNF Operations [1]. In addition, we apply the BNF_{CC} theory to several concrete functors.

For an informal description of the abstract proofs see [2].

Contents

1	Preliminaries	4
2	Axiomatisation	5
2.1	First abstract BNF_{CC}	5
2.1.1	Axioms and basic definitions	5
2.1.2	Derived rules	8
2.1.3	F is a BNF	9
2.1.4	Composition witness	10
2.2	Second abstract BNF_{CC}	11
2.2.1	Axioms and basic definitions	11
2.2.2	Derived rules	13
2.2.3	G is a BNF	14
2.2.4	Composition witness	15
3	Simple operations: demotion, merging, composition	15
3.1	Composition in a live position	16
3.2	Composition in a covariant position	21
3.3	Composition in a contravariant position	25
3.4	Composition in a fixed position	30
4	Least and greatest fixpoints	33
4.1	Least fixpoint	34
4.1.1	BNF_{CC} structure	34
4.1.2	Parametricity laws	38
4.2	Greatest fixpoints	38
4.2.1	BNF_{CC} structure	38
4.2.2	Parametricity laws	43
5	Subtypes	43
5.1	BNF_{CC} structure	43
5.2	Closedness under zippings	47
5.3	Subtypes of BNFs without co- and contravariance	50
6	Quotient preservation	51
7	Concrete BNF_{CCs}	51
7.1	Function space	52
7.2	Covariant powerset	53
7.3	Bounded sets	53
7.4	Contravariant powerset (sets as predicates)	54
7.5	Filter	56
7.6	Almost-everywhere equal sequences	58

8	Example: deterministic discrete system	59
8.1	Definition and generalised mapper and relator	59
8.2	Evenness of partial sums	60
8.3	Composition	61
8.4	Graph traversal: refinement and quotients	61
8.5	Generalised rewriting	63

1 Preliminaries

theory *Preliminaries* **imports**

Main

begin

alias *Grp* = *BNF-Def.Grp*

alias *vimage2p* = *BNF-Def.vimage2p*

lemma *Domainp-conversep*: $\text{Domainp } R^{-1-1} = \text{Rangep } R$
<proof>

lemma *Grp-apply*: $\text{Grp } A \ f \ x \ y \longleftrightarrow y = f \ x \wedge x \in A$
<proof>

lemma *conversep-Grp-id*: $(\text{Grp } A \ \text{id})^{-1-1} = \text{Grp } A \ \text{id}$
<proof>

lemma *eq-onp-compp-Grp*: $\text{eq-onp } P \ \text{OO } \text{Grp } A \ f = \text{Grp } (\text{Collect } P \cap A) \ f$
<proof>

consts *relcompp-witness* :: $('a \Rightarrow 'b \Rightarrow \text{bool}) \Rightarrow ('b \Rightarrow 'c \Rightarrow \text{bool}) \Rightarrow 'a \times 'c \Rightarrow 'b$

specification (*relcompp-witness*)

relcompp-witness1: $(A \ \text{OO } B) \ (\text{fst } xy) \ (\text{snd } xy) \Longrightarrow A \ (\text{fst } xy) \ (\text{relcompp-witness } A \ B \ xy)$

relcompp-witness2: $(A \ \text{OO } B) \ (\text{fst } xy) \ (\text{snd } xy) \Longrightarrow B \ (\text{relcompp-witness } A \ B \ xy) \ (\text{snd } xy)$
<proof>

lemmas *relcompp-witness*[*of - - (x, y) for x y, simplified*] = *relcompp-witness1*
relcompp-witness2

hide-fact (**open**) *relcompp-witness1* *relcompp-witness2*

lemma *relcompp-witness-eq* [*simp*]: *relcompp-witness* (=) (=) $(x, x) = x$
<proof>

lemma *Quotient-equiv-abs1*: $\llbracket \text{Quotient } R \ \text{Abs } \text{Rep } T; R \ x \ y \rrbracket \Longrightarrow T \ x \ (\text{Abs } y)$
<proof>

lemma *Quotient-equiv-abs2*: $\llbracket \text{Quotient } R \ \text{Abs } \text{Rep } T; R \ x \ y \rrbracket \Longrightarrow T \ y \ (\text{Abs } x)$
<proof>

lemma *Quotient-rep-equiv1*: $\llbracket \text{Quotient } R \ \text{Abs } \text{Rep } T; T \ a \ b \rrbracket \Longrightarrow R \ a \ (\text{Rep } b)$
<proof>

lemma *Quotient-rep-equiv2*: $\llbracket \text{Quotient } R \text{ Abs Rep } T; T \text{ a } b \rrbracket \implies R (\text{Rep } b) a$
<proof>

end

2 Axiomatisation

theory *Axiomatised-BNF-CC* **imports**

Preliminaries

HOL-Library.Rewrite

begin

unbundle *cardinal-syntax*

This theory axiomatises two BNF_{CCS} , which will be used to demonstrate the closedness of BNF_{CCS} under various operations.

2.1 First abstract BNF_{CC}

2.1.1 Axioms and basic definitions

typedecl (*'l1, 'l2, 'l3, 'co1, 'co2, 'co3, 'contra1, 'contra2, 'contra3, 'f*) *F*

F has each three live, co-, and contravariant parameters, and one fixed parameter.

consts

rel-F :: (*'l1* \Rightarrow *'l1'* \Rightarrow *bool*) \Rightarrow (*'l2* \Rightarrow *'l2'* \Rightarrow *bool*) \Rightarrow (*'l3* \Rightarrow *'l3'* \Rightarrow *bool*) \Rightarrow
'co1 \Rightarrow *'co1'* \Rightarrow *bool*) \Rightarrow (*'co2* \Rightarrow *'co2'* \Rightarrow *bool*) \Rightarrow (*'co3* \Rightarrow *'co3'* \Rightarrow *bool*) \Rightarrow
'contra1 \Rightarrow *'contra1'* \Rightarrow *bool*) \Rightarrow (*'contra2* \Rightarrow *'contra2'* \Rightarrow *bool*) \Rightarrow
'contra3 \Rightarrow *'contra3'* \Rightarrow *bool*) \Rightarrow
(*'l1, 'l2, 'l3, 'co1, 'co2, 'co3, 'contra1, 'contra2, 'contra3, 'f*) *F* \Rightarrow
(*'l1', 'l2', 'l3', 'co1', 'co2', 'co3', 'contra1', 'contra2', 'contra3', 'f*) *F* \Rightarrow *bool*

map-F :: (*'l1* \Rightarrow *'l1'*) \Rightarrow (*'l2* \Rightarrow *'l2'*) \Rightarrow (*'l3* \Rightarrow *'l3'*) \Rightarrow
(*'co1* \Rightarrow *'co1'*) \Rightarrow (*'co2* \Rightarrow *'co2'*) \Rightarrow (*'co3* \Rightarrow *'co3'*) \Rightarrow
(*'contra1'* \Rightarrow *'contra1'*) \Rightarrow (*'contra2'* \Rightarrow *'contra2'*) \Rightarrow (*'contra3'* \Rightarrow *'contra3'*) \Rightarrow
(*'l1, 'l2, 'l3, 'co1, 'co2, 'co3, 'contra1, 'contra2, 'contra3, 'f*) *F* \Rightarrow
(*'l1', 'l2', 'l3', 'co1', 'co2', 'co3', 'contra1', 'contra2', 'contra3', 'f*) *F*

axiomatization where

rel-F-mono [*mono*]:

$\bigwedge L1 L1' L2 L2' L3 L3' Co1 Co1' Co2 Co2' Co3 Co3'$

Contra1 Contra1' Contra2 Contra2' Contra3 Contra3'.

$\llbracket L1 \leq L1'; L2 \leq L2'; L3 \leq L3'; Co1 \leq Co1'; Co2 \leq Co2'; Co3 \leq Co3';$

Contra1' \leq Contra1; Contra2' \leq Contra2; Contra3' \leq Contra3 $\rrbracket \implies$

rel-F *L1 L2 L3 Co1 Co2 Co3 Contra1 Contra2 Contra3* \leq

rel-F *L1' L2' L3' Co1' Co2' Co3' Contra1' Contra2' Contra3'* **and**

rel-F-eq: *rel-F* (=) (=) (=) (=) (=) (=) (=) (=) (=) **and**

rel-F-conversep: $\bigwedge L1 L2 L3 Co1 Co2 Co3 Contra1 Contra2 Contra3$.

$rel-F L1^{-1-1} L2^{-1-1} L3^{-1-1} Co1^{-1-1} Co2^{-1-1} Co3^{-1-1} Contra1^{-1-1} Contra2^{-1-1} Contra3^{-1-1} =$
 $(rel-F L1 L2 L3 Co1 Co2 Co3 Contra1 Contra2 Contra3)^{-1-1}$ **and**
 $map-F-id0: map-F id id id id id id id id id = id$ **and**
 $map-F-comp: \bigwedge l1 l1' l2 l2' l3 l3' co1 co1' co2 co2' co3 co3'$
 $contra1 contra1' contra2 contra2' contra3 contra3'.$
 $map-F l1 l2 l3 co1 co2 co3 contra1 contra2 contra3 \circ$
 $map-F l1' l2' l3' co1' co2' co3' contra1' contra2' contra3' =$
 $map-F (l1 \circ l1') (l2 \circ l2') (l3 \circ l3') (co1 \circ co1') (co2 \circ co2') (co3 \circ co3')$
 $(contra1' \circ contra1) (contra2' \circ contra2) (contra3' \circ contra3)$ **and**
 $map-F-parametric:$
 $\bigwedge L1 L1' L2 L2' L3 L3' Co1 Co1' Co2 Co2' Co3 Co3'$
 $Contra1 Contra1' Contra2 Contra2' Contra3 Contra3'.$
 $rel-fun (rel-fun L1 L1') (rel-fun (rel-fun L2 L2')) (rel-fun (rel-fun L3 L3'))$
 $(rel-fun (rel-fun Co1 Co1')) (rel-fun (rel-fun Co2 Co2')) (rel-fun (rel-fun Co3 Co3'))$
 $(rel-fun (rel-fun Contra1' Contra1)) (rel-fun (rel-fun Contra2' Contra2))$
 $(rel-fun (rel-fun Contra3' Contra3))$
 $(rel-fun (rel-F L1 L2 L3 Co1 Co2 Co3 Contra1 Contra2 Contra3))$
 $(rel-F L1' L2' L3' Co1' Co2' Co3' Contra1' Contra2' Contra3'))))))))$
 $map-F map-F$

definition $rel-F-pos-distr-cond :: ('co1 \Rightarrow 'co1' \Rightarrow bool) \Rightarrow ('co1' \Rightarrow 'co1'' \Rightarrow bool) \Rightarrow$
 $bool) \Rightarrow$
 $('co2 \Rightarrow 'co2' \Rightarrow bool) \Rightarrow ('co2' \Rightarrow 'co2'' \Rightarrow bool) \Rightarrow$
 $('co3 \Rightarrow 'co3' \Rightarrow bool) \Rightarrow ('co3' \Rightarrow 'co3'' \Rightarrow bool) \Rightarrow$
 $('contra1 \Rightarrow 'contra1' \Rightarrow bool) \Rightarrow ('contra1' \Rightarrow 'contra1'' \Rightarrow bool) \Rightarrow$
 $('contra2 \Rightarrow 'contra2' \Rightarrow bool) \Rightarrow ('contra2' \Rightarrow 'contra2'' \Rightarrow bool) \Rightarrow$
 $('contra3 \Rightarrow 'contra3' \Rightarrow bool) \Rightarrow ('contra3' \Rightarrow 'contra3'' \Rightarrow bool) \Rightarrow$
 $('l1 \times 'l1' \times 'l1'' \times 'l2 \times 'l2' \times 'l2'' \times 'l3 \times 'l3' \times 'l3'' \times 'f) itself \Rightarrow bool$

where

$rel-F-pos-distr-cond Co1 Co1' Co2 Co2' Co3 Co3'$
 $Contra1 Contra1' Contra2 Contra2' Contra3 Contra3' - \longleftrightarrow$
 $(\forall (L1 :: 'l1 \Rightarrow 'l1' \Rightarrow bool) (L1' :: 'l1' \Rightarrow 'l1'' \Rightarrow bool)$
 $(L2 :: 'l2 \Rightarrow 'l2' \Rightarrow bool) (L2' :: 'l2' \Rightarrow 'l2'' \Rightarrow bool)$
 $(L3 :: 'l3 \Rightarrow 'l3' \Rightarrow bool) (L3' :: 'l3' \Rightarrow 'l3'' \Rightarrow bool).$
 $(rel-F L1 L2 L3 Co1 Co2 Co3 Contra1 Contra2 Contra3 ::$
 $(-, -, -, -, -, -, -, -, 'f) F \Rightarrow -) OO$
 $rel-F L1' L2' L3' Co1' Co2' Co3' Contra1' Contra2' Contra3' \leq$
 $rel-F (L1 OO L1') (L2 OO L2') (L3 OO L3') (Co1 OO Co1') (Co2 OO Co2')$
 $(Co3 OO Co3')$
 $(Contra1 OO Contra1') (Contra2 OO Contra2') (Contra3 OO Contra3'))$

definition $rel-F-neg-distr-cond :: ('co1 \Rightarrow 'co1' \Rightarrow bool) \Rightarrow ('co1' \Rightarrow 'co1'' \Rightarrow bool) \Rightarrow$
 $bool) \Rightarrow$
 $('co2 \Rightarrow 'co2' \Rightarrow bool) \Rightarrow ('co2' \Rightarrow 'co2'' \Rightarrow bool) \Rightarrow$
 $('co3 \Rightarrow 'co3' \Rightarrow bool) \Rightarrow ('co3' \Rightarrow 'co3'' \Rightarrow bool) \Rightarrow$
 $('contra1 \Rightarrow 'contra1' \Rightarrow bool) \Rightarrow ('contra1' \Rightarrow 'contra1'' \Rightarrow bool) \Rightarrow$
 $('contra2 \Rightarrow 'contra2' \Rightarrow bool) \Rightarrow ('contra2' \Rightarrow 'contra2'' \Rightarrow bool) \Rightarrow$

$(\text{'contra3} \Rightarrow \text{'contra3}' \Rightarrow \text{bool}) \Rightarrow (\text{'contra3}' \Rightarrow \text{'contra3}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'l1} \times \text{'l1}' \times \text{'l1}'' \times \text{'l2} \times \text{'l2}' \times \text{'l2}'' \times \text{'l3} \times \text{'l3}' \times \text{'l3}'' \times \text{'f}) \text{ itself} \Rightarrow \text{bool}$

where

$\text{rel-F-neg-distr-cond } \text{Co1 } \text{Co1}' \text{ Co2 } \text{Co2}' \text{ Co3 } \text{Co3}'$
 $\text{Contra1 } \text{Contra1}' \text{ Contra2 } \text{Contra2}' \text{ Contra3 } \text{Contra3}' - \longleftrightarrow$
 $(\forall (L1 :: \text{'l1} \Rightarrow \text{'l1}' \Rightarrow \text{bool}) (L1' :: \text{'l1}' \Rightarrow \text{'l1}'' \Rightarrow \text{bool})$
 $(L2 :: \text{'l2} \Rightarrow \text{'l2}' \Rightarrow \text{bool}) (L2' :: \text{'l2}' \Rightarrow \text{'l2}'' \Rightarrow \text{bool})$
 $(L3 :: \text{'l3} \Rightarrow \text{'l3}' \Rightarrow \text{bool}) (L3' :: \text{'l3}' \Rightarrow \text{'l3}'' \Rightarrow \text{bool}).$
 $\text{rel-F } (L1 \text{ OO } L1') (L2 \text{ OO } L2') (L3 \text{ OO } L3') (Co1 \text{ OO } Co1') (Co2 \text{ OO } Co2')$
 $(Co3 \text{ OO } Co3')$
 $(\text{Contra1 OO Contra1}') (\text{Contra2 OO Contra2}') (\text{Contra3 OO Contra3}') \leq$
 $(\text{rel-F } L1 \text{ L2 } L3 \text{ Co1 } \text{Co2 } \text{Co3 } \text{Contra1 } \text{Contra2 } \text{Contra3} ::$
 $(\neg, \neg, \neg, \neg, \neg, \neg, \neg, \neg, \neg, \text{'f}) \text{ F} \Rightarrow \neg) \text{ OO}$
 $\text{rel-F } L1' \text{ L2}' \text{ L3}' \text{ Co1}' \text{ Co2}' \text{ Co3}' \text{ Contra1}' \text{ Contra2}' \text{ Contra3}')$

axiomatization where

$\text{rel-F-pos-distr-cond-eq:}$
 $\bigwedge \text{tytok. rel-F-pos-distr-cond } (=) (=) (=) (=) (=) (=) (=) (=) (=) (=) (=) (=)$
 tytok
and
 $\text{rel-F-neg-distr-cond-eq:}$
 $\bigwedge \text{tytok. rel-F-neg-distr-cond } (=) (=) (=) (=) (=) (=) (=) (=) (=) (=) (=) (=)$
 tytok

Restrictions to live variables.

definition $\text{rell-F } L1 \text{ L2 } L3 = \text{rel-F } L1 \text{ L2 } L3 (=) (=) (=) (=) (=) (=)$

definition $\text{mapl-F } l1 \text{ l2 } l3 = \text{map-F } l1 \text{ l2 } l3 \text{ id id id id id id id}$

typedecl $(\text{'co1}, \text{'co2}, \text{'co3}, \text{'contra1}, \text{'contra2}, \text{'contra3}, \text{'f}) \text{ Fbd}$

consts

$\text{set1-F} :: (\text{'l1}, \text{'l2}, \text{'l3}, \text{'co1}, \text{'co2}, \text{'co3}, \text{'contra1}, \text{'contra2}, \text{'contra3}, \text{'f}) \text{ F} \Rightarrow \text{'l1}$
 set
 $\text{set2-F} :: (\text{'l1}, \text{'l2}, \text{'l3}, \text{'co1}, \text{'co2}, \text{'co3}, \text{'contra1}, \text{'contra2}, \text{'contra3}, \text{'f}) \text{ F} \Rightarrow \text{'l2}$
 set
 $\text{set3-F} :: (\text{'l1}, \text{'l2}, \text{'l3}, \text{'co1}, \text{'co2}, \text{'co3}, \text{'contra1}, \text{'contra2}, \text{'contra3}, \text{'f}) \text{ F} \Rightarrow \text{'l3}$
 set
 $\text{bd-F} :: (\text{'co1}, \text{'co2}, \text{'co3}, \text{'contra1}, \text{'contra2}, \text{'contra3}, \text{'f}) \text{ Fbd rel}$

axiomatization where

$\text{set1-F-map: } \bigwedge l1 \text{ l2 } l3. \text{set1-F} \circ \text{mapl-F } l1 \text{ l2 } l3 = \text{image } l1 \circ \text{set1-F}$ **and**
 $\text{set2-F-map: } \bigwedge l1 \text{ l2 } l3. \text{set2-F} \circ \text{mapl-F } l1 \text{ l2 } l3 = \text{image } l2 \circ \text{set2-F}$ **and**
 $\text{set3-F-map: } \bigwedge l1 \text{ l2 } l3. \text{set3-F} \circ \text{mapl-F } l1 \text{ l2 } l3 = \text{image } l3 \circ \text{set3-F}$ **and**
 $\text{bd-F-card-order: card-order bd-F}$ **and**
 $\text{bd-F-cinfinite: cinfinite bd-F}$ **and**
 $\text{set1-F-bound: } \bigwedge x :: (\neg, \neg, \neg, \text{'co1}, \text{'co2}, \text{'co3}, \text{'contra1}, \text{'contra2}, \text{'contra3}, \text{'f}) \text{ F.}$
 $\text{card-of } (\text{set1-F } x) \leq o (\text{bd-F} :: (\text{'co1}, \text{'co2}, \text{'co3}, \text{'contra1}, \text{'contra2}, \text{'contra3},$
 $\text{'f}) \text{ Fbd rel})$ **and**
 $\text{set2-F-bound: } \bigwedge x :: (\neg, \neg, \neg, \text{'co1}, \text{'co2}, \text{'co3}, \text{'contra1}, \text{'contra2}, \text{'contra3}, \text{'f}) \text{ F.}$

$\text{card-of } (\text{set2-F } x) \leq o (\text{bd-F} :: ('co1, 'co2, 'co3, 'contra1, 'contra2, 'contra3,$
 'f) Fbd rel) **and**
 $\text{set3-F-bound: } \bigwedge x :: (-, -, -, 'co1, 'co2, 'co3, 'contra1, 'contra2, 'contra3, 'f) F.$
 $\text{card-of } (\text{set3-F } x) \leq o (\text{bd-F} :: ('co1, 'co2, 'co3, 'contra1, 'contra2, 'contra3,$
 'f) Fbd rel) **and**
 $\text{mapl-F-cong: } \bigwedge l1 l1' l2 l2' l3 l3' x.$
 $\llbracket \bigwedge z. z \in \text{set1-F } x \implies l1 z = l1' z; \bigwedge z. z \in \text{set2-F } x \implies l2 z = l2' z;$
 $\bigwedge z. z \in \text{set3-F } x \implies l3 z = l3' z \rrbracket \implies$
 $\text{mapl-F } l1 l2 l3 x = \text{mapl-F } l1' l2' l3' x$ **and**
 $\text{rell-F-mono-strong: } \bigwedge L1 L1' L2 L2' L3 L3' x y.$
 $\llbracket \text{rell-F } L1 L2 L3 x y;$
 $\bigwedge a b. a \in \text{set1-F } x \implies b \in \text{set1-F } y \implies L1 a b \implies L1' a b;$
 $\bigwedge a b. a \in \text{set2-F } x \implies b \in \text{set2-F } y \implies L2 a b \implies L2' a b;$
 $\bigwedge a b. a \in \text{set3-F } x \implies b \in \text{set3-F } y \implies L3 a b \implies L3' a b \rrbracket \implies$
 $\text{rell-F } L1' L2' L3' x y$

2.1.2 Derived rules

lemmas $\text{rel-F-mono}' = \text{rel-F-mono}[\text{THEN predicate2D, rotated } -1]$

lemma $\text{rel-F-eq-refl: rel-F } (=) (=) (=) (=) (=) (=) (=) (=) (=) x x$
 ⟨proof⟩

lemma $\text{map-F-id: map-F id id id id id id id id id id } x = x$
 ⟨proof⟩

lemmas $\text{map-F-rel-cong} = \text{map-F-parametric}[\text{unfolded rel-fun-def, rule-format, ro-}$
 $\text{tated } -1]$

lemma $\text{rell-F-mono: } \llbracket L1 \leq L1'; L2 \leq L2'; L3 \leq L3' \rrbracket \implies \text{rell-F } L1 L2 L3 \leq$
 $\text{rell-F } L1' L2' L3'$
 ⟨proof⟩

lemma $\text{mapl-F-id0: mapl-F id id id} = \text{id}$
 ⟨proof⟩

lemma $\text{mapl-F-id: mapl-F id id id } x = x$
 ⟨proof⟩

lemma $\text{mapl-F-comp: mapl-F } l1 l2 l3 \circ \text{mapl-F } l1' l2' l3' = \text{mapl-F } (l1 \circ l1') (l2$
 $\circ l2') (l3 \circ l3')$
 ⟨proof⟩

lemma $\text{map-F-mapl-F: map-F } l1 l2 l3 \text{ co1 co2 co3 contra1 contra2 contra3 } x =$
 $\text{map-F id id id co1 co2 co3 contra1 contra2 contra3 } (\text{mapl-F } l1 l2 l3 x)$
 ⟨proof⟩

lemma $\text{mapl-F-map-F: mapl-F } l1 l2 l3 (\text{map-F id id id co1 co2 co3 contra1 contra2}$
 $\text{contra3 } x) =$

map-F l1 l2 l3 co1 co2 co3 contra1 contra2 contra3 x
 ⟨proof⟩

Parametric mappers are unique:

lemma *rel-F-Grp-weak*: *rel-F (Grp UNIV l1) (Grp UNIV l2) (Grp UNIV l3)*
(Grp UNIV co1) (Grp UNIV co2) (Grp UNIV co3)
(Grp UNIV contra1)⁻¹⁻¹ (Grp UNIV contra2)⁻¹⁻¹ (Grp UNIV contra3)⁻¹⁻¹
 =
Grp UNIV (map-F l1 l2 l3 co1 co2 co3 contra1 contra2 contra3)
 ⟨proof⟩

lemmas

rel-F-pos-distr = rel-F-pos-distr-cond-def[THEN iffD1, rule-format] **and**
rel-F-neg-distr = rel-F-neg-distr-cond-def[THEN iffD1, rule-format]

lemma *rell-F-compp*:

rell-F (L1 OO L1') (L2 OO L2') (L3 OO L3') = rell-F L1 L2 L3 OO rell-F L1'
L2' L3'
 ⟨proof⟩

2.1.3 F is a BNF

lemma *rell-F-eq-onp*: *rell-F (eq-onp P1) (eq-onp P2) (eq-onp P3) =*
eq-onp (λx. (∀ z ∈ set1-F x. P1 z) ∧ (∀ z ∈ set2-F x. P2 z) ∧ (∀ z ∈ set3-F x. P3 z))
(is ?rel-eq-onp = ?eq-onp-pred)
 ⟨proof⟩

lemma *rell-F-Grp*: *rell-F (Grp A1 f1) (Grp A2 f2) (Grp A3 f3) =*
Grp {x. set1-F x ⊆ A1 ∧ set2-F x ⊆ A2 ∧ set3-F x ⊆ A3} (mapl-F f1 f2 f3)
 ⟨proof⟩

lemma *rell-F-compp-Grp*: *rell-F L1 L2 L3 =*
(Grp {x. set1-F x ⊆ {(x, y). L1 x y} ∧ set2-F x ⊆ {(x, y). L2 x y} ∧ set3-F x
⊆ {(x, y). L3 x y}})
(mapl-F fst fst fst)⁻¹⁻¹ OO
Grp {x. set1-F x ⊆ {(x, y). L1 x y} ∧ set2-F x ⊆ {(x, y). L2 x y} ∧ set3-F x ⊆
{(x, y). L3 x y}})
(mapl-F snd snd snd)
 ⟨proof⟩

lemma *F-in-rell*: *rell-F L1 L2 L3 = (λx y. ∃ z. (set1-F z ⊆ {(x, y). L1 x y} ∧*
set2-F z ⊆ {(x, y). L2 x y} ∧ set3-F z ⊆ {(x, y). L3 x y}) ∧
mapl-F fst fst fst z = x ∧ mapl-F snd snd snd z = y)
 ⟨proof⟩

bnf ('l1, 'l2, 'l3, 'co1, 'co2, 'co3, 'contra1, 'contra2, 'contra3, 'f) F
 map: *mapl-F*
 sets: *set1-F set2-F set3-F*
 bd: *bd-F :: ('co1, 'co2, 'co3, 'contra1, 'contra2, 'contra3, 'f) Fbd rel*

rel: *rel-F*
 ⟨*proof*⟩

2.1.4 Composition witness

consts

rel-F-witness :: ('l1 ⇒ 'l1'' ⇒ bool) ⇒ ('l2 ⇒ 'l2'' ⇒ bool) ⇒ ('l3 ⇒ 'l3'' ⇒ bool) ⇒
 ('co1 ⇒ 'co1' ⇒ bool) ⇒ ('co1'' ⇒ 'co1''' ⇒ bool) ⇒
 ('co2 ⇒ 'co2' ⇒ bool) ⇒ ('co2'' ⇒ 'co2''' ⇒ bool) ⇒
 ('co3 ⇒ 'co3' ⇒ bool) ⇒ ('co3'' ⇒ 'co3''' ⇒ bool) ⇒
 ('contra1 ⇒ 'contra1' ⇒ bool) ⇒ ('contra1'' ⇒ 'contra1''' ⇒ bool) ⇒
 ('contra2 ⇒ 'contra2' ⇒ bool) ⇒ ('contra2'' ⇒ 'contra2''' ⇒ bool) ⇒
 ('contra3 ⇒ 'contra3' ⇒ bool) ⇒ ('contra3'' ⇒ 'contra3''' ⇒ bool) ⇒
 ('l1, 'l2, 'l3, 'co1, 'co2, 'co3, 'contra1, 'contra2, 'contra3, 'f) F ×
 ('l1'', 'l2'', 'l3'', 'co1'', 'co2'', 'co3'', 'contra1'', 'contra2'', 'contra3'', 'f) F ⇒
 ('l1 × 'l1'', 'l2 × 'l2'', 'l3 × 'l3'', 'co1', 'co2', 'co3', 'contra1', 'contra2',
 'contra3',
 'f) F

specification (*rel-F-witness*)

rel-F-witness1: $\bigwedge L1 L2 L3 Co1 Co1' Co2 Co2' Co3 Co3'$
 Contra1 Contra1' Contra2 Contra2' Contra3 Contra3'
 (tytok :: ('l1 × ('l1 × 'l1'') × 'l1'' × 'l2 × ('l2 × 'l2'') × 'l2'' ×
 'l3 × ('l3 × 'l3'') × 'l3'' × 'f) itself)
 (x :: ('l1, 'l2, 'l3, -, -, -, -, -, 'f) F)
 (y :: ('l1'', 'l2'', 'l3'', -, -, -, -, -, 'f) F).
 [rel-F-neg-distr-cond Co1 Co1' Co2 Co2' Co3 Co3'
 Contra1 Contra1' Contra2 Contra2' Contra3 Contra3' tytok;
 rel-F L1 L2 L3 (Co1 OO Co1') (Co2 OO Co2') (Co3 OO Co3')
 (Contra1 OO Contra1') (Contra2 OO Contra2') (Contra3 OO Contra3')
 x y] ⇒⇒
 rel-F (λx (x', y). x' = x ∧ L1 x y) (λx (x', y). x' = x ∧ L2 x y)
 (λx (x', y). x' = x ∧ L3 x y) Co1 Co2 Co3 Contra1 Contra2 Contra3 x
 (rel-F-witness L1 L2 L3 Co1 Co1' Co2 Co2' Co3 Co3'
 Contra1 Contra1' Contra2 Contra2' Contra3 Contra3' (x, y))
rel-F-witness2: $\bigwedge L1 L2 L3 Co1 Co1' Co2 Co2' Co3 Co3'$
 Contra1 Contra1' Contra2 Contra2' Contra3 Contra3'
 (tytok :: ('l1 × ('l1 × 'l1'') × 'l1'' × 'l2 × ('l2 × 'l2'') × 'l2'' ×
 'l3 × ('l3 × 'l3'') × 'l3'' × 'f) itself)
 (x :: ('l1, 'l2, 'l3, -, -, -, -, -, 'f) F)
 (y :: ('l1'', 'l2'', 'l3'', -, -, -, -, -, 'f) F).
 [rel-F-neg-distr-cond Co1 Co1' Co2 Co2' Co3 Co3'
 Contra1 Contra1' Contra2 Contra2' Contra3 Contra3' tytok;
 rel-F L1 L2 L3 (Co1 OO Co1') (Co2 OO Co2') (Co3 OO Co3')
 (Contra1 OO Contra1') (Contra2 OO Contra2') (Contra3 OO Contra3')
 x y] ⇒⇒
 rel-F (λ(x, y') y. y' = y ∧ L1 x y) (λ(x, y') y. y' = y ∧ L2 x y)
 (λ(x, y') y. y' = y ∧ L3 x y) Co1' Co2' Co3' Contra1' Contra2' Contra3'

(*rel-F-witness* $L1\ L2\ L3\ Co1\ Co1'\ Co2\ Co2'\ Co3\ Co3'$
 $Contra1\ Contra1'\ Contra2\ Contra2'\ Contra3\ Contra3'$ (x, y)) y
 ⟨*proof*⟩

2.2 Second abstract BNF_{CC}

2.2.1 Axioms and basic definitions

typeddecl ($'l1, 'l2, 'co1, 'co2, 'contra1, 'contra2, 'f$) G

G has each two live, co, and contravariant parameters, and one fixed parameter.

consts

$rel-G :: ('l1 \Rightarrow 'l1' \Rightarrow bool) \Rightarrow ('l2 \Rightarrow 'l2' \Rightarrow bool) \Rightarrow$
 $('co1 \Rightarrow 'co1' \Rightarrow bool) \Rightarrow ('co2 \Rightarrow 'co2' \Rightarrow bool) \Rightarrow$
 $('contra1 \Rightarrow 'contra1' \Rightarrow bool) \Rightarrow ('contra2 \Rightarrow 'contra2' \Rightarrow bool) \Rightarrow$
 $('l1, 'l2, 'co1, 'co2, 'contra1, 'contra2, 'f) G \Rightarrow$
 $('l1', 'l2', 'co1', 'co2', 'contra1', 'contra2', 'f) G \Rightarrow bool$
 $map-G :: ('l1 \Rightarrow 'l1') \Rightarrow ('l2 \Rightarrow 'l2') \Rightarrow$
 $('co1 \Rightarrow 'co1') \Rightarrow ('co2 \Rightarrow 'co2') \Rightarrow$
 $('contra1 \Rightarrow 'contra1) \Rightarrow ('contra2 \Rightarrow 'contra2) \Rightarrow$
 $('l1, 'l2, 'co1, 'co2, 'contra1, 'contra2, 'f) G \Rightarrow$
 $('l1', 'l2', 'co1', 'co2', 'contra1', 'contra2', 'f) G$

axiomatization where

rel-G-mono [*mono*]:

$\bigwedge L1\ L1'\ L2\ L2'\ Co1\ Co1'\ Co2\ Co2'\ Contra1\ Contra1'\ Contra2\ Contra2'$
 $\llbracket L1 \leq L1'; L2 \leq L2'; Co1 \leq Co1'; Co2 \leq Co2'; Contra1' \leq Contra1;$
 $Contra2' \leq Contra2 \rrbracket \implies$
 $rel-G\ L1\ L2\ Co1\ Co2\ Contra1\ Contra2 \leq rel-G\ L1'\ L2'\ Co1'\ Co2'\ Contra1'$
 $Contra2'$ **and**

rel-G-eq: $rel-G (=) (=) (=) (=) (=) (=) (=) (=)$ **and**

rel-G-conversep: $\bigwedge L1\ L2\ Co1\ Co2\ Contra1\ Contra2$.

$rel-G\ L1^{-1-1}\ L2^{-1-1}\ Co1^{-1-1}\ Co2^{-1-1}\ Contra1^{-1-1}\ Contra2^{-1-1} = (rel-G$
 $L1\ L2\ Co1\ Co2\ Contra1\ Contra2)^{-1-1}$ **and**

map-G-id0: $map-G\ id\ id\ id\ id\ id\ id = id$ **and**

map-G-comp: $\bigwedge l1\ l1'\ l2\ l2'\ co1\ co1'\ co2\ co2'\ contra1\ contra1'\ contra2\ contra2'$.

$map-G\ l1\ l2\ co1\ co2\ contra1\ contra2 \circ map-G\ l1'\ l2'\ co1'\ co2'\ contra1'\ contra2'$
 $=$

$map-G\ (l1 \circ l1')\ (l2 \circ l2')\ (co1 \circ co1')\ (co2 \circ co2')$
 $(contra1' \circ contra1)\ (contra2' \circ contra2)$ **and**

map-G-parametric:

$\bigwedge L1\ L1'\ L2\ L2'\ Co1\ Co1'\ Co2\ Co2'\ Contra1\ Contra1'\ Contra2\ Contra2'$.

$rel-fun\ (rel-fun\ L1\ L1')\ (rel-fun\ (rel-fun\ L2\ L2')$
 $(rel-fun\ (rel-fun\ Co1\ Co1')\ (rel-fun\ (rel-fun\ Co2\ Co2')$
 $(rel-fun\ (rel-fun\ Contra1'\ Contra1)\ (rel-fun\ (rel-fun\ Contra2'\ Contra2)$
 $(rel-fun\ (rel-G\ L1\ L2\ Co1\ Co2\ Contra1\ Contra2)$
 $(rel-G\ L1'\ L2'\ Co1'\ Co2'\ Contra1'\ Contra2'))))))))$
 $map-G\ map-G$

definition *rel-G-pos-distr-cond* :: ('co1 ⇒ 'co1' ⇒ bool) ⇒ ('co1' ⇒ 'co1'' ⇒ bool) ⇒
('co2 ⇒ 'co2' ⇒ bool) ⇒ ('co2' ⇒ 'co2'' ⇒ bool) ⇒
('contra1 ⇒ 'contra1' ⇒ bool) ⇒ ('contra1' ⇒ 'contra1'' ⇒ bool) ⇒
('contra2 ⇒ 'contra2' ⇒ bool) ⇒ ('contra2' ⇒ 'contra2'' ⇒ bool) ⇒
('l1 × 'l1' × 'l1'' × 'l2 × 'l2' × 'l2'' × 'f) itself ⇒ bool **where**
rel-G-pos-distr-cond Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2' -
←→
(∀ (L1 :: 'l1 ⇒ 'l1' ⇒ bool) (L1' :: 'l1' ⇒ 'l1'' ⇒ bool)
(L2 :: 'l2 ⇒ 'l2' ⇒ bool) (L2' :: 'l2' ⇒ 'l2'' ⇒ bool).
(*rel-G* L1 L2 Co1 Co2 Contra1 Contra2 :: (-, -, -, -, -, 'f) G ⇒ -) OO
rel-G L1' L2' Co1' Co2' Contra1' Contra2' ≤
rel-G (L1 OO L1') (L2 OO L2') (Co1 OO Co1') (Co2 OO Co2')
(Contra1 OO Contra1') (Contra2 OO Contra2'))

definition *rel-G-neg-distr-cond* :: ('co1 ⇒ 'co1' ⇒ bool) ⇒ ('co1' ⇒ 'co1'' ⇒ bool) ⇒
('co2 ⇒ 'co2' ⇒ bool) ⇒ ('co2' ⇒ 'co2'' ⇒ bool) ⇒
('contra1 ⇒ 'contra1' ⇒ bool) ⇒ ('contra1' ⇒ 'contra1'' ⇒ bool) ⇒
('contra2 ⇒ 'contra2' ⇒ bool) ⇒ ('contra2' ⇒ 'contra2'' ⇒ bool) ⇒
('l1 × 'l1' × 'l1'' × 'l2 × 'l2' × 'l2'' × 'f) itself ⇒ bool **where**
rel-G-neg-distr-cond Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2' -
←→
(∀ (L1 :: 'l1 ⇒ 'l1' ⇒ bool) (L1' :: 'l1' ⇒ 'l1'' ⇒ bool)
(L2 :: 'l2 ⇒ 'l2' ⇒ bool) (L2' :: 'l2' ⇒ 'l2'' ⇒ bool).
rel-G (L1 OO L1') (L2 OO L2') (Co1 OO Co1') (Co2 OO Co2')
(Contra1 OO Contra1') (Contra2 OO Contra2') ≤
(*rel-G* L1 L2 Co1 Co2 Contra1 Contra2 :: (-, -, -, -, -, 'f) G ⇒ -) OO
rel-G L1' L2' Co1' Co2' Contra1' Contra2')

axiomatization where

rel-G-pos-distr-cond-eq:

∧ *tytok. rel-G-pos-distr-cond* (=) (=) (=) (=) (=) (=) (=) (=) *tytok* **and**

rel-G-neg-distr-cond-eq:

∧ *tytok. rel-G-neg-distr-cond* (=) (=) (=) (=) (=) (=) (=) (=) *tytok*

Restrictions to live variables.

definition *rell-G* L1 L2 = *rel-G* L1 L2 (=) (=) (=) (=)

definition *mapl-G* l1 l2 = *map-G* l1 l2 id id id id

typedecl ('co1, 'co2, 'contra1, 'contra2, 'f) Gbd

consts

set1-G :: ('l1, 'l2, 'co1, 'co2, 'contra1, 'contra2, 'f) G ⇒ 'l1 set

set2-G :: ('l1, 'l2, 'co1, 'co2, 'contra1, 'contra2, 'f) G ⇒ 'l2 set

bd-G :: ('co1, 'co2, 'contra1, 'contra2, 'f) Gbd rel

wit-G :: 'l2 ⇒ ('l1, 'l2, 'co1, 'co2, 'contra1, 'contra2, 'f) G

— non-emptiness witness for least fixpoint

axiomatization where

set1-G-map: $\bigwedge l1\ l2. \text{set1-G} \circ \text{mapl-G } l1\ l2 = \text{image } l1 \circ \text{set1-G}$ **and**
set2-G-map: $\bigwedge l1\ l2. \text{set2-G} \circ \text{mapl-G } l1\ l2 = \text{image } l2 \circ \text{set2-G}$ **and**
bd-G-card-order: *card-order* *bd-G* **and**
bd-G-cinfinite: *cinfinite* *bd-G* **and**
set1-G-bound: $\bigwedge x :: (-, -, 'co1, 'co2, 'contra1, 'contra2, 'f) G.$
card-of (*set1-G* *x*) $\leq o$ (*bd-G* :: ('co1, 'co2, 'contra1, 'contra2, 'f) *Gbd rel*) **and**
set2-G-bound: $\bigwedge x :: (-, -, 'co1, 'co2, 'contra1, 'contra2, 'f) G.$
card-of (*set2-G* *x*) $\leq o$ (*bd-G* :: ('co1, 'co2, 'contra1, 'contra2, 'f) *Gbd rel*) **and**
mapl-G-cong: $\bigwedge l1\ l1'\ l2\ l2'\ l3\ l3'\ x.$
 $\llbracket \bigwedge z. z \in \text{set1-G } x \implies l1\ z = l1'\ z; \bigwedge z. z \in \text{set2-G } x \implies l2\ z = l2'\ z \rrbracket \implies$
mapl-G *l1* *l2* *x* = *mapl-G* *l1'* *l2'* *x* **and**
rell-G-mono-strong: $\bigwedge L1\ L1'\ L2\ L2'\ x\ y.$
 $\llbracket \text{rell-G } L1\ L2\ x\ y;$
 $\bigwedge a\ b. a \in \text{set1-G } x \implies b \in \text{set1-G } y \implies L1\ a\ b \implies L1'\ a\ b;$
 $\bigwedge a\ b. a \in \text{set2-G } x \implies b \in \text{set2-G } y \implies L2\ a\ b \implies L2'\ a\ b \rrbracket \implies$
rell-G *L1'* *L2'* *x* *y* **and**
wit-G-set1: $\bigwedge l2\ x. x \in \text{set1-G } (wit-G\ l2) \implies \text{False}$ **and**
wit-G-set2: $\bigwedge l2\ x. x \in \text{set2-G } (wit-G\ l2) \implies x = l2$

2.2.2 Derived rules

lemmas *rel-G-mono'* = *rel-G-mono*[*THEN predicate2D, rotated -1*]

lemma *rel-G-eq-refl*: *rel-G* (=) (=) (=) (=) (=) (=) *x x*
{*proof*}

lemma *map-G-id*: *map-G id id id id id id x = x*
{*proof*}

lemmas *map-G-rel-cong* = *map-G-parametric*[*unfolded rel-fun-def, rule-format, rotated -1*]

lemma *rell-G-mono*: $\llbracket L1 \leq L1'; L2 \leq L2' \rrbracket \implies \text{rell-G } L1\ L2 \leq \text{rell-G } L1'\ L2'$
{*proof*}

lemma *mapl-G-id0*: *mapl-G id id = id*
{*proof*}

lemma *mapl-G-id*: *mapl-G id id x = x*
{*proof*}

lemma *mapl-G-comp*: *mapl-G* *l1* *l2* \circ *mapl-G* *l1'* *l2'* = *mapl-G* (*l1* \circ *l1'*) (*l2* \circ *l2'*)
{*proof*}

lemma *map-G-mapl-G*:
map-G *l1* *l2* *co1* *co2* *contra1* *contra2* *x* = *map-G id id co1 co2 contra1 contra2*
(*mapl-G* *l1* *l2* *x*)

<proof>

lemma *mapl-G-map-G*:

$\text{mapl-G } l1 \ l2 \ (\text{map-G } id \ id \ co1 \ co2 \ contra1 \ contra2 \ x) = \text{map-G } l1 \ l2 \ co1 \ co2 \ contra1 \ contra2 \ x$

<proof>

Parametric mappers are unique:

lemma *rel-G-Grp-weak*: $\text{rel-G } (\text{Grp } UNIV \ l1) \ (\text{Grp } UNIV \ l2) \ (\text{Grp } UNIV \ co1) \ (\text{Grp } UNIV \ co2)$

$(\text{Grp } UNIV \ contra1)^{-1-1} \ (\text{Grp } UNIV \ contra2)^{-1-1} = \text{Grp } UNIV \ (\text{map-G } l1 \ l2 \ co1 \ co2 \ contra1 \ contra2)$

<proof>

lemmas

$\text{rel-G-pos-distr} = \text{rel-G-pos-distr-cond-def}[\text{THEN } iffD1, \text{rule-format}]$ **and**

$\text{rel-G-neg-distr} = \text{rel-G-neg-distr-cond-def}[\text{THEN } iffD1, \text{rule-format}]$

lemma *rell-G-compp*:

$\text{rell-G } (L1 \ OO \ L1') \ (L2 \ OO \ L2') = \text{rell-G } L1 \ L2 \ OO \ \text{rell-G } L1' \ L2'$

<proof>

2.2.3 G is a BNF

lemma *rell-G-eq-onp*:

$\text{rell-G } (\text{eq-onp } P1) \ (\text{eq-onp } P2) = \text{eq-onp } (\lambda x. (\forall z \in \text{set1-G } x. P1 \ z) \wedge (\forall z \in \text{set2-G } x. P2 \ z))$

(**is** $?rel\text{-eq-onp} = ?eq\text{-onp-pred}$)

<proof>

lemma *rell-G-Grp*:

$\text{rell-G } (\text{Grp } A1 \ f1) \ (\text{Grp } A2 \ f2) = \text{Grp } \{x. \text{set1-G } x \subseteq A1 \wedge \text{set2-G } x \subseteq A2\} \ (\text{mapl-G } f1 \ f2)$

<proof>

lemma *rell-G-compp-Grp*: $\text{rell-G } L1 \ L2 =$

$(\text{Grp } \{x. \text{set1-G } x \subseteq \{(x, y). L1 \ x \ y\} \wedge \text{set2-G } x \subseteq \{(x, y). L2 \ x \ y\}\} \ (\text{mapl-G } fst \ fst))^{-1-1} \ OO$

$\text{Grp } \{x. \text{set1-G } x \subseteq \{(x, y). L1 \ x \ y\} \wedge \text{set2-G } x \subseteq \{(x, y). L2 \ x \ y\}\} \ (\text{mapl-G } snd \ snd)$

<proof>

lemma *G-in-rell*: $\text{rell-G } L1 \ L2 = (\lambda x \ y. \exists z. (\text{set1-G } z \subseteq \{(x, y). L1 \ x \ y\} \wedge$

$\text{set2-G } z \subseteq \{(x, y). L2 \ x \ y\}) \wedge \text{mapl-G } fst \ fst \ z = x \wedge \text{mapl-G } snd \ snd \ z = y)$

<proof>

bnf (*'l1, 'l2, 'co1, 'co2, 'contra1, 'contra2, 'f*) *G*

map: *mapl-G*

sets: *set1-G set2-G*

bd: *bd-G* :: ('co1, 'co2, 'contra1, 'contra2, 'f) *Gbd rel*
wits: *wit-G*
rel: *rell-G*
 ⟨*proof*⟩

2.2.4 Composition witness

consts

rel-G-witness :: ('l1 ⇒ 'l1'' ⇒ bool) ⇒ ('l2 ⇒ 'l2'' ⇒ bool) ⇒
 ('co1 ⇒ 'co1' ⇒ bool) ⇒ ('co1' ⇒ 'co1'' ⇒ bool) ⇒
 ('co2 ⇒ 'co2' ⇒ bool) ⇒ ('co2' ⇒ 'co2'' ⇒ bool) ⇒
 ('contra1 ⇒ 'contra1' ⇒ bool) ⇒ ('contra1' ⇒ 'contra1'' ⇒ bool) ⇒
 ('contra2 ⇒ 'contra2' ⇒ bool) ⇒ ('contra2' ⇒ 'contra2'' ⇒ bool) ⇒
 ('l1, 'l2, 'co1, 'co2, 'contra1, 'contra2, 'f) *G* ×
 ('l1'', 'l2'', 'co1'', 'co2'', 'contra1'', 'contra2'', 'f) *G* ⇒
 ('l1 × 'l1'', 'l2 × 'l2'', 'co1', 'co2', 'contra1', 'contra2', 'f) *G*

specification (*rel-G-witness*)

rel-G-witness1: $\bigwedge L1 L2 Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2'$
 (*tytok* :: ('l1 × ('l1 × 'l1'') × 'l1'' × 'l2 × ('l2 × 'l2'') × 'l2'' × 'f) *itself*)
 (*x* :: ('l1, 'l2, -, -, -, 'f) *G*) (*y* :: ('l1'', 'l2'', -, -, -, 'f) *G*).
 [*rel-G-neg-distr-cond Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2'*
tytok;
rel-G L1 L2 (Co1 OO Co1') (Co2 OO Co2') (Contra1 OO Contra1') (Contra2
OO Contra2') *x y*] ⇒
rel-G ($\lambda x (x', y). x' = x \wedge L1 x y$) ($\lambda x (x', y). x' = x \wedge L2 x y$) *Co1 Co2*
Contra1 Contra2 x
 (*rel-G-witness L1 L2 Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2'*
 (*x, y*))
rel-G-witness2: $\bigwedge L1 L2 Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2'$
 (*tytok* :: ('l1 × ('l1 × 'l1'') × 'l1'' × 'l2 × ('l2 × 'l2'') × 'l2'' × 'f) *itself*)
 (*x* :: ('l1, 'l2, -, -, -, 'f) *G*) (*y* :: ('l1'', 'l2'', -, -, -, 'f) *G*).
 [*rel-G-neg-distr-cond Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2'*
tytok;
rel-G L1 L2 (Co1 OO Co1') (Co2 OO Co2') (Contra1 OO Contra1') (Contra2
OO Contra2') *x y*] ⇒
rel-G ($\lambda(x, y') y. y' = y \wedge L1 x y$) ($\lambda(x, y') y. y' = y \wedge L2 x y$) *Co1' Co2'*
Contra1' Contra2'
 (*rel-G-witness L1 L2 Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2'*
 (*x, y*)) *y*
 ⟨*proof*⟩

end

3 Simple operations: demotion, merging, composition

theory *Composition* imports

Axiomatised-BNF-CC

begin

We illustrate the composition of BNF_{CCs} with one example for each kind of parameters (live/co-/contravariant/fixed). We do not show demotion and merging in isolation, as the examples for composition use these operations, too.

3.1 Composition in a live position

type-synonym

$(l1, l2, l3, co1, co2, co3, co4, contra1, contra2, contra3, contra4, f1, f2)$ $\text{FGL} =$
 $((l1, l2, co1, co2, contra1, contra2, f1) G,$
 $l1, l3, co1, co3, co4, contra1, contra3, contra4, f2) F$

The type variables $l1$, $co1$ and $contra1$ have each been merged.

definition $\text{rel-FGL } L1 L2 L3 Co1 Co2 Co3 Co4 Contra1 Contra2 Contra3 Contra4 =$
 $=$
 $\text{rel-F } (\text{rel-G } L1 L2 Co1 Co2 Contra1 Contra2) L1 L3 Co1 Co3 Co4 Contra1$
 $Contra3 Contra4$

definition $\text{map-FGL } l1 l2 l3 co1 co2 co3 co4 contra1 contra2 contra3 contra4 =$
 $\text{map-F } (\text{map-G } l1 l2 co1 co2 contra1 contra2) l1 l3 co1 co3 co4 contra1 contra3$
 $contra4$

lemma rel-FGL-mono :

$\llbracket L1 \leq L1'; L2 \leq L2'; L3 \leq L3'; Co1 \leq Co1'; Co2 \leq Co2'; Co3 \leq Co3'; Co4 \leq Co4';$
 $Contra1' \leq Contra1; Contra2' \leq Contra2; Contra3' \leq Contra3; Contra4' \leq$
 $Contra4 \rrbracket \implies$
 $\text{rel-FGL } L1 L2 L3 Co1 Co2 Co3 Co4 Contra1 Contra2 Contra3 Contra4 \leq$
 $\text{rel-FGL } L1' L2' L3' Co1' Co2' Co3' Co4' Contra1' Contra2' Contra3' Contra4'$
 $\langle \text{proof} \rangle$

lemma rel-FGL-eq : $\text{rel-FGL } (=) (=) (=) (=) (=) (=) (=) (=) (=) (=) (=) (=)$
 $\langle \text{proof} \rangle$

lemma rel-FGL-conversep :

$\text{rel-FGL } L1^{-1-1} L2^{-1-1} L3^{-1-1} Co1^{-1-1} Co2^{-1-1} Co3^{-1-1} Co4^{-1-1} Con-$
 $tra1^{-1-1} Contra2^{-1-1} Contra3^{-1-1} Contra4^{-1-1} =$
 $(\text{rel-FGL } L1 L2 L3 Co1 Co2 Co3 Co4 Contra1 Contra2 Contra3 Contra4)^{-1-1}$
 $\langle \text{proof} \rangle$

lemma map-FGL-id0 : $\text{map-FGL } id id id id id id id id id id id id = id$
 $\langle \text{proof} \rangle$

lemma map-FGL-comp : $\text{map-FGL } l1 l2 l3 co1 co2 co3 co4 contra1 contra2 contra3$
 $contra4 \circ$

$\text{map-FG}l\ l1'\ l2'\ l3'\ co1'\ co2'\ co3'\ co4'\ contra1'\ contra2'\ contra3'\ contra4' =$
 $\text{map-FG}l\ (l1 \circ l1')\ (l2 \circ l2')\ (l3 \circ l3')\ (co1 \circ co1')\ (co2 \circ co2')\ (co3 \circ co3')$
 $(co4 \circ co4')$
 $(contra1' \circ contra1)\ (contra2' \circ contra2)\ (contra3' \circ contra3)\ (contra4' \circ$
 $contra4)$
 $\langle \text{proof} \rangle$

lemma *map-FG}l-parametric:*

$\text{rel-fun}\ (\text{rel-fun}\ L1\ L1')\ (\text{rel-fun}\ (\text{rel-fun}\ L2\ L2')\ (\text{rel-fun}\ (\text{rel-fun}\ L3\ L3')$
 $(\text{rel-fun}\ (\text{rel-fun}\ Co1\ Co1')\ (\text{rel-fun}\ (\text{rel-fun}\ Co2\ Co2')$
 $(\text{rel-fun}\ (\text{rel-fun}\ Co3\ Co3')\ (\text{rel-fun}\ (\text{rel-fun}\ Co4\ Co4')$
 $(\text{rel-fun}\ (\text{rel-fun}\ Contra1'\ Contra1)\ (\text{rel-fun}\ (\text{rel-fun}\ Contra2'\ Contra2)$
 $(\text{rel-fun}\ (\text{rel-fun}\ Contra3'\ Contra3)\ (\text{rel-fun}\ (\text{rel-fun}\ Contra4'\ Contra4)$
 $(\text{rel-fun}\ (\text{rel-FG}l\ L1\ L2\ L3\ Co1\ Co2\ Co3\ Co4\ Contra1\ Contra2\ Contra3\ Contra4)$
 $(\text{rel-FG}l\ L1'\ L2'\ L3'\ Co1'\ Co2'\ Co3'\ Co4'\ Contra1'\ Contra2'\ Contra3'\ Con-$
 $tra4'))))))))))))$
 $\text{map-FG}l\ \text{map-FG}l$
 $\langle \text{proof} \rangle$

definition *rel-FG}l-pos-distr-cond* :: $(co1 \Rightarrow co1' \Rightarrow \text{bool}) \Rightarrow (co1' \Rightarrow co1'' \Rightarrow$
 $\text{bool}) \Rightarrow$
 $(co2 \Rightarrow co2' \Rightarrow \text{bool}) \Rightarrow (co2' \Rightarrow co2'' \Rightarrow \text{bool}) \Rightarrow$
 $(co3 \Rightarrow co3' \Rightarrow \text{bool}) \Rightarrow (co3' \Rightarrow co3'' \Rightarrow \text{bool}) \Rightarrow$
 $(co4 \Rightarrow co4' \Rightarrow \text{bool}) \Rightarrow (co4' \Rightarrow co4'' \Rightarrow \text{bool}) \Rightarrow$
 $(contra1 \Rightarrow contra1' \Rightarrow \text{bool}) \Rightarrow (contra1' \Rightarrow contra1'' \Rightarrow \text{bool}) \Rightarrow$
 $(contra2 \Rightarrow contra2' \Rightarrow \text{bool}) \Rightarrow (contra2' \Rightarrow contra2'' \Rightarrow \text{bool}) \Rightarrow$
 $(contra3 \Rightarrow contra3' \Rightarrow \text{bool}) \Rightarrow (contra3' \Rightarrow contra3'' \Rightarrow \text{bool}) \Rightarrow$
 $(contra4 \Rightarrow contra4' \Rightarrow \text{bool}) \Rightarrow (contra4' \Rightarrow contra4'' \Rightarrow \text{bool}) \Rightarrow$
 $(l1 \times l1' \times l1'' \times l2 \times l2' \times l2'' \times l3 \times l3' \times l3'' \times f1 \times f2)\ \text{itself}$
 $\Rightarrow \text{bool}$

where

$\text{rel-FG}l\text{-pos-distr-cond}\ Co1\ Co1'\ Co2\ Co2'\ Co3\ Co3'\ Co4\ Co4'$
 $Contra1\ Contra1'\ Contra2\ Contra2'\ Contra3\ Contra3'\ Contra4\ Contra4' - \longleftrightarrow$
 $(\forall (L1 :: l1 \Rightarrow l1' \Rightarrow \text{bool})\ (L1' :: l1' \Rightarrow l1'' \Rightarrow \text{bool})$
 $(L2 :: l2 \Rightarrow l2' \Rightarrow \text{bool})\ (L2' :: l2' \Rightarrow l2'' \Rightarrow \text{bool})$
 $(L3 :: l3 \Rightarrow l3' \Rightarrow \text{bool})\ (L3' :: l3' \Rightarrow l3'' \Rightarrow \text{bool}).$
 $(\text{rel-FG}l\ L1\ L2\ L3\ Co1\ Co2\ Co3\ Co4\ Contra1\ Contra2\ Contra3\ Contra4 ::$
 $(-, -, -, -, -, -, -, -, -, f1, f2)\ \text{FG}l \Rightarrow -)\ \text{OO}$
 $\text{rel-FG}l\ L1'\ L2'\ L3'\ Co1'\ Co2'\ Co3'\ Co4'\ Contra1'\ Contra2'\ Contra3'$
 $Contra4' \leq$
 $\text{rel-FG}l\ (L1\ \text{OO}\ L1')\ (L2\ \text{OO}\ L2')\ (L3\ \text{OO}\ L3')\ (Co1\ \text{OO}\ Co1')\ (Co2\ \text{OO}$
 $Co2')\ (Co3\ \text{OO}\ Co3')\ (Co4\ \text{OO}\ Co4')$
 $(Contra1\ \text{OO}\ Contra1')\ (Contra2\ \text{OO}\ Contra2')\ (Contra3\ \text{OO}\ Contra3')$
 $(Contra4\ \text{OO}\ Contra4')$

definition *rel-FG}l-neg-distr-cond* :: $(co1 \Rightarrow co1' \Rightarrow \text{bool}) \Rightarrow (co1' \Rightarrow co1'' \Rightarrow$
 $\text{bool}) \Rightarrow$
 $(co2 \Rightarrow co2' \Rightarrow \text{bool}) \Rightarrow (co2' \Rightarrow co2'' \Rightarrow \text{bool}) \Rightarrow$
 $(co3 \Rightarrow co3' \Rightarrow \text{bool}) \Rightarrow (co3' \Rightarrow co3'' \Rightarrow \text{bool}) \Rightarrow$

```

('co4 ⇒ 'co4' ⇒ bool) ⇒ ('co4' ⇒ 'co4'' ⇒ bool) ⇒
('contra1 ⇒ 'contra1' ⇒ bool) ⇒ ('contra1' ⇒ 'contra1'' ⇒ bool) ⇒
('contra2 ⇒ 'contra2' ⇒ bool) ⇒ ('contra2' ⇒ 'contra2'' ⇒ bool) ⇒
('contra3 ⇒ 'contra3' ⇒ bool) ⇒ ('contra3' ⇒ 'contra3'' ⇒ bool) ⇒
('contra4 ⇒ 'contra4' ⇒ bool) ⇒ ('contra4' ⇒ 'contra4'' ⇒ bool) ⇒
('l1 × 'l1' × 'l1'' × 'l2 × 'l2' × 'l2'' × 'l3 × 'l3' × 'l3'' × 'f1 × 'f2) itself
⇒ bool
where
rel-FGL-neg-distr-cond Co1 Co1' Co2 Co2' Co3 Co3' Co4 Co4'
  Contra1 Contra1' Contra2 Contra2' Contra3 Contra3' Contra4 Contra4' - <—>
(∀ (L1 :: 'l1 ⇒ 'l1' ⇒ bool) (L1' :: 'l1' ⇒ 'l1'' ⇒ bool)
  (L2 :: 'l2 ⇒ 'l2' ⇒ bool) (L2' :: 'l2' ⇒ 'l2'' ⇒ bool)
  (L3 :: 'l3 ⇒ 'l3' ⇒ bool) (L3' :: 'l3' ⇒ 'l3'' ⇒ bool).
rel-FGL (L1 OO L1') (L2 OO L2') (L3 OO L3')
  (Co1 OO Co1') (Co2 OO Co2') (Co3 OO Co3') (Co4 OO Co4')
  (Contra1 OO Contra1') (Contra2 OO Contra2') (Contra3 OO Contra3')
(Contra4 OO Contra4') ≤
  (rel-FGL L1 L2 L3 Co1 Co2 Co3 Co4 Contra1 Contra2 Contra3 Contra4 ::
    (¬, ¬, ¬, ¬, ¬, ¬, ¬, ¬, ¬, ¬, 'f1, 'f2) FGL ⇒ -) OO
rel-FGL L1' L2' L3' Co1' Co2' Co3' Co4' Contra1' Contra2' Contra3'
Contra4')

```

Sufficient conditions for subdistributivity over relation composition.

lemma *rel-FGL-pos-distr-imp*:

```

fixes Co1 :: 'co1 ⇒ 'co1' ⇒ bool and Co1' :: 'co1' ⇒ 'co1'' ⇒ bool
and Co2 :: 'co2 ⇒ 'co2' ⇒ bool and Co2' :: 'co2' ⇒ 'co2'' ⇒ bool
and Contra1 :: 'contra1 ⇒ 'contra1' ⇒ bool and Contra1' :: 'contra1' ⇒
'contra1'' ⇒ bool
and Contra2 :: 'contra2 ⇒ 'contra2' ⇒ bool and Contra2' :: 'contra2' ⇒
'contra2'' ⇒ bool
and tytok-F :: (('l1, 'l2, 'co1, 'co2, 'contra1, 'contra2, 'f1) G ×
('l1', 'l2', 'co1', 'co2', 'contra1', 'contra2', 'f1) G ×
('l1'', 'l2'', 'co1'', 'co2'', 'contra1'', 'contra2'', 'f1) G ×
'l1 × 'l1' × 'l1'' × 'l2 × 'l2' × 'l2'' × 'l3 × 'l3' × 'l3'' × 'f1) itself
and tytok-G :: ('l1 × 'l1' × 'l1'' × 'l2 × 'l2' × 'l2'' × 'f1) itself
and tytok-FGL :: ('l1 × 'l1' × 'l1'' × 'l2 × 'l2' × 'l2'' × 'l3 × 'l3' × 'l3'' ×
'f1 × 'f2) itself
assumes rel-F-pos-distr-cond Co1 Co1' Co3 Co3' Co4 Co4'
  Contra1 Contra1' Contra3 Contra3' Contra4 Contra4' tytok-F
and rel-G-pos-distr-cond Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Con-
tra2' tytok-G
shows rel-FGL-pos-distr-cond Co1 Co1' Co2 Co2' Co3 Co3' Co4 Co4'
  Contra1 Contra1' Contra2 Contra2' Contra3 Contra3' Contra4 Contra4' ty-
tok-FGL
⟨proof⟩

```

lemma *rel-FGL-neg-distr-imp*:

```

fixes Co1 :: 'co1 ⇒ 'co1' ⇒ bool and Co1' :: 'co1' ⇒ 'co1'' ⇒ bool
and Co2 :: 'co2 ⇒ 'co2' ⇒ bool and Co2' :: 'co2' ⇒ 'co2'' ⇒ bool

```

and *Contra1* :: 'contra1 ⇒ 'contra1' ⇒ bool **and** *Contra1'* :: 'contra1' ⇒ 'contra1'' ⇒ bool
and *Contra2* :: 'contra2 ⇒ 'contra2' ⇒ bool **and** *Contra2'* :: 'contra2' ⇒ 'contra2'' ⇒ bool
and *tytok-F* :: (('l1, 'l2, 'co1, 'co2, 'contra1, 'contra2, 'f1) G × ('l1', 'l2', 'co1', 'co2', 'contra1', 'contra2', 'f1) G × ('l1'', 'l2'', 'co1'', 'co2'', 'contra1'', 'contra2'', 'f1) G × 'l1 × 'l1' × 'l1'' × 'l3 × 'l3' × 'l3'' × 'f2) itself
and *tytok-G* :: ('l1 × 'l1' × 'l1'' × 'l2 × 'l2' × 'l2'' × 'f1) itself
and *tytok-FGl* :: ('l1 × 'l1' × 'l1'' × 'l2 × 'l2' × 'l2'' × 'l3 × 'l3' × 'l3'' × 'f1 × 'f2) itself
assumes *rel-F-neg-distr-cond* *Co1 Co1' Co3 Co3' Co4 Co4'*
Contra1 Contra1' Contra3 Contra3' Contra4 Contra4' tytok-F
and *rel-G-neg-distr-cond* *Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2' tytok-G*
shows *rel-FGl-neg-distr-cond* *Co1 Co1' Co2 Co2' Co3 Co3' Co4 Co4'*
Contra1 Contra1' Contra2 Contra2' Contra3 Contra3' Contra4 Contra4' tytok-FGl
⟨proof⟩

lemma *rel-FGl-pos-distr-cond-eq*:

fixes *tytok* :: ('l1 × 'l1' × 'l1'' × 'l2 × 'l2' × 'l2'' × 'l3 × 'l3' × 'l3'' × 'f1 × 'f2) itself
shows *rel-FGl-pos-distr-cond* (=) (=) (=) (=) (=) (=) (=) (=) (=) (=) (=) (=) *tytok*
⟨proof⟩

lemma *rel-FGl-neg-distr-cond-eq*:

fixes *tytok* :: ('l1 × 'l1' × 'l1'' × 'l2 × 'l2' × 'l2'' × 'l3 × 'l3' × 'l3'' × 'f1 × 'f2) itself
shows *rel-FGl-neg-distr-cond* (=) (=) (=) (=) (=) (=) (=) (=) (=) (=) (=) (=) *tytok*
⟨proof⟩

definition *rell-FGl* *L1 L2 L3* = *rel-FGl* *L1 L2 L3* (=) (=) (=) (=) (=) (=) (=) (=)

definition *mapl-FGl* *l1 l2 l3* = *map-FGl* *l1 l2 l3 id id id id id id id id*

type-synonym ('co1, 'co2, 'co3, 'co4, 'contra1, 'contra2, 'contra3, 'contra4, 'f1, 'f2) *FGlbd* =

('co1, 'co3, 'co4, 'contra1, 'contra3, 'contra4, 'f2) *Fbd* × ('co1, 'co2, 'contra1, 'contra2, 'f1) *Gbd* + ('co1, 'co3, 'co4, 'contra1, 'contra3, 'contra4, 'f2) *Fbd*

definition *set1-FGl* :: ('l1, 'l2, 'l3, 'co1, 'co2, 'co3, 'co4, 'contra1, 'contra2, 'contra3, 'contra4, 'f1, 'f2) *FGl* ⇒ 'l1 *set* **where**
set1-FGl *x* = (⋃ *y* ∈ *set1-F* *x*. *set1-G* *y*) ∪ *set2-F* *x*

definition *set2-FGl* :: ('l1, 'l2, 'l3, 'co1, 'co2, 'co3, 'co4,

'contra1, 'contra2, 'contra3, 'contra4, 'f1, 'f2) FGl \Rightarrow 'l2 set **where**
 set2-FGl x = ($\bigcup y \in \text{set1-F } x. \text{set2-G } y$)

definition set3-FGl :: ('l1, 'l2, 'l3, 'co1, 'co2, 'co3, 'co4,
 'contra1, 'contra2, 'contra3, 'contra4, 'f1, 'f2) FGl \Rightarrow 'l3 set **where**
 set3-FGl x = set3-F x

definition

bd-FGl :: ('co1, 'co2, 'co3, 'co4, 'contra1, 'contra2, 'contra3, 'contra4, 'f1, 'f2)
 FGldb rel

where bd-FGl = bd-F *c bd-G +c bd-F

lemma set1-FGl-map: set1-FGl \circ mapl-FGl l1 l2 l3 = image l1 \circ set1-FGl
 <proof>

lemma set2-FGl-map: set2-FGl \circ mapl-FGl l1 l2 l3 = image l2 \circ set2-FGl
 <proof>

lemma set3-FGl-map: set3-FGl \circ mapl-FGl l1 l2 l3 = image l3 \circ set3-FGl
 <proof>

lemma bd-FGl-card-order: card-order bd-FGl
 <proof>

lemma bd-FGl-cinfinite: cinfinite bd-FGl
 <proof>

lemma

fixes x :: (-, -, -, 'co1, 'co2, 'co3, 'co4, 'contra1, 'contra2, 'contra3, 'contra4,
 'f1, 'f2) FGl

shows set1-FGl-bound: card-of (set1-FGl x) $\leq o$

(bd-FGl :: ('co1, 'co2, 'co3, 'co4, 'contra1, 'contra2, 'contra3, 'contra4, 'f1,
 'f2) FGldb rel)

and set2-FGl-bound: card-of (set2-FGl x) $\leq o$

(bd-FGl :: ('co1, 'co2, 'co3, 'co4, 'contra1, 'contra2, 'contra3, 'contra4, 'f1,
 'f2) FGldb rel)

and set3-FGl-bound: card-of (set3-FGl x) $\leq o$

(bd-FGl :: ('co1, 'co2, 'co3, 'co4, 'contra1, 'contra2, 'contra3, 'contra4, 'f1,
 'f2) FGldb rel)

<proof>

lemma mapl-FGl-cong:

assumes $\bigwedge z. z \in \text{set1-FGl } x \Rightarrow l1 z = l1' z$ **and** $\bigwedge z. z \in \text{set2-FGl } x \Rightarrow l2 z = l2' z$

and $\bigwedge z. z \in \text{set3-FGl } x \Rightarrow l3 z = l3' z$

shows mapl-FGl l1 l2 l3 x = mapl-FGl l1' l2' l3' x

<proof>

lemma rell-FGl-mono-strong:

assumes $rel-FGl\ L1\ L2\ L3\ x\ y$
and $\bigwedge a\ b. a \in set1-FGl\ x \implies b \in set1-FGl\ y \implies L1\ a\ b \implies L1'\ a\ b$
and $\bigwedge a\ b. a \in set2-FGl\ x \implies b \in set2-FGl\ y \implies L2\ a\ b \implies L2'\ a\ b$
and $\bigwedge a\ b. a \in set3-FGl\ x \implies b \in set3-FGl\ y \implies L3\ a\ b \implies L3'\ a\ b$
shows $rel-FGl\ L1'\ L2'\ L3'\ x\ y$
 $\langle proof \rangle$

3.2 Composition in a covariant position

type-synonym

$(l1, 'co1, 'co2, 'co3, 'co4, 'co5, 'co6, 'contra1, 'contra2, 'contra3, 'contra4, 'f1,$
 $'f2)\ FGco =$
 $(l1, 'co1, 'co5, ('co1, 'co2, 'co3, 'co4, 'contra1, 'contra2, 'f1)\ G, 'co3, 'co6,$
 $'contra1, 'contra3, 'contra4, 'f2)\ F$

The type variables $'co1$, $'co3$ and $'contra1$ have each been merged.

definition $rel-FGco\ L1\ Co1\ Co2\ Co3\ Co4\ Co5\ Co6\ Contra1\ Contra2\ Contra3$
 $Contra4 =$
 $rel-F\ L1\ Co1\ Co5\ (rel-G\ Co1\ Co2\ Co3\ Co4\ Contra1\ Contra2)\ Co3\ Co6\ Contra1$
 $Contra3\ Contra4$

definition $map-FGco\ l1\ co1\ co2\ co3\ co4\ co5\ co6\ contra1\ contra2\ contra3\ contra4$
 $=$
 $map-F\ l1\ co1\ co5\ (map-G\ co1\ co2\ co3\ co4\ contra1\ contra2)\ co3\ co6\ contra1$
 $contra3\ contra4$

lemma $rel-FGco-mono$:

$\llbracket L1 \leq L1'; Co1 \leq Co1'; Co2 \leq Co2'; Co3 \leq Co3'; Co4 \leq Co4'; Co5 \leq Co5';$
 $Co6 \leq Co6';$
 $Contra1' \leq Contra1; Contra2' \leq Contra2; Contra3' \leq Contra3; Contra4' \leq$
 $Contra4 \rrbracket \implies$
 $rel-FGco\ L1\ Co1\ Co2\ Co3\ Co4\ Co5\ Co6\ Contra1\ Contra2\ Contra3\ Contra4 \leq$
 $rel-FGco\ L1'\ Co1'\ Co2'\ Co3'\ Co4'\ Co5'\ Co6'\ Contra1'\ Contra2'\ Contra3'$
 $Contra4'$
 $\langle proof \rangle$

lemma $rel-FGco-eq$: $rel-FGco\ (=)\ (=)\ (=)\ (=)\ (=)\ (=)\ (=)\ (=)\ (=)\ (=)\ (=)\ (=)$
 $\langle proof \rangle$

lemma $rel-FGco-conversep$:

$rel-FGco\ L1^{-1-1}\ Co1^{-1-1}\ Co2^{-1-1}\ Co3^{-1-1}\ Co4^{-1-1}\ Co5^{-1-1}\ Co6^{-1-1}$
 $Contra1^{-1-1}\ Contra2^{-1-1}\ Contra3^{-1-1}\ Contra4^{-1-1} =$
 $(rel-FGco\ L1\ Co1\ Co2\ Co3\ Co4\ Co5\ Co6\ Contra1\ Contra2\ Contra3\ Contra4)^{-1-1}$
 $\langle proof \rangle$

lemma $map-FGco-id0$: $map-FGco\ id\ id\ id\ id\ id\ id\ id\ id\ id\ id\ id\ id = id$
 $\langle proof \rangle$

lemma *map-FGco-comp*: $\text{map-FGco } l1 \ co1 \ co2 \ co3 \ co4 \ co5 \ co6 \ \text{contra1} \ \text{contra2} \ \text{contra3} \ \text{contra4} \ \circ$
 $\text{map-FGco } l1' \ co1' \ co2' \ co3' \ co4' \ co5' \ co6' \ \text{contra1}' \ \text{contra2}' \ \text{contra3}' \ \text{contra4}'$
 $=$
 $\text{map-FGco } (l1 \ \circ \ l1') \ (co1 \ \circ \ co1') \ (co2 \ \circ \ co2') \ (co3 \ \circ \ co3') \ (co4 \ \circ \ co4') \ (co5 \ \circ \ co5') \ (co6 \ \circ \ co6')$
 $(\text{contra1}' \ \circ \ \text{contra1}) \ (\text{contra2}' \ \circ \ \text{contra2}) \ (\text{contra3}' \ \circ \ \text{contra3}) \ (\text{contra4}' \ \circ \ \text{contra4})$
 $\langle \text{proof} \rangle$

lemma *map-FGco-parametric*:
 $\text{rel-fun } (\text{rel-fun } L1 \ L1') \ (\text{rel-fun } (\text{rel-fun } Co1 \ Co1') \ (\text{rel-fun } (\text{rel-fun } Co2 \ Co2') \ (\text{rel-fun } (\text{rel-fun } Co3 \ Co3') \ (\text{rel-fun } (\text{rel-fun } Co4 \ Co4') \ (\text{rel-fun } (\text{rel-fun } Co5 \ Co5') \ (\text{rel-fun } (\text{rel-fun } Co6 \ Co6') \ (\text{rel-fun } (\text{rel-fun } Contra1' \ Contra1) \ (\text{rel-fun } (\text{rel-fun } Contra2' \ Contra2) \ (\text{rel-fun } (\text{rel-fun } Contra3' \ Contra3) \ (\text{rel-fun } (\text{rel-fun } Contra4' \ Contra4) \ (\text{rel-fun } (\text{rel-FGco } L1 \ Co1 \ Co2 \ Co3 \ Co4 \ Co5 \ Co6 \ Contra1 \ Contra2 \ Contra3 \ Contra4) \ (\text{rel-FGco } L1' \ Co1' \ Co2' \ Co3' \ Co4' \ Co5' \ Co6' \ Contra1' \ Contra2' \ Contra3' \ Contra4'))))))))))))$
 $\text{map-FGco } \text{map-FGco}$
 $\langle \text{proof} \rangle$

definition *rel-FGco-pos-distr-cond* :: $(co1 \Rightarrow co1' \Rightarrow \text{bool}) \Rightarrow (co1' \Rightarrow co1'' \Rightarrow \text{bool}) \Rightarrow$
 $(co2 \Rightarrow co2' \Rightarrow \text{bool}) \Rightarrow (co2' \Rightarrow co2'' \Rightarrow \text{bool}) \Rightarrow$
 $(co3 \Rightarrow co3' \Rightarrow \text{bool}) \Rightarrow (co3' \Rightarrow co3'' \Rightarrow \text{bool}) \Rightarrow$
 $(co4 \Rightarrow co4' \Rightarrow \text{bool}) \Rightarrow (co4' \Rightarrow co4'' \Rightarrow \text{bool}) \Rightarrow$
 $(co5 \Rightarrow co5' \Rightarrow \text{bool}) \Rightarrow (co5' \Rightarrow co5'' \Rightarrow \text{bool}) \Rightarrow$
 $(co6 \Rightarrow co6' \Rightarrow \text{bool}) \Rightarrow (co6' \Rightarrow co6'' \Rightarrow \text{bool}) \Rightarrow$
 $(contra1 \Rightarrow contra1' \Rightarrow \text{bool}) \Rightarrow (contra1' \Rightarrow contra1'' \Rightarrow \text{bool}) \Rightarrow$
 $(contra2 \Rightarrow contra2' \Rightarrow \text{bool}) \Rightarrow (contra2' \Rightarrow contra2'' \Rightarrow \text{bool}) \Rightarrow$
 $(contra3 \Rightarrow contra3' \Rightarrow \text{bool}) \Rightarrow (contra3' \Rightarrow contra3'' \Rightarrow \text{bool}) \Rightarrow$
 $(contra4 \Rightarrow contra4' \Rightarrow \text{bool}) \Rightarrow (contra4' \Rightarrow contra4'' \Rightarrow \text{bool}) \Rightarrow$
 $(l1 \times l1' \times l1'' \times f1 \times f2) \text{ itself} \Rightarrow \text{bool}$ **where**
 $\text{rel-FGco-pos-distr-cond } Co1 \ Co1' \ Co2 \ Co2' \ Co3 \ Co3' \ Co4 \ Co4' \ Co5 \ Co5' \ Co6 \ Co6'$
 $Contra1 \ Contra1' \ Contra2 \ Contra2' \ Contra3 \ Contra3' \ Contra4 \ Contra4' \ \text{-} \longleftrightarrow$
 $(\forall (L1 :: l1 \Rightarrow l1' \Rightarrow \text{bool}) (L1' :: l1' \Rightarrow l1'' \Rightarrow \text{bool}).$
 $(\text{rel-FGco } L1 \ Co1 \ Co2 \ Co3 \ Co4 \ Co5 \ Co6 \ Contra1 \ Contra2 \ Contra3 \ Contra4 ::$
 $(-, -, -, -, -, -, -, -, -, f1, f2) \text{ FGco} \Rightarrow -) \text{ OO}$
 $\text{rel-FGco } L1' \ Co1' \ Co2' \ Co3' \ Co4' \ Co5' \ Co6' \ Contra1' \ Contra2' \ Contra3' \ Contra4' \leq$
 $\text{rel-FGco } (L1 \ \text{OO} \ L1') \ (Co1 \ \text{OO} \ Co1') \ (Co2 \ \text{OO} \ Co2') \ (Co3 \ \text{OO} \ Co3') \ (Co4 \ \text{OO} \ Co4') \ (Co5 \ \text{OO} \ Co5') \ (Co6 \ \text{OO} \ Co6')$
 $(Contra1 \ \text{OO} \ Contra1') \ (Contra2 \ \text{OO} \ Contra2') \ (Contra3 \ \text{OO} \ Contra3') \ (Contra4 \ \text{OO} \ Contra4'))$

definition *rel-FGco-neg-distr-cond* :: $(co1 \Rightarrow co1' \Rightarrow \text{bool}) \Rightarrow (co1' \Rightarrow co1'' \Rightarrow$

bool) \Rightarrow
 ('co2 \Rightarrow 'co2' \Rightarrow bool) \Rightarrow ('co2' \Rightarrow 'co2'' \Rightarrow bool) \Rightarrow
 ('co3 \Rightarrow 'co3' \Rightarrow bool) \Rightarrow ('co3' \Rightarrow 'co3'' \Rightarrow bool) \Rightarrow
 ('co4 \Rightarrow 'co4' \Rightarrow bool) \Rightarrow ('co4' \Rightarrow 'co4'' \Rightarrow bool) \Rightarrow
 ('co5 \Rightarrow 'co5' \Rightarrow bool) \Rightarrow ('co5' \Rightarrow 'co5'' \Rightarrow bool) \Rightarrow
 ('co6 \Rightarrow 'co6' \Rightarrow bool) \Rightarrow ('co6' \Rightarrow 'co6'' \Rightarrow bool) \Rightarrow
 ('contra1 \Rightarrow 'contra1' \Rightarrow bool) \Rightarrow ('contra1' \Rightarrow 'contra1'' \Rightarrow bool) \Rightarrow
 ('contra2 \Rightarrow 'contra2' \Rightarrow bool) \Rightarrow ('contra2' \Rightarrow 'contra2'' \Rightarrow bool) \Rightarrow
 ('contra3 \Rightarrow 'contra3' \Rightarrow bool) \Rightarrow ('contra3' \Rightarrow 'contra3'' \Rightarrow bool) \Rightarrow
 ('contra4 \Rightarrow 'contra4' \Rightarrow bool) \Rightarrow ('contra4' \Rightarrow 'contra4'' \Rightarrow bool) \Rightarrow
 ('l1 \times 'l1' \times 'l1'' \times 'f1 \times 'f2) *itself* \Rightarrow bool **where**
rel-FGco-neg-distr-cond Co1 Co1' Co2 Co2' Co3 Co3' Co4 Co4' Co5 Co5' Co6
 Co6'
 Contra1 Contra1' Contra2 Contra2' Contra3 Contra3' Contra4 Contra4' - \longleftrightarrow
 (\forall (L1 :: 'l1 \Rightarrow 'l1' \Rightarrow bool) (L1' :: 'l1' \Rightarrow 'l1'' \Rightarrow bool).
rel-FGco (L1 OO L1') (Co1 OO Co1') (Co2 OO Co2') (Co3 OO Co3')
 (Co4 OO Co4') (Co5 OO Co5') (Co6 OO Co6')
 (Contra1 OO Contra1') (Contra2 OO Contra2') (Contra3 OO Contra3')
 (Contra4 OO Contra4') \leq
 (*rel-FGco* L1 Co1 Co2 Co3 Co4 Co5 Co6 Contra1 Contra2 Contra3 Contra4 ::
 (-, -, -, -, -, -, -, -, -, 'f1, 'f2) *FGco* \Rightarrow -) OO
rel-FGco L1' Co1' Co2' Co3' Co4' Co5' Co6' Contra1' Contra2' Contra3'
 Contra4')

Sufficient conditions for subdistributivity over relation composition.

lemma *rel-FGco-pos-distr-imp*:

fixes Co1 :: 'co1 \Rightarrow 'co1' \Rightarrow bool **and** Co1' :: 'co1' \Rightarrow 'co1'' \Rightarrow bool
and Co2 :: 'co2 \Rightarrow 'co2' \Rightarrow bool **and** Co2' :: 'co2' \Rightarrow 'co2'' \Rightarrow bool
and Co5 :: 'co5 \Rightarrow 'co5' \Rightarrow bool **and** Co5' :: 'co5' \Rightarrow 'co5'' \Rightarrow bool
and *tytok-F* :: ('l1 \times 'l1' \times 'l1'' \times 'co1 \times 'co1' \times 'co1'' \times 'co5 \times 'co5' \times
 'co5'' \times
 'f2) *itself*
and *tytok-G* :: ('co1 \times 'co1' \times 'co1'' \times 'co2 \times 'co2' \times 'co2'' \times 'f1) *itself*
and *tytok-FGco* :: ('l1 \times 'l1' \times 'l1'' \times 'f1 \times 'f2) *itself*
assumes *rel-F-pos-distr-cond*
 (*rel-G* Co1 Co2 Co3 Co4 Contra1 Contra2 :: (-, -, -, -, -, -, 'f1) *G* \Rightarrow -)
 (*rel-G* Co1' Co2' Co3' Co4' Contra1' Contra2') Co3 Co3' Co6 Co6'
 Contra1 Contra1' Contra3 Contra3' Contra4 Contra4' *tytok-F*
and *rel-G-pos-distr-cond* Co3 Co3' Co4 Co4' Contra1 Contra1' Contra2 Con-
 tra2' *tytok-G*
shows *rel-FGco-pos-distr-cond* Co1 Co1' Co2 Co2' Co3 Co3' Co4 Co4' Co5
 Co5' Co6 Co6'
 Contra1 Contra1' Contra2 Contra2' Contra3 Contra3' Contra4 Contra4' *ty-*
tok-FGco
 ⟨*proof*⟩

lemma *rel-FGco-neg-distr-imp*:

fixes Co1 :: 'co1 \Rightarrow 'co1' \Rightarrow bool **and** Co1' :: 'co1' \Rightarrow 'co1'' \Rightarrow bool
and Co2 :: 'co2 \Rightarrow 'co2' \Rightarrow bool **and** Co2' :: 'co2' \Rightarrow 'co2'' \Rightarrow bool

and $Co5 :: 'co5 \Rightarrow 'co5' \Rightarrow bool$ **and** $Co5' :: 'co5' \Rightarrow 'co5'' \Rightarrow bool$
and $tytok-F :: ('l1 \times 'l1' \times 'l1'' \times 'co1 \times 'co1' \times 'co1'' \times 'co5 \times 'co5' \times 'co5'' \times 'f2)$ *itself*
and $tytok-G :: ('co1 \times 'co1' \times 'co1'' \times 'co2 \times 'co2' \times 'co2'' \times 'f1)$ *itself*
and $tytok-FGco :: ('l1 \times 'l1' \times 'l1'' \times 'f1 \times 'f2)$ *itself*
assumes *rel-F-neg-distr-cond*
 $(rel-G\ Co1\ Co2\ Co3\ Co4\ Contra1\ Contra2 :: (-, -, -, -, -, 'f1)\ G \Rightarrow -)$
 $(rel-G\ Co1'\ Co2'\ Co3'\ Co4'\ Contra1'\ Contra2')\ Co3\ Co3'\ Co6\ Co6'$
 $Contra1\ Contra1'\ Contra3\ Contra3'\ Contra4\ Contra4'\ tytok-F$
and *rel-G-neg-distr-cond* $Co3\ Co3'\ Co4\ Co4'\ Contra1\ Contra1'\ Contra2\ Contra2'\ tytok-G$
shows *rel-FGco-neg-distr-cond* $Co1\ Co1'\ Co2\ Co2'\ Co3\ Co3'\ Co4\ Co4'\ Co5\ Co5'\ Co6\ Co6'$
 $Contra1\ Contra1'\ Contra2\ Contra2'\ Contra3\ Contra3'\ Contra4\ Contra4'\ tytok-FGco$
<proof>

lemma *rel-FGco-pos-distr-cond-eq*:
fixes $tytok :: ('l1 \times 'l1' \times 'l1'' \times 'f1 \times 'f2)$ *itself*
shows *rel-FGco-pos-distr-cond* $(=) (=) (=) (=) (=) (=) (=) (=) (=) (=)$
 $(=)$
 $(=) (=) (=) (=) (=) (=) (=) (=) (=) tytok$
<proof>

lemma *rel-FGco-neg-distr-cond-eq*:
fixes $tytok :: ('l1 \times 'l1' \times 'l1'' \times 'f1 \times 'f2)$ *itself*
shows *rel-FGco-neg-distr-cond* $(=) (=) (=) (=) (=) (=) (=) (=) (=) (=)$
 $(=)$
 $(=) (=) (=) (=) (=) (=) (=) (=) (=) tytok$
<proof>

definition *rel-FGco* $L1 = rel-FGco\ L1 (=) (=) (=) (=) (=) (=) (=) (=) (=)$
 $(=)$

definition *mapl-FGco* $l1 = map-FGco\ l1\ id\ id\ id\ id\ id\ id\ id\ id\ id\ id$

type-synonym $('co1, 'co2, 'co3, 'co4, 'co5, 'co6,$
 $'contra1, 'contra2, 'contra3, 'contra4, 'f1, 'f2)\ FGcobd =$
 $((('co1, 'co2, 'co3, 'co4, 'contra1, 'contra2, 'f1)\ G,$
 $'co3, 'co6, 'contra1, 'contra3, 'contra4, 'f2)\ Fbd$

definition *set1-FGco* $:: ('l1, 'co1, 'co2, 'co3, 'co4, 'co5, 'co6,$
 $'contra1, 'contra2, 'contra3, 'contra4, 'f1, 'f2)\ FGco \Rightarrow 'l1\ set$ **where**
 $set1-FGco\ x = set1-F\ x$

definition *bd-FGco* $:: ('co1, 'co2, 'co3, 'co4, 'co5, 'co6,$
 $'contra1, 'contra2, 'contra3, 'contra4, 'f1, 'f2)\ FGcobd\ rel$ **where**
 $bd-FGco = bd-F$

lemma *set1-FGco-map*: $set1-FGco \circ mapl-FGco\ l1 = image\ l1 \circ set1-FGco$

<proof>

lemma *bd-FGco-card-order: card-order bd-FGco*

<proof>

lemma *bd-FGco-cinfinite: cinfinite bd-FGco*

<proof>

lemma *set1-FGco-bound:*

fixes $x :: (-, 'co1, 'co2, 'co3, 'co4, 'co5, 'co6,$
 $'contra1, 'contra2, 'contra3, 'contra4, 'f1, 'f2) FGco$

shows $card-of (set1-FGco x) \leq_o (bd-FGco :: ('co1, 'co2, 'co3, 'co4, 'co5, 'co6,$
 $'contra1, 'contra2, 'contra3, 'contra4, 'f1, 'f2) FGcobd rel)$

<proof>

lemma *mapl-FGco-cong:*

assumes $\bigwedge z. z \in set1-FGco x \implies l1 z = l1' z$

shows $mapl-FGco l1 x = mapl-FGco l1' x$

<proof>

lemma *rell-FGco-mono-strong:*

assumes $rell-FGco L1 x y$

and $\bigwedge a b. a \in set1-FGco x \implies b \in set1-FGco y \implies L1 a b \implies L1' a b$

shows $rell-FGco L1' x y$

<proof>

3.3 Composition in a contravariant position

type-synonym

$(l1, 'co1, 'co2, 'co3, 'co4, 'co5, 'contra1,$
 $'contra2, 'contra3, 'contra4, 'contra5, 'f1, 'f2) FGcontra =$
 $(l1, 'co1, 'co3, 'co1, 'co4, 'co5, ('contra1, 'contra2, 'contra3, 'contra4, 'co1,$
 $'co2, 'f1) G,$
 $'contra1, 'contra5, 'f2) F$

The type variables *'co1* and *'contra1* have each been merged.

definition $rel-FGcontra L1 Co1 Co2 Co3 Co4 Co5 Contra1 Contra2 Contra3 Contra4$
 $Contra5 =$

$rel-F L1 Co1 Co3 Co1 Co4 Co5 (rel-G Contra1 Contra2 Contra3 Contra4 Co1$
 $Co2) Contra1 Contra5$

definition $map-FGcontra l1 co1 co2 co3 co4 co5 contra1 contra2 contra3 contra4$
 $contra5 =$

$map-F l1 co1 co3 co1 co4 co5 (map-G contra1 contra2 contra3 contra4 co1 co2)$
 $contra1 contra5$

lemma *rel-FGcontra-mono:*

$\llbracket L1 \leq L1'; Co1 \leq Co1'; Co2 \leq Co2'; Co3 \leq Co3'; Co4 \leq Co4'; Co5 \leq Co5';$
 $Contra1' \leq Contra1; Contra2' \leq Contra2; Contra3' \leq Contra3;$

$Contra4' \leq Contra4; Contra5' \leq Contra5 \parallel \implies$
 $rel-FGcontra L1 Co1 Co2 Co3 Co4 Co5 Contra1 Contra2 Contra3 Contra4 Contra5 \leq$
 $rel-FGcontra L1' Co1' Co2' Co3' Co4' Co5' Contra1' Contra2' Contra3' Contra4' Contra5'$
 ⟨proof⟩

lemma *rel-FGcontra-eq*: $rel-FGcontra (=) (=) (=) (=) (=) (=) (=) (=) (=) (=)$
 $(=) = (=)$
 ⟨proof⟩

lemma *rel-FGcontra-conversep*:
 $rel-FGcontra L1^{-1-1} Co1^{-1-1} Co2^{-1-1} Co3^{-1-1} Co4^{-1-1} Co5^{-1-1} Contra1^{-1-1} Contra2^{-1-1} Contra3^{-1-1} Contra4^{-1-1} Contra5^{-1-1} =$
 $(rel-FGcontra L1 Co1 Co2 Co3 Co4 Co5 Contra1 Contra2 Contra3 Contra4 Contra5)^{-1-1}$
 ⟨proof⟩

lemma *map-FGcontra-id0*: $map-FGcontra id id id id id id id id id id id = id$
 ⟨proof⟩

lemma *map-FGcontra-comp*:
 $map-FGcontra l1 co1 co2 co3 co4 co5 contra1 contra2 contra3 contra4 contra5 \circ$
 $map-FGcontra l1' co1' co2' co3' co4' co5' contra1' contra2' contra3' contra4' contra5' =$
 $map-FGcontra (l1 \circ l1') (co1 \circ co1') (co2 \circ co2') (co3 \circ co3') (co4 \circ co4') (co5 \circ co5')$
 $(contra1' \circ contra1) (contra2' \circ contra2) (contra3' \circ contra3)$
 $(contra4' \circ contra4) (contra5' \circ contra5)$
 ⟨proof⟩

lemma *map-FGcontra-parametric*:
 $rel-fun (rel-fun L1 L1') (rel-fun (rel-fun Co1 Co1') (rel-fun (rel-fun Co2 Co2')$
 $(rel-fun (rel-fun Co3 Co3') (rel-fun (rel-fun Co4 Co4') (rel-fun (rel-fun Co5 Co5')$
 $(rel-fun (rel-fun Contra1' Contra1) (rel-fun (rel-fun Contra2' Contra2)$
 $(rel-fun (rel-fun Contra3' Contra3) (rel-fun (rel-fun Contra4' Contra4)$
 $(rel-fun (rel-fun Contra5' Contra5)$
 $(rel-fun (rel-FGcontra L1 Co1 Co2 Co3 Co4 Co5 Contra1 Contra2 Contra3 Contra4 Contra5)$
 $(rel-FGcontra L1' Co1' Co2' Co3' Co4' Co5' Contra1' Contra2' Contra3' Contra4' Contra5')))))))))))$
 $map-FGcontra map-FGcontra$
 ⟨proof⟩

definition *rel-FGcontra-pos-distr-cond* :: $('co1 \Rightarrow 'co1' \Rightarrow bool) \Rightarrow ('co1' \Rightarrow 'co1'' \Rightarrow bool) \Rightarrow$
 $('co2 \Rightarrow 'co2' \Rightarrow bool) \Rightarrow ('co2' \Rightarrow 'co2'' \Rightarrow bool) \Rightarrow$
 $('co3 \Rightarrow 'co3' \Rightarrow bool) \Rightarrow ('co3' \Rightarrow 'co3'' \Rightarrow bool) \Rightarrow$

$(\text{'co4} \Rightarrow \text{'co4}' \Rightarrow \text{bool}) \Rightarrow (\text{'co4}' \Rightarrow \text{'co4}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'co5} \Rightarrow \text{'co5}' \Rightarrow \text{bool}) \Rightarrow (\text{'co5}' \Rightarrow \text{'co5}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'contra1} \Rightarrow \text{'contra1}' \Rightarrow \text{bool}) \Rightarrow (\text{'contra1}' \Rightarrow \text{'contra1}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'contra2} \Rightarrow \text{'contra2}' \Rightarrow \text{bool}) \Rightarrow (\text{'contra2}' \Rightarrow \text{'contra2}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'contra3} \Rightarrow \text{'contra3}' \Rightarrow \text{bool}) \Rightarrow (\text{'contra3}' \Rightarrow \text{'contra3}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'contra4} \Rightarrow \text{'contra4}' \Rightarrow \text{bool}) \Rightarrow (\text{'contra4}' \Rightarrow \text{'contra4}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'contra5} \Rightarrow \text{'contra5}' \Rightarrow \text{bool}) \Rightarrow (\text{'contra5}' \Rightarrow \text{'contra5}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'l1} \times \text{'l1}' \times \text{'l1}'' \times \text{'f1} \times \text{'f2}) \text{ itself} \Rightarrow \text{bool}$ **where**
 $\text{rel-FGcontra-pos-distr-cond } Co1\ Co1'\ Co2\ Co2'\ Co3\ Co3'\ Co4\ Co4'\ Co5\ Co5'$
 $\text{Contra1}\ \text{Contra1}'\ \text{Contra2}\ \text{Contra2}'\ \text{Contra3}\ \text{Contra3}'\ \text{Contra4}\ \text{Contra4}'\ \text{Contra5}\ \text{Contra5}' - \longleftrightarrow$
 $(\forall (L1 :: \text{'l1} \Rightarrow \text{'l1}' \Rightarrow \text{bool}) (L1' :: \text{'l1}' \Rightarrow \text{'l1}'' \Rightarrow \text{bool}).$
 $\text{rel-FGcontra } L1\ Co1\ Co2\ Co3\ Co4\ Co5\ \text{Contra1}\ \text{Contra2}\ \text{Contra3}\ \text{Contra4}$
 $\text{Contra5} ::$
 $(-, -, -, -, -, -, -, -, -, \text{'f1}, \text{'f2}) \text{ FGcontra} \Rightarrow -) \text{ OO}$
 $\text{rel-FGcontra } L1'\ Co1'\ Co2'\ Co3'\ Co4'\ Co5'\ \text{Contra1}'\ \text{Contra2}'\ \text{Contra3}'$
 $\text{Contra4}'\ \text{Contra5}' \leq$
 $\text{rel-FGcontra } (L1\ \text{OO } L1') (Co1\ \text{OO } Co1') (Co2\ \text{OO } Co2') (Co3\ \text{OO } Co3')$
 $(Co4\ \text{OO } Co4') (Co5\ \text{OO } Co5')$
 $(\text{Contra1}\ \text{OO } \text{Contra1}') (\text{Contra2}\ \text{OO } \text{Contra2}') (\text{Contra3}\ \text{OO } \text{Contra3}')$
 $(\text{Contra4}\ \text{OO } \text{Contra4}') (\text{Contra5}\ \text{OO } \text{Contra5}')$

definition $\text{rel-FGcontra-neg-distr-cond} :: (\text{'co1} \Rightarrow \text{'co1}' \Rightarrow \text{bool}) \Rightarrow (\text{'co1}' \Rightarrow \text{'co1}'' \Rightarrow \text{bool}) \Rightarrow$
 $\text{bool} \Rightarrow$
 $(\text{'co2} \Rightarrow \text{'co2}' \Rightarrow \text{bool}) \Rightarrow (\text{'co2}' \Rightarrow \text{'co2}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'co3} \Rightarrow \text{'co3}' \Rightarrow \text{bool}) \Rightarrow (\text{'co3}' \Rightarrow \text{'co3}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'co4} \Rightarrow \text{'co4}' \Rightarrow \text{bool}) \Rightarrow (\text{'co4}' \Rightarrow \text{'co4}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'co5} \Rightarrow \text{'co5}' \Rightarrow \text{bool}) \Rightarrow (\text{'co5}' \Rightarrow \text{'co5}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'contra1} \Rightarrow \text{'contra1}' \Rightarrow \text{bool}) \Rightarrow (\text{'contra1}' \Rightarrow \text{'contra1}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'contra2} \Rightarrow \text{'contra2}' \Rightarrow \text{bool}) \Rightarrow (\text{'contra2}' \Rightarrow \text{'contra2}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'contra3} \Rightarrow \text{'contra3}' \Rightarrow \text{bool}) \Rightarrow (\text{'contra3}' \Rightarrow \text{'contra3}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'contra4} \Rightarrow \text{'contra4}' \Rightarrow \text{bool}) \Rightarrow (\text{'contra4}' \Rightarrow \text{'contra4}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'contra5} \Rightarrow \text{'contra5}' \Rightarrow \text{bool}) \Rightarrow (\text{'contra5}' \Rightarrow \text{'contra5}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'l1} \times \text{'l1}' \times \text{'l1}'' \times \text{'f1} \times \text{'f2}) \text{ itself} \Rightarrow \text{bool}$ **where**
 $\text{rel-FGcontra-neg-distr-cond } Co1\ Co1'\ Co2\ Co2'\ Co3\ Co3'\ Co4\ Co4'\ Co5\ Co5'$
 $\text{Contra1}\ \text{Contra1}'\ \text{Contra2}\ \text{Contra2}'\ \text{Contra3}\ \text{Contra3}'\ \text{Contra4}\ \text{Contra4}'\ \text{Contra5}\ \text{Contra5}' - \longleftrightarrow$
 $(\forall (L1 :: \text{'l1} \Rightarrow \text{'l1}' \Rightarrow \text{bool}) (L1' :: \text{'l1}' \Rightarrow \text{'l1}'' \Rightarrow \text{bool}).$
 $\text{rel-FGcontra } (L1\ \text{OO } L1') (Co1\ \text{OO } Co1') (Co2\ \text{OO } Co2') (Co3\ \text{OO } Co3')$
 $(Co4\ \text{OO } Co4') (Co5\ \text{OO } Co5')$
 $(\text{Contra1}\ \text{OO } \text{Contra1}') (\text{Contra2}\ \text{OO } \text{Contra2}') (\text{Contra3}\ \text{OO } \text{Contra3}')$
 $(\text{Contra4}\ \text{OO } \text{Contra4}') (\text{Contra5}\ \text{OO } \text{Contra5}')$ \leq
 $\text{rel-FGcontra } L1\ Co1\ Co2\ Co3\ Co4\ Co5\ \text{Contra1}\ \text{Contra2}\ \text{Contra3}\ \text{Contra4}$
 $\text{Contra5} ::$
 $(-, -, -, -, -, -, -, -, -, \text{'f1}, \text{'f2}) \text{ FGcontra} \Rightarrow -) \text{ OO}$
 $\text{rel-FGcontra } L1'\ Co1'\ Co2'\ Co3'\ Co4'\ Co5'\ \text{Contra1}'\ \text{Contra2}'\ \text{Contra3}'$
 $\text{Contra4}'\ \text{Contra5}'$

Sufficient conditions for subdistributivity over relation composition.

lemma *rel-FGcontra-pos-distr-imp*:

fixes $Co1 :: 'co1 \Rightarrow 'co1' \Rightarrow bool$ **and** $Co1' :: 'co1' \Rightarrow 'co1'' \Rightarrow bool$
and $Co3 :: 'co3 \Rightarrow 'co3' \Rightarrow bool$ **and** $Co3' :: 'co3' \Rightarrow 'co3'' \Rightarrow bool$
and $Contra1 :: 'contra1 \Rightarrow 'contra1' \Rightarrow bool$ **and** $Contra1' :: 'contra1' \Rightarrow 'contra1'' \Rightarrow bool$
and $Contra2 :: 'contra2 \Rightarrow 'contra2' \Rightarrow bool$ **and** $Contra2' :: 'contra2' \Rightarrow 'contra2'' \Rightarrow bool$
and $tytok-F :: ('l1 \times 'l1' \times 'l1'' \times 'co1 \times 'co1' \times 'co1'' \times 'co3 \times 'co3' \times 'co3'' \times 'f2) \text{ itself}$
and $tytok-G :: ('contra1 \times 'contra1' \times 'contra1'' \times 'contra2 \times 'contra2' \times 'contra2'' \times 'f1) \text{ itself}$
and $tytok-FGcontra :: ('l1 \times 'l1' \times 'l1'' \times 'f1 \times 'f2) \text{ itself}$
assumes *rel-F-pos-distr-cond* $Co1 Co1' Co4 Co4' Co5 Co5'$
 $(rel-G Contra1 Contra2 Contra3 Contra4 Co1 Co2 :: (-, -, -, -, -, -, 'f1) G \Rightarrow -)$
 $(rel-G Contra1' Contra2' Contra3' Contra4' Co1' Co2')$
 $Contra1 Contra1' Contra5 Contra5' tytok-F$
and *rel-G-neg-distr-cond* $Contra3 Contra3' Contra4 Contra4' Co1 Co1' Co2 Co2' tytok-G$
shows *rel-FGcontra-pos-distr-cond* $Co1 Co1' Co2 Co2' Co3 Co3' Co4 Co4' Co5 Co5'$
 $Contra1 Contra1' Contra2 Contra2' Contra3 Contra3' Contra4 Contra4' Contra5 Contra5'$
 $tytok-FGcontra$
 $\langle proof \rangle$

lemma *rel-FGcontra-neg-distr-imp*:

fixes $Co1 :: 'co1 \Rightarrow 'co1' \Rightarrow bool$ **and** $Co1' :: 'co1' \Rightarrow 'co1'' \Rightarrow bool$
and $Co3 :: 'co3 \Rightarrow 'co3' \Rightarrow bool$ **and** $Co3' :: 'co3' \Rightarrow 'co3'' \Rightarrow bool$
and $Contra1 :: 'contra1 \Rightarrow 'contra1' \Rightarrow bool$ **and** $Contra1' :: 'contra1' \Rightarrow 'contra1'' \Rightarrow bool$
and $Contra2 :: 'contra2 \Rightarrow 'contra2' \Rightarrow bool$ **and** $Contra2' :: 'contra2' \Rightarrow 'contra2'' \Rightarrow bool$
and $tytok-F :: ('l1 \times 'l1' \times 'l1'' \times 'co1 \times 'co1' \times 'co1'' \times 'co3 \times 'co3' \times 'co3'' \times 'f2) \text{ itself}$
and $tytok-G :: ('contra1 \times 'contra1' \times 'contra1'' \times 'contra2 \times 'contra2' \times 'contra2'' \times 'f1) \text{ itself}$
and $tytok-FGcontra :: ('l1 \times 'l1' \times 'l1'' \times 'f1 \times 'f2) \text{ itself}$
assumes *rel-F-neg-distr-cond* $Co1 Co1' Co4 Co4' Co5 Co5'$
 $(rel-G Contra1 Contra2 Contra3 Contra4 Co1 Co2 :: (-, -, -, -, -, -, 'f1) G \Rightarrow -)$
 $(rel-G Contra1' Contra2' Contra3' Contra4' Co1' Co2')$
 $Contra1 Contra1' Contra5 Contra5' tytok-F$
and *rel-G-pos-distr-cond* $Contra3 Contra3' Contra4 Contra4' Co1 Co1' Co2 Co2' tytok-G$

shows *rel-FGcontra-neg-distr-cond* *Co1 Co1' Co2 Co2' Co3 Co3' Co4 Co4' Co5 Co5'*

Contra1 Contra1' Contra2 Contra2' Contra3 Contra3' Contra4 Contra4' Contra5 Contra5' *tytok-FGcontra*

<proof>

lemma *rel-FGcontra-pos-distr-cond-eq*:

fixes *tytok* :: (*'l1* × *'l1'* × *'l1''* × *'f1* × *'f2*) *itself*

shows *rel-FGcontra-pos-distr-cond* (=) (=) (=) (=) (=) (=) (=) (=) (=) (=) (=) (=) (=) (=) *tytok*

<proof>

lemma *rel-FGcontra-neg-distr-cond-eq*:

fixes *tytok* :: (*'l1* × *'l1'* × *'l1''* × *'f1* × *'f2*) *itself*

shows *rel-FGcontra-neg-distr-cond* (=) (=) (=) (=) (=) (=) (=) (=) (=) (=) (=) (=) (=) (=) *tytok*

<proof>

definition *rell-FGcontra* *L1* = *rel-FGcontra* *L1* (=) (=) (=) (=) (=) (=) (=) (=) (=) (=)

definition *mapl-FGcontra* *l1* = *map-FGcontra* *l1 id id id id id id id id id id*

type-synonym (*'co1, 'co2, 'co3, 'co4, 'co5, 'contra1, 'contra2, 'contra3, 'contra4, 'contra5,*

'f1, 'f2) *FGcontrabd* =

(*'co1, 'co4, 'co5, 'contra1, 'contra2, 'contra3, 'contra4, 'co1, 'co2, 'f1*) *G,*
'contra1, 'contra5, 'f2) *Fbd*

definition *set1-FGcontra* :: (*'l1, 'co1, 'co2, 'co3, 'co4, 'co5,*

'contra1, 'contra2, 'contra3, 'contra4, 'contra5, 'f1, 'f2) *FGcontra* ⇒ *'l1 set*

where

set1-FGcontra *x* = *set1-F* *x*

definition *bd-FGcontra* :: (*'co1, 'co2, 'co3, 'co4, 'co5,*

'contra1, 'contra2, 'contra3, 'contra4, 'contra5, 'f1, 'f2) *FGcontrabd* **rel** **where**
bd-FGcontra = *bd-F*

lemma *set1-FGcontra-map*: *set1-FGcontra* ◦ *mapl-FGcontra* *l1* = *image* *l1* ◦ *set1-FGcontra*

<proof>

lemma *bd-FGcontra-card-order*: *card-order* *bd-FGcontra*

<proof>

lemma *bd-FGcontra-cinfinite*: *cinfinite* *bd-FGcontra*

<proof>

lemma *set1-FGcontra-bound*:

fixes *x* :: (*-, 'co1, 'co2, 'co3, 'co4, 'co5,*

'contra1, 'contra2, 'contra3, 'contra4, 'contra5, 'f1, 'f2) *FGcontra*

shows $\text{card-of } (\text{set1-FGcontra } x) \leq o (\text{bd-FGcontra } :: ('co1, 'co2, 'co3, 'co4, 'co5, 'contra1, 'contra2, 'contra3, 'contra4, 'contra5, 'f1, 'f2) \text{FGcontrabd rel})$
 ⟨proof⟩

lemma *mapl-FGcontra-contrang*:

assumes $\bigwedge z. z \in \text{set1-FGcontra } x \implies l1 \ z = l1' \ z$

shows $\text{mapl-FGcontra } l1 \ x = \text{mapl-FGcontra } l1' \ x$

⟨proof⟩

lemma *rell-FGcontra-mono-strong*:

assumes $\text{rell-FGcontra } L1 \ x \ y$

and $\bigwedge a \ b. a \in \text{set1-FGcontra } x \implies b \in \text{set1-FGcontra } y \implies L1 \ a \ b \implies L1' \ a \ b$

shows $\text{rell-FGcontra } L1' \ x \ y$

⟨proof⟩

3.4 Composition in a fixed position

type-synonym $('l1, 'l2, 'co1, 'co2, 'contra1, 'contra2, 'f1, 'f2, 'f3, 'f4, 'f5, 'f6, 'f7) \text{FGf} =$

$('l1, 'l2, 'f2, 'co1, 'co2, 'f4, 'contra1, 'contra2, 'f6, ('f1, 'f2, 'f3, 'f4, 'f5, 'f6, 'f7) \text{G}) \text{F}$

The type variables $'f2, 'f4$ and $'f6$ have each been merged.

definition $\text{rel-FGf } L1 \ L2 \ Co1 \ Co2 \ Contra1 \ Contra2 =$

$\text{rel-F } L1 \ L2 \ (=) \ Co1 \ Co2 \ (=) \ Contra1 \ Contra2 \ (=)$

definition $\text{map-FGf } l1 \ l2 \ co1 \ co2 \ contra1 \ contra2 = \text{map-F } l1 \ l2 \ \text{id} \ co1 \ co2 \ \text{id} \ \text{contra1} \ \text{contra2} \ \text{id}$

lemma *rel-FGf-mono*:

$\llbracket L1 \leq L1'; L2 \leq L2'; Co1 \leq Co1'; Co2 \leq Co2'; Contra1' \leq Contra1; Contra2' \leq Contra2 \rrbracket \implies$

$\text{rel-FGf } L1 \ L2 \ Co1 \ Co2 \ Contra1 \ Contra2 \leq \text{rel-FGf } L1' \ L2' \ Co1' \ Co2' \ Contra1' \ Contra2'$

⟨proof⟩

lemma *rel-FGf-eq*: $\text{rel-FGf } (=) \ (=) \ (=) \ (=) \ (=) \ (=) \ (=) \ (=)$

⟨proof⟩

lemma *rel-FGf-conversep*:

$\text{rel-FGf } L1^{-1-1} \ L2^{-1-1} \ Co1^{-1-1} \ Co2^{-1-1} \ Contra1^{-1-1} \ Contra2^{-1-1} = (\text{rel-FGf } L1 \ L2 \ Co1 \ Co2 \ Contra1 \ Contra2)^{-1-1}$

⟨proof⟩

lemma *map-FGf-id0*: $\text{map-FGf } \text{id} \ \text{id} \ \text{id} \ \text{id} \ \text{id} \ \text{id} = \text{id}$

⟨proof⟩

lemma *map-FGf-comp*: $\text{map-FGf } l1 \ l2 \ co1 \ co2 \ contra1 \ contra2 \circ$

$\text{map-FGf } l1' \ l2' \ co1' \ co2' \ \text{contra1}' \ \text{contra2}' =$
 $\text{map-FGf } (l1 \circ l1') \ (l2 \circ l2') \ (co1 \circ co1') \ (co2 \circ co2') \ (\text{contra1}' \circ \text{contra1})$
 $(\text{contra2}' \circ \text{contra2})$
 $\langle \text{proof} \rangle$

lemma *map-FGf-parametric*:

$\text{rel-fun } (\text{rel-fun } L1 \ L1') \ (\text{rel-fun } (\text{rel-fun } L2 \ L2')$
 $(\text{rel-fun } (\text{rel-fun } Co1 \ Co1') \ (\text{rel-fun } (\text{rel-fun } Co2 \ Co2')$
 $(\text{rel-fun } (\text{rel-fun } Contra1' \ Contra1) \ (\text{rel-fun } (\text{rel-fun } Contra2' \ Contra2)$
 $(\text{rel-fun } (\text{rel-FGf } L1 \ L2 \ Co1 \ Co2 \ Contra1 \ Contra2)$
 $(\text{rel-FGf } L1' \ L2' \ Co1' \ Co2' \ Contra1' \ Contra2'))))))))$
 $\text{map-FGf } \text{map-FGf}$
 $\langle \text{proof} \rangle$

definition *rel-FGf-pos-distr-cond* :: $(co1 \Rightarrow co1' \Rightarrow \text{bool}) \Rightarrow (co1' \Rightarrow co1'' \Rightarrow \text{bool}) \Rightarrow$

$\text{bool}) \Rightarrow$
 $(co2 \Rightarrow co2' \Rightarrow \text{bool}) \Rightarrow (co2' \Rightarrow co2'' \Rightarrow \text{bool}) \Rightarrow$
 $(contra1 \Rightarrow contra1' \Rightarrow \text{bool}) \Rightarrow (contra1' \Rightarrow contra1'' \Rightarrow \text{bool}) \Rightarrow$
 $(contra2 \Rightarrow contra2' \Rightarrow \text{bool}) \Rightarrow (contra2' \Rightarrow contra2'' \Rightarrow \text{bool}) \Rightarrow$
 $(l1 \times l1' \times l1'' \times l2 \times l2' \times l2'' \times$
 $f1 \times f2 \times f3 \times f4 \times f5 \times f6 \times f7) \text{ itself} \Rightarrow \text{bool}$ **where**
 $\text{rel-FGf-pos-distr-cond } Co1 \ Co1' \ Co2 \ Co2' \ Contra1 \ Contra1' \ Contra2 \ Contra2'$
 $- \longleftrightarrow$
 $(\forall (L1 :: l1 \Rightarrow l1' \Rightarrow \text{bool}) (L1' :: l1' \Rightarrow l1'' \Rightarrow \text{bool})$
 $(L2 :: l2 \Rightarrow l2' \Rightarrow \text{bool}) (L2' :: l2' \Rightarrow l2'' \Rightarrow \text{bool}).$
 $(\text{rel-FGf } L1 \ L2 \ Co1 \ Co2 \ Contra1 \ Contra2 ::$
 $(-, -, -, -, -, f1, f2, f3, f4, f5, f6, f7) \text{ FGf} \Rightarrow -) \text{ OO}$
 $\text{rel-FGf } L1' \ L2' \ Co1' \ Co2' \ Contra1' \ Contra2' \leq$
 $\text{rel-FGf } (L1 \text{ OO } L1') \ (L2 \text{ OO } L2') \ (Co1 \text{ OO } Co1') \ (Co2 \text{ OO } Co2')$
 $(Contra1 \text{ OO } Contra1') \ (Contra2 \text{ OO } Contra2'))$

definition *rel-FGf-neg-distr-cond* :: $(co1 \Rightarrow co1' \Rightarrow \text{bool}) \Rightarrow (co1' \Rightarrow co1'' \Rightarrow \text{bool}) \Rightarrow$

$\text{bool}) \Rightarrow$
 $(co2 \Rightarrow co2' \Rightarrow \text{bool}) \Rightarrow (co2' \Rightarrow co2'' \Rightarrow \text{bool}) \Rightarrow$
 $(contra1 \Rightarrow contra1' \Rightarrow \text{bool}) \Rightarrow (contra1' \Rightarrow contra1'' \Rightarrow \text{bool}) \Rightarrow$
 $(contra2 \Rightarrow contra2' \Rightarrow \text{bool}) \Rightarrow (contra2' \Rightarrow contra2'' \Rightarrow \text{bool}) \Rightarrow$
 $(l1 \times l1' \times l1'' \times l2 \times l2' \times l2'' \times$
 $f1 \times f2 \times f3 \times f4 \times f5 \times f6 \times f7) \text{ itself} \Rightarrow \text{bool}$ **where**
 $\text{rel-FGf-neg-distr-cond } Co1 \ Co1' \ Co2 \ Co2' \ Contra1 \ Contra1' \ Contra2 \ Contra2'$
 $- \longleftrightarrow$
 $(\forall (L1 :: l1 \Rightarrow l1' \Rightarrow \text{bool}) (L1' :: l1' \Rightarrow l1'' \Rightarrow \text{bool})$
 $(L2 :: l2 \Rightarrow l2' \Rightarrow \text{bool}) (L2' :: l2' \Rightarrow l2'' \Rightarrow \text{bool}).$
 $\text{rel-FGf } (L1 \text{ OO } L1') \ (L2 \text{ OO } L2') \ (Co1 \text{ OO } Co1') \ (Co2 \text{ OO } Co2')$
 $(Contra1 \text{ OO } Contra1') \ (Contra2 \text{ OO } Contra2') \leq$
 $(\text{rel-FGf } L1 \ L2 \ Co1 \ Co2 \ Contra1 \ Contra2 ::$
 $(-, -, -, -, -, f1, f2, f3, f4, f5, f6, f7) \text{ FGf} \Rightarrow -) \text{ OO}$
 $\text{rel-FGf } L1' \ L2' \ Co1' \ Co2' \ Contra1' \ Contra2')$

Sufficient conditions for subdistributivity over relation composition.

lemma *rel-FGf-pos-distr-imp*:

fixes *tytok-F* :: ('l1 × 'l1' × 'l1'' × 'l2 × 'l2' × 'l2'' × 'f2 × 'f2 × 'f2 ×
('f1, 'f2, 'f3, 'f4, 'f5, 'f6, 'f7) G) *itself*
and *tytok-FGf* :: ('l1 × 'l1' × 'l1'' × 'l2 × 'l2' × 'l2'' ×
'f1 × 'f2 × 'f3 × 'f4 × 'f5 × 'f6 × 'f7) *itself*
assumes *rel-F-pos-distr-cond* Co1 Co1' Co2 Co2' ((=) :: 'f4 ⇒ -) ((=) :: 'f4 ⇒
-)
Contra1 *Contra1'* *Contra2* *Contra2'* ((=) :: 'f6 ⇒ -) ((=) :: 'f6 ⇒ -) *tytok-F*
shows *rel-FGf-pos-distr-cond* Co1 Co1' Co2 Co2' *Contra1* *Contra1'* *Contra2*
Contra2' *tytok-FGf*
<proof>

lemma *rel-FGf-neg-distr-imp*:

fixes *tytok-F* :: ('l1 × 'l1' × 'l1'' × 'l2 × 'l2' × 'l2'' × 'f2 × 'f2 × 'f2 ×
('f1, 'f2, 'f3, 'f4, 'f5, 'f6, 'f7) G) *itself*
and *tytok-FGf* :: ('l1 × 'l1' × 'l1'' × 'l2 × 'l2' × 'l2'' ×
'f1 × 'f2 × 'f3 × 'f4 × 'f5 × 'f6 × 'f7) *itself*
assumes *rel-F-neg-distr-cond* Co1 Co1' Co2 Co2' ((=) :: 'f4 ⇒ -) ((=) :: 'f4 ⇒
-)
Contra1 *Contra1'* *Contra2* *Contra2'* ((=) :: 'f6 ⇒ -) ((=) :: 'f6 ⇒ -) *tytok-F*
shows *rel-FGf-neg-distr-cond* Co1 Co1' Co2 Co2' *Contra1* *Contra1'* *Contra2*
Contra2' *tytok-FGf*
<proof>

lemma *rel-FGf-pos-distr-cond-eq*:

fixes *tytok* :: ('l1 × 'l1' × 'l1'' × 'l2 × 'l2' × 'l2'' ×
'f1 × 'f2 × 'f3 × 'f4 × 'f5 × 'f6 × 'f7) *itself*
shows *rel-FGf-pos-distr-cond* (=) (=) (=) (=) (=) (=) (=) (=) *tytok*
<proof>

lemma *rel-FGf-neg-distr-cond-eq*:

fixes *tytok* :: ('l1 × 'l1' × 'l1'' × 'l2 × 'l2' × 'l2'' ×
'f1 × 'f2 × 'f3 × 'f4 × 'f5 × 'f6 × 'f7) *itself*
shows *rel-FGf-neg-distr-cond* (=) (=) (=) (=) (=) (=) (=) (=) *tytok*
<proof>

definition *rell-FGf* L1 L2 = *rel-FGf* L1 L2 (=) (=) (=) (=)

definition *mapl-FGf* l1 l2 = *map-FGf* l1 l2 *id id id id*

type-synonym ('co1, 'co2, 'contra1, 'contra2, 'f1, 'f2, 'f3, 'f4, 'f5, 'f6, 'f7) *FGfbd*
=
('co1, 'co2, 'f4, 'contra1, 'contra2, 'f6, ('f1, 'f2, 'f3, 'f4, 'f5, 'f6, 'f7) G) *Fbd*

definition *set1-FGf* :: ('l1, 'l2, 'co1, 'co2, 'contra1, 'contra2,
'f1, 'f2, 'f3, 'f4, 'f5, 'f6, 'f7) *FGf* ⇒ 'l1 *set* **where**
set1-FGf x = *set1-F* x

definition *set2-FGf* :: ('l1, 'l2, 'co1, 'co2, 'contra1, 'contra2,
'f1, 'f2, 'f3, 'f4, 'f5, 'f6, 'f7) *FGf* ⇒ 'l2 *set* **where**

$set2-FGf\ x = set2-F\ x$

definition $bd-FGf :: ('co1, 'co2, 'contra1, 'contra2, 'f1, 'f2, 'f3, 'f4, 'f5, 'f6, 'f7)$
 $FGfbd\ rel$
where $bd-FGf = bd-F$

lemma $set1-FGf-map: set1-FGf \circ map1-FGf\ l1\ l2 = image\ l1 \circ set1-FGf$
 $\langle proof \rangle$

lemma $bd-FGf-card-order: card-order\ bd-FGf$
 $\langle proof \rangle$

lemma $bd-FGf-cinfinite: cinfinite\ bd-FGf$
 $\langle proof \rangle$

lemma
fixes $x :: (-, -, 'co1, 'co2, 'contra1, 'contra2, 'f1, 'f2, 'f3, 'f4, 'f5, 'f6, 'f7)\ FGf$
shows $set1-FGf-bound: card-of\ (set1-FGf\ x) \leq o\ (bd-FGf :: ('co1, 'co2, 'contra1,$
 $'contra2,$
 $'f1, 'f2, 'f3, 'f4, 'f5, 'f6, 'f7)\ FGfbd\ rel)$
and $set2-FGf-bound: card-of\ (set2-FGf\ x) \leq o\ (bd-FGf :: ('co1, 'co2, 'contra1,$
 $'contra2,$
 $'f1, 'f2, 'f3, 'f4, 'f5, 'f6, 'f7)\ FGfbd\ rel)$
 $\langle proof \rangle$

lemma $map1-FGf-cong:$
assumes $\bigwedge z. z \in set1-FGf\ x \implies l1\ z = l1'\ z$ **and** $\bigwedge z. z \in set2-FGf\ x \implies l2\ z$
 $= l2'\ z$
shows $map1-FGf\ l1\ l2\ x = map1-FGf\ l1'\ l2'\ x$
 $\langle proof \rangle$

lemma $rell-FGf-mono-strong:$
assumes $rell-FGf\ L1\ L2\ x\ y$
and $\bigwedge a\ b. a \in set1-FGf\ x \implies b \in set1-FGf\ y \implies L1\ a\ b \implies L1'\ a\ b$
and $\bigwedge a\ b. a \in set2-FGf\ x \implies b \in set2-FGf\ y \implies L2\ a\ b \implies L2'\ a\ b$
shows $rell-FGf\ L1'\ L2'\ x\ y$
 $\langle proof \rangle$

end

4 Least and greatest fixpoints

theory $Fixpoints$ **imports**
 $Axiomatised-BNF-CC$
begin

4.1 Least fixpoint

4.1.1 BNF_{CC} structure

context notes $[[\text{typedef-overloaded}, \text{bnf-internals}]]$
begin

datatype (*set-T*: 'l1, 'co1, 'co2, 'contra1, 'contra2, 'f) *T* =
 C-T (*D-T*: (('l1, 'co1, 'co2, 'contra1, 'contra2, 'f) *T*, 'l1, 'co1, 'co2, 'contra1,
 'contra2, 'f) *G*)
 for
 map: *mapl-T*
 rel: *rell-T*
end

inductive *rel-T* :: ('l1 \Rightarrow 'l1' \Rightarrow bool) \Rightarrow
 ('co1 \Rightarrow 'co1' \Rightarrow bool) \Rightarrow ('co2 \Rightarrow 'co2' \Rightarrow bool) \Rightarrow
 ('contra1 \Rightarrow 'contra1' \Rightarrow bool) \Rightarrow ('contra2 \Rightarrow 'contra2' \Rightarrow bool) \Rightarrow
 ('l1, 'co1, 'co2, 'contra1, 'contra2, 'f) *T* \Rightarrow
 ('l1', 'co1', 'co2', 'contra1', 'contra2', 'f) *T* \Rightarrow bool
 for *L1 Co1 Co2 Contra1 Contra2* **where**
 rel-T L1 Co1 Co2 Contra1 Contra2 (*C-T* *x*) (*C-T* *y*)
 if *rel-G* (*rel-T L1 Co1 Co2 Contra1 Contra2*) *L1 Co1 Co2 Contra1 Contra2* *x*
 y

primrec *map-T* :: ('l1 \Rightarrow 'l1') \Rightarrow ('co1 \Rightarrow 'co1') \Rightarrow ('co2 \Rightarrow 'co2') \Rightarrow
 ('contra1' \Rightarrow 'contra1') \Rightarrow ('contra2' \Rightarrow 'contra2') \Rightarrow
 ('l1, 'co1, 'co2, 'contra1, 'contra2, 'f) *T* \Rightarrow
 ('l1', 'co1', 'co2', 'contra1', 'contra2', 'f) *T* **where**
 map-T l1 co1 co2 contra1 contra2 (*C-T* *x*) =
 C-T (*map-G id id co1 co2 contra1 contra2* (*mapl-G* (*map-T l1 co1 co2 contra1*
 contra2) *l1* *x*))

The mapper and relator generated by the datatype package coincide with our generalised definitions restricted to live arguments.

lemma *rell-T-alt-def*: *rell-T L1* = *rel-T L1* (=) (=) (=) (=)
 $\langle \text{proof} \rangle$

lemma *mapl-T-alt-def*: *mapl-T l1* = *map-T l1 id id id id*
 $\langle \text{proof} \rangle$

lemma *rel-T-mono* [*mono*]:
 $[[L1 \leq L1'; Co1 \leq Co1'; Co2 \leq Co2'; Contra1' \leq Contra1; Contra2' \leq Contra2$
 $]] \Rightarrow$
 rel-T L1 Co1 Co2 Contra1 Contra2 \leq *rel-T L1' Co1' Co2' Contra1' Contra2'*
 $\langle \text{proof} \rangle$

lemma *rel-T-eq*: *rel-T* (=) (=) (=) (=) (=) = (=)
 $\langle \text{proof} \rangle$

lemma *rel-T-conversep*:

$rel-T\ L1^{-1-1}\ Co1^{-1-1}\ Co2^{-1-1}\ Contra1^{-1-1}\ Contra2^{-1-1} = (rel-T\ L1\ Co1\ Co2\ Contra1\ Contra2)^{-1-1}$
 ⟨proof⟩

lemma *map-T-id0*: $map-T\ id\ id\ id\ id\ id = id$

⟨proof⟩

lemma *map-T-id*: $map-T\ id\ id\ id\ id\ id\ x = x$

⟨proof⟩

lemma *map-T-comp*: $map-T\ l1\ co1\ co2\ contra1\ contra2 \circ map-T\ l1'\ co1'\ co2'\ contra1'\ contra2' =$

$map-T\ (l1 \circ l1')\ (co1 \circ co1')\ (co2 \circ co2')\ (contra1' \circ contra1)\ (contra2' \circ contra2)$
 ⟨proof⟩

lemma *map-T-parametric*: $rel-fun\ (rel-fun\ L1\ L1')$

$(rel-fun\ (rel-fun\ Co1\ Co1')\ (rel-fun\ (rel-fun\ Co2\ Co2')\ (rel-fun\ (rel-fun\ Contra1'\ Contra1)\ (rel-fun\ (rel-fun\ Contra2'\ Contra2)\ (rel-fun\ (rel-T\ L1\ Co1\ Co2\ Contra1\ Contra2)\ (rel-T\ L1'\ Co1'\ Co2'\ Contra1'\ Contra2'))))))))$
 $map-T\ map-T$
 ⟨proof⟩

definition *rel-T-pos-distr-cond* :: $(co1 \Rightarrow co1' \Rightarrow bool) \Rightarrow (co1' \Rightarrow co1'' \Rightarrow bool) \Rightarrow$

$(co2 \Rightarrow co2' \Rightarrow bool) \Rightarrow (co2' \Rightarrow co2'' \Rightarrow bool) \Rightarrow$
 $(contra1 \Rightarrow contra1' \Rightarrow bool) \Rightarrow (contra1' \Rightarrow contra1'' \Rightarrow bool) \Rightarrow$
 $(contra2 \Rightarrow contra2' \Rightarrow bool) \Rightarrow (contra2' \Rightarrow contra2'' \Rightarrow bool) \Rightarrow$
 $(l1 \times l1' \times l1'' \times f)\ itself \Rightarrow bool$ **where**
 $rel-T-pos-distr-cond\ Co1\ Co1'\ Co2\ Co2'\ Contra1\ Contra1'\ Contra2\ Contra2' -$
 \longleftrightarrow

$(\forall (L1 :: l1 \Rightarrow l1' \Rightarrow bool)\ (L1' :: l1' \Rightarrow l1'' \Rightarrow bool).$
 $(rel-T\ L1\ Co1\ Co2\ Contra1\ Contra2 :: (-, -, -, -, -, f)\ T \Rightarrow -)\ OO$
 $rel-T\ L1'\ Co1'\ Co2'\ Contra1'\ Contra2' \leq$
 $rel-T\ (L1\ OO\ L1')\ (Co1\ OO\ Co1')\ (Co2\ OO\ Co2')\ (Contra1\ OO\ Contra1')\ (Contra2\ OO\ Contra2')$)

definition *rel-T-neg-distr-cond* :: $(co1 \Rightarrow co1' \Rightarrow bool) \Rightarrow (co1' \Rightarrow co1'' \Rightarrow bool) \Rightarrow$

$(co2 \Rightarrow co2' \Rightarrow bool) \Rightarrow (co2' \Rightarrow co2'' \Rightarrow bool) \Rightarrow$
 $(contra1 \Rightarrow contra1' \Rightarrow bool) \Rightarrow (contra1' \Rightarrow contra1'' \Rightarrow bool) \Rightarrow$
 $(contra2 \Rightarrow contra2' \Rightarrow bool) \Rightarrow (contra2' \Rightarrow contra2'' \Rightarrow bool) \Rightarrow$
 $(l1 \times l1' \times l1'' \times f)\ itself \Rightarrow bool$ **where**
 $rel-T-neg-distr-cond\ Co1\ Co1'\ Co2\ Co2'\ Contra1\ Contra1'\ Contra2\ Contra2' -$
 \longleftrightarrow

$(\forall (L1 :: l1 \Rightarrow l1' \Rightarrow bool)\ (L1' :: l1' \Rightarrow l1'' \Rightarrow bool).$

$$\begin{aligned}
& \text{rel-T } (L1 \text{ OO } L1') (Co1 \text{ OO } Co1') (Co2 \text{ OO } Co2') (Contra1 \text{ OO } Contra1') \\
& (Contra2 \text{ OO } Contra2') \leq \\
& (\text{rel-T } L1 \text{ Co1 } Co2 \text{ Contra1 } Contra2 :: (-, -, -, -, -, 'f) T \Rightarrow -) \text{ OO} \\
& \text{rel-T } L1' \text{ Co1}' \text{ Co2}' \text{ Contra1}' \text{ Contra2}'
\end{aligned}$$

We inherit the conditions for subdistributivity over relation composition via a composition witness, which is derived from a witness for the underlying functor G .

primrec *rel-T-witness* :: ('l1 \Rightarrow 'l1'' \Rightarrow bool) \Rightarrow
('co1 \Rightarrow 'co1' \Rightarrow bool) \Rightarrow ('co1' \Rightarrow 'co1'' \Rightarrow bool) \Rightarrow
('co2 \Rightarrow 'co2' \Rightarrow bool) \Rightarrow ('co2' \Rightarrow 'co2'' \Rightarrow bool) \Rightarrow
('contra1 \Rightarrow 'contra1' \Rightarrow bool) \Rightarrow ('contra1' \Rightarrow 'contra1'' \Rightarrow bool) \Rightarrow
('contra2 \Rightarrow 'contra2' \Rightarrow bool) \Rightarrow ('contra2' \Rightarrow 'contra2'' \Rightarrow bool) \Rightarrow
('l1, 'co1, 'co2, 'contra1, 'contra2, 'f) T \Rightarrow
('l1'', 'co1'', 'co2'', 'contra1'', 'contra2'', 'f) T \Rightarrow
('l1 \times 'l1'', 'co1', 'co2', 'contra1', 'contra2', 'f) T **where**
rel-T-witness L1 Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2' (C-T
x) Cy = C-T
(mapl-G (λ ((x, f), y). f y) id
(rel-G-witness (λ (x, f) y. rel-T (λ x (x', y). x' = x \wedge L1 x y) Co1 Co2 Contra1
Contra2 x (f y) \wedge
rel-T (λ (x, y') y. y' = y \wedge L1 x y) Co1' Co2' Contra1' Contra2' (f y) y)
L1 Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2'
(mapl-G (λ x. (x, rel-T-witness L1 Co1 Co1' Co2 Co2' Contra1 Contra1' Con-
tra2 Contra2' x)) id x,
D-T Cy)))

lemma *rel-T-pos-distr-imp*:

fixes Co1 :: 'co1 \Rightarrow 'co1' \Rightarrow bool **and** Co1' :: 'co1' \Rightarrow 'co1'' \Rightarrow bool
and Co2 :: 'co2 \Rightarrow 'co2' \Rightarrow bool **and** Co2' :: 'co2' \Rightarrow 'co2'' \Rightarrow bool
and Contra1 :: 'contra1 \Rightarrow 'contra1' \Rightarrow bool **and** Contra1' :: 'contra1' \Rightarrow
'contra1'' \Rightarrow bool
and Contra2 :: 'contra2 \Rightarrow 'contra2' \Rightarrow bool **and** Contra2' :: 'contra2' \Rightarrow
'contra2'' \Rightarrow bool
and tytok-G :: (('l1, 'co1, 'co2, 'contra1, 'contra2, 'f) T \times
('l1', 'co1', 'co2', 'contra1', 'contra2', 'f) T \times
('l1'', 'co1'', 'co2'', 'contra1'', 'contra2'', 'f) T \times 'l1 \times 'l1' \times 'l1'' \times 'f) itself
and tytok-T :: ('l1 \times 'l1' \times 'l1'' \times 'f) itself
assumes rel-G-pos-distr-cond Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2
Contra2' tytok-G
shows rel-T-pos-distr-cond Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Con-
tra2' tytok-T
<proof>

lemma

fixes L1 :: 'l1 \Rightarrow 'l1'' \Rightarrow bool
and Co1 :: 'co1 \Rightarrow 'co1' \Rightarrow bool **and** Co1' :: 'co1' \Rightarrow 'co1'' \Rightarrow bool
and Co2 :: 'co2 \Rightarrow 'co2' \Rightarrow bool **and** Co2' :: 'co2' \Rightarrow 'co2'' \Rightarrow bool
and Contra1 :: 'contra1 \Rightarrow 'contra1' \Rightarrow bool **and** Contra1' :: 'contra1' \Rightarrow

'contra1'' \Rightarrow *bool*
and *Contra2* :: *'contra2* \Rightarrow *'contra2'* \Rightarrow *bool* **and** *Contra2'* :: *'contra2'* \Rightarrow
'contra2'' \Rightarrow *bool*
and *tytok-G* :: (((*'l1*, *'co1*, *'co2*, *'contra1*, *'contra2*, *'f*) *T* \times
((*'l1''*, *'co1''*, *'co2''*, *'contra1''*, *'contra2''*, *'f*) *T*
 \Rightarrow (*'l1* \times *'l1''*, *'co1'*, *'co2'*, *'contra1'*, *'contra2'*, *'f*) *T*)) \times
(((*'l1*, *'co1*, *'co2*, *'contra1*, *'contra2*, *'f*) *T* \times
((*'l1''*, *'co1''*, *'co2''*, *'contra1''*, *'contra2''*, *'f*) *T*
 \Rightarrow (*'l1* \times *'l1''*, *'co1'*, *'co2'*, *'contra1'*, *'contra2'*, *'f*) *T*)) \times
(*'l1''*, *'co1''*, *'co2''*, *'contra1''*, *'contra2''*, *'f*) *T*) \times
(*'l1''*, *'co1''*, *'co2''*, *'contra1''*, *'contra2''*, *'f*) *T* \times
'l1 \times (*'l1* \times *'l1''*) \times *'l1''* \times *'f*) *itself*
and *x* :: (-, -, -, -, -, *'f*) *T*
assumes *cond*: *rel-G-neg-distr-cond* *Co1* *Co1'* *Co2* *Co2'* *Contra1* *Contra1'* *Contra2* *Contra2'* *tytok-G*
and *rel-OO*: *rel-T* *L1* (*Co1* *OO* *Co1'*) (*Co2* *OO* *Co2'*) (*Contra1* *OO* *Contra1'*)
(*Contra2* *OO* *Contra2'*) *x* *y*
shows *rel-T-witness1*: *rel-T* (λx (*x'*, *y*). *x'* = *x* \wedge *L1* *x* *y*) *Co1* *Co2* *Contra1*
Contra2 *x*
(*rel-T-witness* *L1* *Co1* *Co1'* *Co2* *Co2'* *Contra1* *Contra1'* *Contra2* *Contra2'* *x*
y)
and *rel-T-witness2*: *rel-T* ($\lambda(x, y')$ *y*. *y'* = *y* \wedge *L1* *x* *y*) *Co1'* *Co2'* *Contra1'*
Contra2'
(*rel-T-witness* *L1* *Co1* *Co1'* *Co2* *Co2'* *Contra1* *Contra1'* *Contra2* *Contra2'* *x*
y) *y*
{proof}

lemma *rel-T-neg-distr-imp*:

fixes *Co1* :: *'co1* \Rightarrow *'co1'* \Rightarrow *bool* **and** *Co1'* :: *'co1'* \Rightarrow *'co1''* \Rightarrow *bool*
and *Co2* :: *'co2* \Rightarrow *'co2'* \Rightarrow *bool* **and** *Co2'* :: *'co2'* \Rightarrow *'co2''* \Rightarrow *bool*
and *Contra1* :: *'contra1* \Rightarrow *'contra1'* \Rightarrow *bool* **and** *Contra1'* :: *'contra1'* \Rightarrow
'contra1'' \Rightarrow *bool*
and *Contra2* :: *'contra2* \Rightarrow *'contra2'* \Rightarrow *bool* **and** *Contra2'* :: *'contra2'* \Rightarrow
'contra2'' \Rightarrow *bool*
and *tytok-G* :: (((*'l1*, *'co1*, *'co2*, *'contra1*, *'contra2*, *'f*) *T* \times
((*'l1''*, *'co1''*, *'co2''*, *'contra1''*, *'contra2''*, *'f*) *T*
 \Rightarrow (*'l1* \times *'l1''*, *'co1'*, *'co2'*, *'contra1'*, *'contra2'*, *'f*) *T*)) \times
(((*'l1*, *'co1*, *'co2*, *'contra1*, *'contra2*, *'f*) *T* \times
((*'l1''*, *'co1''*, *'co2''*, *'contra1''*, *'contra2''*, *'f*) *T*
 \Rightarrow (*'l1* \times *'l1''*, *'co1'*, *'co2'*, *'contra1'*, *'contra2'*, *'f*) *T*)) \times
(*'l1''*, *'co1''*, *'co2''*, *'contra1''*, *'contra2''*, *'f*) *T*) \times
(*'l1''*, *'co1''*, *'co2''*, *'contra1''*, *'contra2''*, *'f*) *T* \times
'l1 \times (*'l1* \times *'l1''*) \times *'l1''* \times *'f*) *itself*
and *tytok-T* :: (*'l1* \times *'l1'* \times *'l1''* \times *'f*) *itself*
assumes *rel-G-neg-distr-cond* *Co1* *Co1'* *Co2* *Co2'* *Contra1* *Contra1'* *Contra2*
Contra2' *tytok-G*
shows *rel-T-neg-distr-cond* *Co1* *Co1'* *Co2* *Co2'* *Contra1* *Contra1'* *Contra2* *Contra2'* *tytok-T*
{proof}

lemma *rel-T-pos-distr-cond-eq*:
 $\bigwedge \text{tytok. rel-T-pos-distr-cond } (=) (=) (=) (=) (=) (=) (=) (=) \text{ tytok}$
 $\langle \text{proof} \rangle$

lemma *rel-T-neg-distr-cond-eq*:
 $\bigwedge \text{tytok. rel-T-neg-distr-cond } (=) (=) (=) (=) (=) (=) (=) (=) \text{ tytok}$
 $\langle \text{proof} \rangle$

The BNF axioms are proved by the datatype package.

thm *T.set-map T.bd-card-order T.bd-cinfinite T.set-bd T.map-cong[OF refl]*
T.rel-mono-strong T.wit

4.1.2 Parametricity laws

context includes *lifting-syntax* **begin**

lemma *C-T-parametric*: $(\text{rel-G } (\text{rel-T } L1 \text{ Co1 } Co2 \text{ Contra1 } Contra2) L1 \text{ Co1 } Co2$
 $\text{Contra1 } Contra2 \implies$
 $\text{rel-T } L1 \text{ Co1 } Co2 \text{ Contra1 } Contra2) \text{ C-T C-T}$
 $\langle \text{proof} \rangle$

lemma *D-T-parametric*: $(\text{rel-T } L1 \text{ Co1 } Co2 \text{ Contra1 } Contra2 \implies$
 $\text{rel-G } (\text{rel-T } L1 \text{ Co1 } Co2 \text{ Contra1 } Contra2) L1 \text{ Co1 } Co2 \text{ Contra1 } Contra2) \text{ D-T}$
 D-T
 $\langle \text{proof} \rangle$

lemma *rec-T-parametric*:
 $((\text{rel-G } (\text{rel-prod } (\text{rel-T } L1 \text{ Co1 } Co2 \text{ Contra1 } Contra2) A) L1 \text{ Co1 } Co2 \text{ Contra1}$
 $\text{Contra2} \implies A) \implies$
 $\text{rel-T } L1 \text{ Co1 } Co2 \text{ Contra1 } Contra2 \implies A) \text{ rec-T rec-T}$
 $\langle \text{proof} \rangle$

end

4.2 Greatest fixpoints

4.2.1 BNF_{CC} structure

context notes $[[\text{typedef-overloaded}, \text{bnf-internals}]]$
begin

codatatype $(\text{set-U: 'l1, 'co1, 'co2, 'contra1, 'contra2, 'f}) \text{ U} =$
 $\text{C-U } (\text{D-U: } ((\text{'l1, 'co1, 'co2, 'contra1, 'contra2, 'f}) \text{ U, 'l1, 'co1, 'co2, 'contra1, 'contra2, 'f}) \text{ G})$
for
 map: mapl-U
 rel: rell-U

end

coinductive $rel-U :: ('l1 \Rightarrow 'l1' \Rightarrow bool) \Rightarrow$
 $('co1 \Rightarrow 'co1' \Rightarrow bool) \Rightarrow ('co2 \Rightarrow 'co2' \Rightarrow bool) \Rightarrow$
 $('contra1 \Rightarrow 'contra1' \Rightarrow bool) \Rightarrow ('contra2 \Rightarrow 'contra2' \Rightarrow bool) \Rightarrow$
 $('l1, 'co1, 'co2, 'contra1, 'contra2, 'f) U \Rightarrow$
 $('l1', 'co1', 'co2', 'contra1', 'contra2', 'f) U \Rightarrow bool$
for $L1 Co1 Co2 Contra1 Contra2$ **where**
 $rel-U L1 Co1 Co2 Contra1 Contra2 x y$
if $rel-G (rel-U L1 Co1 Co2 Contra1 Contra2) L1 Co1 Co2 Contra1 Contra2$
 $(D-U x) (D-U y)$

primcorec $map-U :: ('l1 \Rightarrow 'l1') \Rightarrow ('co1 \Rightarrow 'co1') \Rightarrow ('co2 \Rightarrow 'co2') \Rightarrow$
 $('contra1' \Rightarrow 'contra1') \Rightarrow ('contra2' \Rightarrow 'contra2') \Rightarrow$
 $('l1, 'co1, 'co2, 'contra1, 'contra2, 'f) U \Rightarrow$
 $('l1', 'co1', 'co2', 'contra1', 'contra2', 'f) U$ **where**
 $D-U (map-U l1 co1 co2 contra1 contra2 x) =$
 $mapl-G (map-U l1 co1 co2 contra1 contra2) l1 (map-G id id co1 co2 contra1$
 $contra2 (D-U x))$

lemma $rell-U-alt-def: rell-U L1 = rel-U L1 (=) (=) (=) (=)$
 $\langle proof \rangle$

lemma $mapl-U-alt-def: mapl-U l1 = map-U l1 id id id id$
 $\langle proof \rangle$

lemma $rel-U-mono$ [$mono$]:
 $\llbracket L1 \leq L1'; Co1 \leq Co1'; Co2 \leq Co2'; Contra1' \leq Contra1; Contra2' \leq Contra2$
 $\rrbracket \Rightarrow$
 $rel-U L1 Co1 Co2 Contra1 Contra2 \leq rel-U L1' Co1' Co2' Contra1' Contra2'$
 $\langle proof \rangle$

lemma $rel-U-eq: rel-U (=) (=) (=) (=) (=) (=) (=)$
 $\langle proof \rangle$

lemma $rel-U-conversep:$
 $rel-U L1^{-1-1} Co1^{-1-1} Co2^{-1-1} Contra1^{-1-1} Contra2^{-1-1} = (rel-U L1 Co1$
 $Co2 Contra1 Contra2)^{-1-1}$
 $\langle proof \rangle$

lemma $map-U-id0: map-U id id id id id = id$
 $\langle proof \rangle$

lemma $map-U-id: map-U id id id id id x = x$
 $\langle proof \rangle$

lemma $map-U-comp: map-U l1 co1 co2 contra1 contra2 \circ map-U l1' co1' co2'$
 $contra1' contra2' =$
 $map-U (l1 \circ l1') (co1 \circ co1') (co2 \circ co2') (contra1' \circ contra1) (contra2' \circ$
 $contra2)$

$\langle \text{proof} \rangle$

lemma *map-U-parametric*: $\text{rel-fun } (\text{rel-fun } L1 \ L1')$
 $(\text{rel-fun } (\text{rel-fun } Co1 \ Co1') \ (\text{rel-fun } (\text{rel-fun } Co2 \ Co2'))$
 $(\text{rel-fun } (\text{rel-fun } Contra1' \ Contra1) \ (\text{rel-fun } (\text{rel-fun } Contra2' \ Contra2)$
 $(\text{rel-fun } (\text{rel-U } L1 \ Co1 \ Co2 \ Contra1 \ Contra2) \ (\text{rel-U } L1' \ Co1' \ Co2' \ Contra1'$
 $Contra2'))))))$
map-U map-U
 $\langle \text{proof} \rangle$

definition *rel-U-pos-distr-cond* :: $(\text{'co1} \Rightarrow \text{'co1}' \Rightarrow \text{bool}) \Rightarrow (\text{'co1}' \Rightarrow \text{'co1}'' \Rightarrow$
 $\text{bool}) \Rightarrow$
 $(\text{'co2} \Rightarrow \text{'co2}' \Rightarrow \text{bool}) \Rightarrow (\text{'co2}' \Rightarrow \text{'co2}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'contra1} \Rightarrow \text{'contra1}' \Rightarrow \text{bool}) \Rightarrow (\text{'contra1}' \Rightarrow \text{'contra1}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'contra2} \Rightarrow \text{'contra2}' \Rightarrow \text{bool}) \Rightarrow (\text{'contra2}' \Rightarrow \text{'contra2}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'l1} \times \text{'l1}' \times \text{'l1}'' \times \text{'f}) \text{ itself} \Rightarrow \text{bool}$ **where**
rel-U-pos-distr-cond $Co1 \ Co1' \ Co2 \ Co2' \ Contra1 \ Contra1' \ Contra2 \ Contra2' -$
 \longleftrightarrow
 $(\forall (L1 :: \text{'l1} \Rightarrow \text{'l1}' \Rightarrow \text{bool}) (L1' :: \text{'l1}' \Rightarrow \text{'l1}'' \Rightarrow \text{bool}).$
 $(\text{rel-U } L1 \ Co1 \ Co2 \ Contra1 \ Contra2 :: (-, -, -, -, -, \text{'f}) \ U \Rightarrow -) \ OO$
 $\text{rel-U } L1' \ Co1' \ Co2' \ Contra1' \ Contra2' \leq$
 $\text{rel-U } (L1 \ OO \ L1') (Co1 \ OO \ Co1') (Co2 \ OO \ Co2') (Contra1 \ OO \ Contra1')$
 $(Contra2 \ OO \ Contra2'))$

definition *rel-U-neg-distr-cond* :: $(\text{'co1} \Rightarrow \text{'co1}' \Rightarrow \text{bool}) \Rightarrow (\text{'co1}' \Rightarrow \text{'co1}'' \Rightarrow$
 $\text{bool}) \Rightarrow$
 $(\text{'co2} \Rightarrow \text{'co2}' \Rightarrow \text{bool}) \Rightarrow (\text{'co2}' \Rightarrow \text{'co2}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'contra1} \Rightarrow \text{'contra1}' \Rightarrow \text{bool}) \Rightarrow (\text{'contra1}' \Rightarrow \text{'contra1}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'contra2} \Rightarrow \text{'contra2}' \Rightarrow \text{bool}) \Rightarrow (\text{'contra2}' \Rightarrow \text{'contra2}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'l1} \times \text{'l1}' \times \text{'l1}'' \times \text{'f}) \text{ itself} \Rightarrow \text{bool}$ **where**
rel-U-neg-distr-cond $Co1 \ Co1' \ Co2 \ Co2' \ Contra1 \ Contra1' \ Contra2 \ Contra2' -$
 \longleftrightarrow
 $(\forall (L1 :: \text{'l1} \Rightarrow \text{'l1}' \Rightarrow \text{bool}) (L1' :: \text{'l1}' \Rightarrow \text{'l1}'' \Rightarrow \text{bool}).$
 $\text{rel-U } (L1 \ OO \ L1') (Co1 \ OO \ Co1') (Co2 \ OO \ Co2') (Contra1 \ OO \ Contra1')$
 $(Contra2 \ OO \ Contra2')) \leq$
 $(\text{rel-U } L1 \ Co1 \ Co2 \ Contra1 \ Contra2 :: (-, -, -, -, -, \text{'f}) \ U \Rightarrow -) \ OO$
 $\text{rel-U } L1' \ Co1' \ Co2' \ Contra1' \ Contra2')$

primcorec *rel-U-witness* :: $(\text{'l1} \Rightarrow \text{'l1}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'co1} \Rightarrow \text{'co1}' \Rightarrow \text{bool}) \Rightarrow (\text{'co1}' \Rightarrow \text{'co1}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'co2} \Rightarrow \text{'co2}' \Rightarrow \text{bool}) \Rightarrow (\text{'co2}' \Rightarrow \text{'co2}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'contra1} \Rightarrow \text{'contra1}' \Rightarrow \text{bool}) \Rightarrow (\text{'contra1}' \Rightarrow \text{'contra1}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'contra2} \Rightarrow \text{'contra2}' \Rightarrow \text{bool}) \Rightarrow (\text{'contra2}' \Rightarrow \text{'contra2}'' \Rightarrow \text{bool}) \Rightarrow$
 $(\text{'l1}, \text{'co1}, \text{'co2}, \text{'contra1}, \text{'contra2}, \text{'f}) \ U \times$
 $(\text{'l1}'', \text{'co1}'', \text{'co2}'', \text{'contra1}'', \text{'contra2}'', \text{'f}) \ U \Rightarrow$
 $(\text{'l1} \times \text{'l1}'', \text{'co1}', \text{'co2}', \text{'contra1}', \text{'contra2}', \text{'f}) \ U$ **where**
D-U $(\text{rel-U-witness } L1 \ Co1 \ Co1' \ Co2 \ Co2' \ Contra1 \ Contra1' \ Contra2 \ Contra2'$
 $xy) =$
mapl-G $(\text{rel-U-witness } L1 \ Co1 \ Co1' \ Co2 \ Co2' \ Contra1 \ Contra1' \ Contra2 \ Con-$

$tra2'$) id
 $(rel-G-witness (rel-U L1 (Co1 OO Co1') (Co2 OO Co2') (Contra1 OO Contra1')$
 $(Contra2 OO Contra2'))$
 $L1 Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2' (D-U (fst xy), D-U$
 $(snd xy)))$

lemma *rel-U-pos-distr-imp:*

fixes $Co1 :: 'co1 \Rightarrow 'co1' \Rightarrow bool$ **and** $Co1' :: 'co1' \Rightarrow 'co1'' \Rightarrow bool$
and $Co2 :: 'co2 \Rightarrow 'co2' \Rightarrow bool$ **and** $Co2' :: 'co2' \Rightarrow 'co2'' \Rightarrow bool$
and $Contra1 :: 'contra1 \Rightarrow 'contra1' \Rightarrow bool$ **and** $Contra1' :: 'contra1' \Rightarrow$
 $'contra1'' \Rightarrow bool$
and $Contra2 :: 'contra2 \Rightarrow 'contra2' \Rightarrow bool$ **and** $Contra2' :: 'contra2' \Rightarrow$
 $'contra2'' \Rightarrow bool$
and $tytok-G :: (('l1, 'co1, 'co2, 'contra1, 'contra2, 'f) U \times$
 $('l1', 'co1', 'co2', 'contra1', 'contra2', 'f) U \times$
 $('l1'', 'co1'', 'co2'', 'contra1'', 'contra2'', 'f) U \times 'l1 \times 'l1' \times 'l1'' \times 'f) itself$
and $tytok-T :: ('l1 \times 'l1' \times 'l1'' \times 'f) itself$
assumes $rel-G-pos-distr-cond Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2$
 $Contra2' tytok-G$
shows $rel-U-pos-distr-cond Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Con-$
 $tra2' tytok-T$
 $\langle proof \rangle$

lemma *rel-U-witness1:*

fixes $L1 :: 'l1 \Rightarrow 'l1'' \Rightarrow bool$
and $Co1 :: 'co1 \Rightarrow 'co1' \Rightarrow bool$ **and** $Co1' :: 'co1' \Rightarrow 'co1'' \Rightarrow bool$
and $Co2 :: 'co2 \Rightarrow 'co2' \Rightarrow bool$ **and** $Co2' :: 'co2' \Rightarrow 'co2'' \Rightarrow bool$
and $Contra1 :: 'contra1 \Rightarrow 'contra1' \Rightarrow bool$ **and** $Contra1' :: 'contra1' \Rightarrow$
 $'contra1'' \Rightarrow bool$
and $Contra2 :: 'contra2 \Rightarrow 'contra2' \Rightarrow bool$ **and** $Contra2' :: 'contra2' \Rightarrow$
 $'contra2'' \Rightarrow bool$
and $tytok-G :: (('l1, 'co1, 'co2, 'contra1, 'contra2, 'f) U \times$
 $(('l1, 'co1, 'co2, 'contra1, 'contra2, 'f) U \times$
 $('l1'', 'co1'', 'co2'', 'contra1'', 'contra2'', 'f) U) \times$
 $('l1'', 'co1'', 'co2'', 'contra1'', 'contra2'', 'f) U \times$
 $'l1 \times ('l1 \times 'l1'') \times 'l1'' \times 'f) itself$
and $x :: (-, -, -, -, -, 'f) U$
assumes $cond: rel-G-neg-distr-cond Co1 Co1' Co2 Co2' Contra1 Contra1' Con-$
 $tra2 Contra2' tytok-G$
and $rel-OO: rel-U L1 (Co1 OO Co1') (Co2 OO Co2') (Contra1 OO Contra1')$
 $(Contra2 OO Contra2') x y$
shows $rel-U (\lambda x (x', y). x' = x \wedge L1 x y) Co1 Co2 Contra1 Contra2 x$
 $(rel-U-witness L1 Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2'$
 $(x, y))$
 $\langle proof \rangle$

lemma *rel-U-witness2:*

fixes $L1 :: 'l1 \Rightarrow 'l1'' \Rightarrow bool$
and $Co1 :: 'co1 \Rightarrow 'co1' \Rightarrow bool$ **and** $Co1' :: 'co1' \Rightarrow 'co1'' \Rightarrow bool$

and $Co2 :: 'co2 \Rightarrow 'co2' \Rightarrow bool$ **and** $Co2' :: 'co2' \Rightarrow 'co2'' \Rightarrow bool$
and $Contra1 :: 'contra1 \Rightarrow 'contra1' \Rightarrow bool$ **and** $Contra1' :: 'contra1' \Rightarrow 'contra1'' \Rightarrow bool$
and $Contra2 :: 'contra2 \Rightarrow 'contra2' \Rightarrow bool$ **and** $Contra2' :: 'contra2' \Rightarrow 'contra2'' \Rightarrow bool$
and $tytok-G :: (('l1, 'co1, 'co2, 'contra1, 'contra2, 'f) U \times ('l1, 'co1, 'co2, 'contra1, 'contra2, 'f) U \times ('l1'', 'co1'', 'co2'', 'contra1'', 'contra2'', 'f) U) \times ('l1'', 'co1'', 'co2'', 'contra1'', 'contra2'', 'f) U \times 'l1 \times ('l1 \times 'l1'') \times 'l1'' \times 'f) itself$
and $x :: (-, -, -, -, -, 'f) U$
assumes $cond: rel-G-neg-distr-cond Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2' tytok-G$
and $rel-OO: rel-U L1 (Co1 OO Co1') (Co2 OO Co2') (Contra1 OO Contra1') (Contra2 OO Contra2') x y$
shows $rel-U (\lambda(x, y') y. y' = y \wedge L1 x y) Co1' Co2' Contra1' Contra2'$
 $(rel-U-witness L1 Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2' (x, y)) y$
 $\langle proof \rangle$

lemma $rel-U-neg-distr-imp:$

fixes $Co1 :: 'co1 \Rightarrow 'co1' \Rightarrow bool$ **and** $Co1' :: 'co1' \Rightarrow 'co1'' \Rightarrow bool$
and $Co2 :: 'co2 \Rightarrow 'co2' \Rightarrow bool$ **and** $Co2' :: 'co2' \Rightarrow 'co2'' \Rightarrow bool$
and $Contra1 :: 'contra1 \Rightarrow 'contra1' \Rightarrow bool$ **and** $Contra1' :: 'contra1' \Rightarrow 'contra1'' \Rightarrow bool$
and $Contra2 :: 'contra2 \Rightarrow 'contra2' \Rightarrow bool$ **and** $Contra2' :: 'contra2' \Rightarrow 'contra2'' \Rightarrow bool$
and $tytok-G :: (('l1, 'co1, 'co2, 'contra1, 'contra2, 'f) U \times ('l1, 'co1, 'co2, 'contra1, 'contra2, 'f) U \times ('l1'', 'co1'', 'co2'', 'contra1'', 'contra2'', 'f) U) \times ('l1'', 'co1'', 'co2'', 'contra1'', 'contra2'', 'f) U \times 'l1 \times ('l1 \times 'l1'') \times 'l1'' \times 'f) itself$
and $tytok-T :: ('l1 \times 'l1' \times 'l1'' \times 'f) itself$
assumes $rel-G-neg-distr-cond Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2' tytok-G$
shows $rel-U-neg-distr-cond Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2' tytok-T$
 $\langle proof \rangle$

lemma $rel-U-pos-distr-cond-eq:$

$\bigwedge tytok. rel-U-pos-distr-cond (=) (=) (=) (=) (=) (=) (=) (=) tytok$
 $\langle proof \rangle$

lemma $rel-U-neg-distr-cond-eq:$

$\bigwedge tytok. rel-U-neg-distr-cond (=) (=) (=) (=) (=) (=) (=) (=) tytok$
 $\langle proof \rangle$

The BNF axioms are proved by the datatype package.

thm $U.set-map U.bd-card-order U.bd-cinfinite U.set-bd U.map-cong[OF refl]$

U.rel-mono-strong U.wit

4.2.2 Parametricity laws

context includes *lifting-syntax* **begin**

lemma *C-U-parametric*: $(rel-G (rel-U L1 Co1 Co2 Contra1 Contra2) L1 Co1 Co2 Contra1 Contra2) ==>$
 $rel-U L1 Co1 Co2 Contra1 Contra2) C-U C-U$
<proof>

lemma *D-U-parametric*: $(rel-U L1 Co1 Co2 Contra1 Contra2 ==>$
 $rel-G (rel-U L1 Co1 Co2 Contra1 Contra2) L1 Co1 Co2 Contra1 Contra2) D-U$
D-U
<proof>

lemma *corec-U-parametric*:
 $((A ==> rel-G (rel-sum (rel-U L1 Co1 Co2 Contra1 Contra2) A) L1 Co1 Co2 Contra1 Contra2) ==>$
 $A ==> rel-U L1 Co1 Co2 Contra1 Contra2) corec-U corec-U$
<proof>

end

end

5 Subtypes

theory *Subtypes* **imports**
Axiomatised-BNF-CC
HOL-Library.BNF-Axiomatization
begin

5.1 BNF_{CC} structure

consts $P :: ('live1, 'live2, 'co1, 'co2, 'contra1, 'contra2, 'fixed) G \Rightarrow bool$

axiomatization **where**

$P\text{-map}: \bigwedge x\ l1\ l2\ co1\ co2\ contra1\ contra2. P\ x \implies P\ (\text{map-G}\ l1\ l2\ co1\ co2\ contra1\ contra2\ x)$

— $\{x. P\ x\}$ is closed under the mapper of G

and

$ex\text{-}P: \exists x. P\ x$ — $\{x. P\ x\}$ is non-empty

typedef (**overloaded**) $('live1, 'live2, 'co1, 'co2, 'contra1, 'contra2, 'fixed) S =$
 $\{x :: ('live1, 'live2, 'co1, 'co2, 'contra1, 'contra2, 'fixed) G. P\ x\}$ *<proof>*

The subtype S is isomorphic to the set $\{x. P\ x\}$.

context includes *lifting-syntax*

begin

definition $rel-S :: ('live1 \Rightarrow 'live1' \Rightarrow bool) \Rightarrow ('live2 \Rightarrow 'live2' \Rightarrow bool) \Rightarrow$
 $('co1 \Rightarrow 'co1' \Rightarrow bool) \Rightarrow ('co2 \Rightarrow 'co2' \Rightarrow bool) \Rightarrow$
 $('contra1 \Rightarrow 'contra1' \Rightarrow bool) \Rightarrow ('contra2 \Rightarrow 'contra2' \Rightarrow bool) \Rightarrow$
 $('live1, 'live2, 'co1, 'co2, 'contra1, 'contra2, 'fixed) S \Rightarrow$
 $('live1', 'live2', 'co1', 'co2', 'contra1', 'contra2', 'fixed) S \Rightarrow bool$

where

$rel-S L1 L2 Co1 Co2 Contra1 Contra2 = vimage2p Rep-S Rep-S (rel-G L1 L2$
 $Co1 Co2 Contra1 Contra2)$

definition $map-S :: ('live1 \Rightarrow 'live1') \Rightarrow ('live2 \Rightarrow 'live2') \Rightarrow$
 $('co1 \Rightarrow 'co1') \Rightarrow ('co2 \Rightarrow 'co2') \Rightarrow$
 $('contra1' \Rightarrow 'contra1') \Rightarrow ('contra2' \Rightarrow 'contra2') \Rightarrow$
 $('live1, 'live2, 'co1, 'co2, 'contra1, 'contra2, 'fixed) S \Rightarrow$
 $('live1', 'live2', 'co1', 'co2', 'contra1', 'contra2', 'fixed) S$

where

$map-S = (id \dashrightarrow id \dashrightarrow id \dashrightarrow id \dashrightarrow id \dashrightarrow id \dashrightarrow id \dashrightarrow Rep-S$
 $\dashrightarrow Abs-S) map-G$

lemma $rel-S-mono$:

$\llbracket L1 \leq L1'; L2 \leq L2'; Co1 \leq Co1'; Co2 \leq Co2'; Contra1' \leq Contra1; Contra2'$
 $\leq Contra2 \rrbracket$
 $\implies rel-S L1 L2 Co1 Co2 Contra1 Contra2 \leq rel-S L1' L2' Co1' Co2' Contra1'$
 $Contra2'$
 $\langle proof \rangle$

lemma $rel-S-eq$: $rel-S (=) (=) (=) (=) (=) (=) (=) (=)$

$\langle proof \rangle$

lemma $rel-S-conversep$:

$rel-S L1^{-1-1} L2^{-1-1} Co1^{-1-1} Co2^{-1-1} Contra1^{-1-1} Contra2^{-1-1} = (rel-S$
 $L1 L2 Co1 Co2 Contra1 Contra2)^{-1-1}$
 $\langle proof \rangle$

lemma $map-S-id0$: $map-S id id id id id id = id$

$\langle proof \rangle$

lemma $map-S-id$: $map-S id id id id id id x = x$

$\langle proof \rangle$

lemma $map-S-comp$:

$map-S l1 l2 co1 co2 contra1 contra2 \circ map-S l1' l2' co1' co2' contra1' contra2'$
 $=$
 $map-S (l1 \circ l1') (l2 \circ l2') (co1 \circ co1') (co2 \circ co2') (contra1' \circ contra1) (contra2'$
 $\circ contra2)$
 $\langle proof \rangle$

lemma $map-S-parametric$:

```

((L1 ==> L1') ==> (L2 ==> L2') ==> (Co1 ==> Co1') ==>
(Co2 ==> Co2') ==>
  (Contra1' ==> Contra1) ==> (Contra2' ==> Contra2) ==>
  rel-S L1 L2 Co1 Co2 Contra1 Contra2 ==> rel-S L1' L2' Co1' Co2' Contra1'
  Contra2')
  map-S map-S
  ⟨proof⟩

```

lemmas *map-S-rel-cong* = *map-S-parametric*[*unfolded rel-fun-def*, *rule-format*, *rotated -1*]

end

definition *rel-S-pos-distr-cond* :: ('co1 ⇒ 'co1' ⇒ bool) ⇒ ('co1' ⇒ 'co1'' ⇒ bool) ⇒

```

⇒
  ('co2 ⇒ 'co2' ⇒ bool) ⇒ ('co2' ⇒ 'co2'' ⇒ bool) ⇒
  ('contra1 ⇒ 'contra1' ⇒ bool) ⇒ ('contra1' ⇒ 'contra1'' ⇒ bool) ⇒
  ('contra2 ⇒ 'contra2' ⇒ bool) ⇒ ('contra2' ⇒ 'contra2'' ⇒ bool) ⇒
  ('l1 × 'l1' × 'l1'' × 'l2 × 'l2' × 'l2'' × 'f) itself ⇒ bool where
  rel-S-pos-distr-cond Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2' -
  ←→
  (∀ (L1 :: 'l1 ⇒ 'l1' ⇒ bool) (L1' :: 'l1' ⇒ 'l1'' ⇒ bool)
   (L2 :: 'l2 ⇒ 'l2' ⇒ bool) (L2' :: 'l2' ⇒ 'l2'' ⇒ bool).
   (rel-S L1 L2 Co1 Co2 Contra1 Contra2 :: (-, -, -, -, -, -, 'f) S ⇒ -) OO
   rel-S L1' L2' Co1' Co2' Contra1' Contra2' ≤
   rel-S (L1 OO L1') (L2 OO L2') (Co1 OO Co1') (Co2 OO Co2')
   (Contra1 OO Contra1') (Contra2 OO Contra2'))

```

definition *rel-S-neg-distr-cond* :: ('co1 ⇒ 'co1' ⇒ bool) ⇒ ('co1' ⇒ 'co1'' ⇒ bool) ⇒

```

⇒
  ('co2 ⇒ 'co2' ⇒ bool) ⇒ ('co2' ⇒ 'co2'' ⇒ bool) ⇒
  ('contra1 ⇒ 'contra1' ⇒ bool) ⇒ ('contra1' ⇒ 'contra1'' ⇒ bool) ⇒
  ('contra2 ⇒ 'contra2' ⇒ bool) ⇒ ('contra2' ⇒ 'contra2'' ⇒ bool) ⇒
  ('l1 × 'l1' × 'l1'' × 'l2 × 'l2' × 'l2'' × 'f) itself ⇒ bool where
  rel-S-neg-distr-cond Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2' -
  ←→
  (∀ (L1 :: 'l1 ⇒ 'l1' ⇒ bool) (L1' :: 'l1' ⇒ 'l1'' ⇒ bool)
   (L2 :: 'l2 ⇒ 'l2' ⇒ bool) (L2' :: 'l2' ⇒ 'l2'' ⇒ bool).
   rel-S (L1 OO L1') (L2 OO L2') (Co1 OO Co1') (Co2 OO Co2')
   (Contra1 OO Contra1') (Contra2 OO Contra2') ≤
   (rel-S L1 L2 Co1 Co2 Contra1 Contra2 :: (-, -, -, -, -, -, 'f) S ⇒ -) OO
   rel-S L1' L2' Co1' Co2' Contra1' Contra2')

```

axiomatization where

rel-S-neg-distr-cond-eq:

⊡*tytok*. *rel-S-neg-distr-cond* (=) (=) (=) (=) (=) (=) (=) (=) *tytok*

The subtype inherits the conditions for positive subdistributivity.

lemma *rel-S-pos-distr-imp*:

fixes $Co1 :: 'co1 \Rightarrow 'co1' \Rightarrow bool$ **and** $Co1' :: 'co1' \Rightarrow 'co1'' \Rightarrow bool$
and $Co2 :: 'co2 \Rightarrow 'co2' \Rightarrow bool$ **and** $Co2' :: 'co2' \Rightarrow 'co2'' \Rightarrow bool$
and $Contra1 :: 'contra1 \Rightarrow 'contra1' \Rightarrow bool$ **and** $Contra1' :: 'contra1' \Rightarrow 'contra1'' \Rightarrow bool$
and $Contra2 :: 'contra2 \Rightarrow 'contra2' \Rightarrow bool$ **and** $Contra2' :: 'contra2' \Rightarrow 'contra2'' \Rightarrow bool$
and $tytok-G :: ('l1 \times 'l1' \times 'l1'' \times 'l2 \times 'l2' \times 'l2'' \times 'f)$ *itself*
and $tytok-S :: ('l1 \times 'l1' \times 'l1'' \times 'l2 \times 'l2' \times 'l2'' \times 'f)$ *itself*
assumes $rel-G-pos-distr-cond$ $Co1$ $Co1'$ $Co2$ $Co2'$ $Contra1$ $Contra1'$ $Contra2$ $Contra2'$ $tytok-G$
shows $rel-S-pos-distr-cond$ $Co1$ $Co1'$ $Co2$ $Co2'$ $Contra1$ $Contra1'$ $Contra2$ $Contra2'$ $tytok-S$
<proof>

lemma $rel-S-pos-distr-cond-eq$:

$\bigwedge tytok. rel-S-pos-distr-cond (=) (=) (=) (=) (=) (=) (=) (=) tytok$
<proof>

lemmas

$rel-S-pos-distr = rel-S-pos-distr-cond-def[THEN iffD1, rule-format]$ **and**
 $rel-S-neg-distr = rel-S-neg-distr-cond-def[THEN iffD1, rule-format]$

The following composition witness depends only on the abstract condition $rel-S-neg-distr-cond$, without additional assumptions.

consts

$rel-S-witness :: ('l1 \Rightarrow 'l1'' \Rightarrow bool) \Rightarrow ('l2 \Rightarrow 'l2'' \Rightarrow bool) \Rightarrow$
 $('co1 \Rightarrow 'co1' \Rightarrow bool) \Rightarrow ('co1' \Rightarrow 'co1'' \Rightarrow bool) \Rightarrow$
 $('co2 \Rightarrow 'co2' \Rightarrow bool) \Rightarrow ('co2' \Rightarrow 'co2'' \Rightarrow bool) \Rightarrow$
 $('contra1 \Rightarrow 'contra1' \Rightarrow bool) \Rightarrow ('contra1' \Rightarrow 'contra1'' \Rightarrow bool) \Rightarrow$
 $('contra2 \Rightarrow 'contra2' \Rightarrow bool) \Rightarrow ('contra2' \Rightarrow 'contra2'' \Rightarrow bool) \Rightarrow$
 $('l1, 'l2, 'co1, 'co2, 'contra1, 'contra2, 'f) S \times$
 $('l1'', 'l2'', 'co1'', 'co2'', 'contra1'', 'contra2'', 'f) S \Rightarrow$
 $('l1 \times 'l1'', 'l2 \times 'l2'', 'co1', 'co2', 'contra1', 'contra2', 'f) S$

specification ($rel-S-witness$)

$rel-S-witness1: \bigwedge L1 L2 Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2'$
 $(tytok :: ('l1 \times ('l1 \times 'l1'') \times 'l1'' \times 'l2 \times ('l2 \times 'l2'') \times 'l2'' \times 'f)$ *itself*)
 $(x :: ('l1, 'l2, -, -, -, 'f) S) (y :: ('l1'', 'l2'', -, -, -, 'f) S).$
 $\llbracket rel-S-neg-distr-cond Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2'$
 $tytok;$
 $rel-S L1 L2 (Co1 OO Co1') (Co2 OO Co2') (Contra1 OO Contra1') (Contra2$
 $OO Contra2') x y \rrbracket \Longrightarrow$
 $rel-S (\lambda x (x', y). x' = x \wedge L1 x y) (\lambda x (x', y). x' = x \wedge L2 x y) Co1 Co2$
 $Contra1 Contra2 x$
 $(rel-S-witness L1 L2 Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2'$
 $(x, y))$
 $rel-S-witness2: \bigwedge L1 L2 Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2'$
 $(tytok :: ('l1 \times ('l1 \times 'l1'') \times 'l1'' \times 'l2 \times ('l2 \times 'l2'') \times 'l2'' \times 'f)$ *itself*)
 $(x :: ('l1, 'l2, -, -, -, 'f) S) (y :: ('l1'', 'l2'', -, -, -, 'f) S).$

$\llbracket \text{rel-S-neg-distr-cond } Co1 \ Co1' \ Co2 \ Co2' \ Contra1 \ Contra1' \ Contra2 \ Contra2' \text{ tytok};$
 $\text{rel-S } L1 \ L2 \ (Co1 \ OO \ Co1') \ (Co2 \ OO \ Co2') \ (Contra1 \ OO \ Contra1') \ (Contra2 \ OO \ Contra2') \ x \ y \rrbracket \implies$
 $\text{rel-S } (\lambda(x, y') \ y. \ y' = y \wedge L1 \ x \ y) \ (\lambda(x, y') \ y. \ y' = y \wedge L2 \ x \ y) \ Co1' \ Co2' \ Contra1' \ Contra2'$
 $(\text{rel-S-witness } L1 \ L2 \ Co1 \ Co1' \ Co2 \ Co2' \ Contra1 \ Contra1' \ Contra2 \ Contra2' \ (x, y)) \ y$
 $\langle \text{proof} \rangle$

definition $\text{set1-S} :: ('live1, 'live2, 'co1, 'co2, 'contra1, 'contra2, 'fixed) S \Rightarrow 'live1 \text{ set}$

where $\text{set1-S} = \text{set1-G} \circ \text{Rep-S}$

definition $\text{set2-S} :: ('live1, 'live2, 'co1, 'co2, 'contra1, 'contra2, 'fixed) S \Rightarrow 'live2 \text{ set}$

where $\text{set2-S} = \text{set2-G} \circ \text{Rep-S}$

lemma rel-S-alt :

$\text{rel-S } L1 \ L2 \ (=) \ (=) \ (=) \ (=) \ x \ y \longleftrightarrow (\exists z. (\text{set1-S } z \subseteq \{(x, y). L1 \ x \ y\} \wedge$
 $\text{set2-S } z \subseteq \{(x, y). L2 \ x \ y\}) \wedge \text{map-S } \text{fst } \text{fst } \text{id } \text{id } \text{id } \text{id } \text{id } z = x \wedge \text{map-S } \text{snd } \text{snd } \text{id } \text{id } \text{id } \text{id } z = y)$
 $\langle \text{proof} \rangle$

bnf $('live1, 'live2, 'co1, 'co2, 'contra1, 'contra2, 'fixed) S$

$\text{map}: \lambda l1 \ l2. \ \text{map-S } l1 \ l2 \ \text{id } \text{id } \text{id } \text{id}$

$\text{sets}: \text{set1-S} \quad \text{set2-S}$

$\text{bd}: \text{bd-G} :: ('co1, 'co2, 'contra1, 'contra2, 'fixed) \text{ Gbd } \text{rel}$

$\text{rel}: \lambda L1 \ L2. \ \text{rel-S } L1 \ L2 \ (=) \ (=) \ (=) \ (=)$

$\langle \text{proof} \rangle$

5.2 Closedness under zippings

lemma $P\text{-zip-closed}$: — This is **lift-bnf**'s property that is too strong.

assumes $P \ (\text{mapl-G } \text{fst } \text{fst } z) \quad P \ (\text{mapl-G } \text{snd } \text{snd } z)$

shows $P \ z$

$\langle \text{proof} \rangle$

consts $\text{rel-S-neg-distr-cond}' :: ('co1 \Rightarrow 'co1' \Rightarrow \text{bool}) \Rightarrow ('co1' \Rightarrow 'co1'' \Rightarrow \text{bool}) \Rightarrow$

\Rightarrow
 $('co2 \Rightarrow 'co2' \Rightarrow \text{bool}) \Rightarrow ('co2' \Rightarrow 'co2'' \Rightarrow \text{bool}) \Rightarrow$
 $('contra1 \Rightarrow 'contra1' \Rightarrow \text{bool}) \Rightarrow ('contra1' \Rightarrow 'contra1'' \Rightarrow \text{bool}) \Rightarrow$
 $('contra2 \Rightarrow 'contra2' \Rightarrow \text{bool}) \Rightarrow ('contra2' \Rightarrow 'contra2'' \Rightarrow \text{bool}) \Rightarrow$
 $('l1 \times 'l1' \times 'l1'' \times 'l2 \times 'l2' \times 'l2'' \times 'f) \text{ itself} \Rightarrow \text{bool}$

If the set $\{x. P \ x\}$ is closed under zippings for $\text{rel-S-neg-distr-cond}'$, we inherit the condition for negative subdistributivity from G .

axiomatization **where**

$P\text{-rel-G-zipping}: \bigwedge (L1 :: 'l1 \Rightarrow 'l1'' \Rightarrow \text{bool}) \ (L2 :: 'l2 \Rightarrow 'l2'' \Rightarrow \text{bool})$

$Co1\ Co1'\ Co2\ Co2'\ Contra1\ Contra1'\ Contra2\ Contra2'$
 $(tytok :: ('l1 \times ('l1 \times 'l1'') \times 'l1'' \times 'l2 \times ('l2 \times 'l2'') \times 'l2'' \times 'f) \text{ itself})\ x$
 $y\ z.$
 $\llbracket P\ x; P\ y;$
 $\text{rel-G } L1\ L2\ (Co1\ OO\ Co1')\ (Co2\ OO\ Co2')\ (Contra1\ OO\ Contra1')\ (Contra2$
 $OO\ Contra2')\ x\ y;$
 $\text{rel-G } (\lambda x\ (x',\ y).\ x' = x \wedge L1\ x\ y)\ (\lambda x\ (x',\ y).\ x' = x \wedge L2\ x\ y)\ Co1\ Co2$
 $Contra1\ Contra2\ x\ z;$
 $\text{rel-G } (\lambda(x,\ y')\ y.\ y' = y \wedge L1\ x\ y)\ (\lambda(x,\ y')\ y.\ y' = y \wedge L2\ x\ y)\ Co1'\ Co2'$
 $Contra1'\ Contra2'\ z\ y;$
 $\text{rel-S-neg-distr-cond}'\ Co1\ Co1'\ Co2\ Co2'\ Contra1\ Contra1'\ Contra2\ Contra2'$
 $tytok\ \rrbracket$
 $\implies P\ z$
and
 $\text{rel-S-neg-distr-cond}'\text{-stronger}: \bigwedge Co1\ Co1'\ Co2\ Co2'\ Contra1\ Contra1'\ Contra2$
 $Contra2'\ tytok.$
 $\text{rel-S-neg-distr-cond}'\ Co1\ Co1'\ Co2\ Co2'\ Contra1\ Contra1'\ Contra2\ Contra2'$
 $tytok \implies$
 $\text{rel-G-neg-distr-cond}\ Co1\ Co1'\ Co2\ Co2'\ Contra1\ Contra1'\ Contra2\ Contra2'$
 $tytok$
and
 $\text{rel-S-neg-distr-cond}'\text{-eq}:$
 $\bigwedge tytok.\ \text{rel-S-neg-distr-cond}'\ (=)\ (=)\ (=)\ (=)\ (=)\ (=)\ (=)\ (=)\ tytok$

context includes *lifting-syntax*
begin

definition $\text{rel-S-witness}' :: ('live1 \Rightarrow 'live1'' \Rightarrow bool) \Rightarrow ('live2 \Rightarrow 'live2'' \Rightarrow bool)$
 \Rightarrow
 $('co1 \Rightarrow 'co1' \Rightarrow bool) \Rightarrow ('co1' \Rightarrow 'co1'' \Rightarrow bool) \Rightarrow$
 $('co2 \Rightarrow 'co2' \Rightarrow bool) \Rightarrow ('co2' \Rightarrow 'co2'' \Rightarrow bool) \Rightarrow$
 $('contra1 \Rightarrow 'contra1' \Rightarrow bool) \Rightarrow ('contra1' \Rightarrow 'contra1'' \Rightarrow bool) \Rightarrow$
 $('contra2 \Rightarrow 'contra2' \Rightarrow bool) \Rightarrow ('contra2' \Rightarrow 'contra2'' \Rightarrow bool) \Rightarrow$
 $('live1,\ 'live2,\ 'co1,\ 'co2,\ 'contra1,\ 'contra2,\ 'fixed)\ S \times$
 $('live1'', 'live2'', 'co1'', 'co2'', 'contra1'', 'contra2'', 'fixed)\ S \Rightarrow$
 $('live1 \times 'live1'', 'live2 \times 'live2'', 'co1', 'co2', 'contra1', 'contra2', 'fixed)\ S$
where
 $\text{rel-S-witness}' = (id \text{ ----} \> id \text{ ----} \> id \text{ ----} \> id \text{ ----} \> id \text{ ----} \> id \text{ ----} \>$
 $id \text{ ----} \> id \text{ ----} \> id \text{ ----} \> id \text{ ----} \> \text{map-prod Rep-S Rep-S ----} \> \text{Abs-S})$
 rel-G-witness

lemma $\text{rel-S-witness}'1:$

fixes $L1 :: 'l1 \Rightarrow 'l1'' \Rightarrow bool$ **and** $L2 :: 'l2 \Rightarrow 'l2'' \Rightarrow bool$
and $Co1 :: 'co1 \Rightarrow 'co1' \Rightarrow bool$ **and** $Co1' :: 'co1' \Rightarrow 'co1'' \Rightarrow bool$
and $Co2 :: 'co2 \Rightarrow 'co2' \Rightarrow bool$ **and** $Co2' :: 'co2' \Rightarrow 'co2'' \Rightarrow bool$
and $Contra1 :: 'contra1 \Rightarrow 'contra1' \Rightarrow bool$ **and** $Contra1' :: 'contra1' \Rightarrow$
 $'contra1'' \Rightarrow bool$
and $Contra2 :: 'contra2 \Rightarrow 'contra2' \Rightarrow bool$ **and** $Contra2' :: 'contra2' \Rightarrow$
 $'contra2'' \Rightarrow bool$

and *tytok* :: ('l1 × ('l1 × 'l1'') × 'l1'' × 'l2 × ('l2 × 'l2'') × 'l2'' × 'f) *itself*
and *x* :: (-, -, -, -, -, -, 'f) *S*
assumes *rel-S L1 L2 (Co1 OO Co1') (Co2 OO Co2') (Contra1 OO Contra1')*
(*Contra2 OO Contra2'*) *x y*
and *rel-S-neg-distr-cond' Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2'* *tytok*
shows *rel-S (λx (x', y). x' = x ∧ L1 x y) (λx (x', y). x' = x ∧ L2 x y) Co1 Co2*
Contra1 Contra2 x
(*rel-S-witness' L1 L2 Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2'*
(*x, y*))
{proof}

lemma *rel-S-witness'2*:

fixes *L1* :: 'l1 ⇒ 'l1'' ⇒ *bool* **and** *L2* :: 'l2 ⇒ 'l2'' ⇒ *bool*
and *Co1* :: 'co1 ⇒ 'co1' ⇒ *bool* **and** *Co1'* :: 'co1' ⇒ 'co1'' ⇒ *bool*
and *Co2* :: 'co2 ⇒ 'co2' ⇒ *bool* **and** *Co2'* :: 'co2' ⇒ 'co2'' ⇒ *bool*
and *Contra1* :: 'contra1 ⇒ 'contra1' ⇒ *bool* **and** *Contra1'* :: 'contra1' ⇒
'*contra1''* ⇒ *bool*
and *Contra2* :: 'contra2 ⇒ 'contra2' ⇒ *bool* **and** *Contra2'* :: 'contra2' ⇒
'*contra2''* ⇒ *bool*
and *tytok* :: ('l1 × ('l1 × 'l1'') × 'l1'' × 'l2 × ('l2 × 'l2'') × 'l2'' × 'f) *itself*
and *x* :: (-, -, -, -, -, -, 'f) *S*
assumes *rel-S L1 L2 (Co1 OO Co1') (Co2 OO Co2') (Contra1 OO Contra1')*
(*Contra2 OO Contra2'*) *x y*
and *rel-S-neg-distr-cond' Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2'* *tytok*
shows *rel-S (λ(x, y') y. y' = y ∧ L1 x y) (λ(x, y') y. y' = y ∧ L2 x y) Co1'*
Co2' Contra1' Contra2'
(*rel-S-witness' L1 L2 Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2'*
(*x, y*)) *y*
{proof}

lemma *rel-S-neg-distr-imp*:

fixes *Co1* :: 'co1 ⇒ 'co1' ⇒ *bool* **and** *Co1'* :: 'co1' ⇒ 'co1'' ⇒ *bool*
and *Co2* :: 'co2 ⇒ 'co2' ⇒ *bool* **and** *Co2'* :: 'co2' ⇒ 'co2'' ⇒ *bool*
and *Contra1* :: 'contra1 ⇒ 'contra1' ⇒ *bool* **and** *Contra1'* :: 'contra1' ⇒
'*contra1''* ⇒ *bool*
and *Contra2* :: 'contra2 ⇒ 'contra2' ⇒ *bool* **and** *Contra2'* :: 'contra2' ⇒
'*contra2''* ⇒ *bool*
and *tytok-S'* :: ('l1 × ('l1 × 'l1'') × 'l1'' × 'l2 × ('l2 × 'l2'') × 'l2'' × 'f)
itself
and *tytok-S* :: ('l1 × 'l1' × 'l1'' × 'l2 × 'l2' × 'l2'' × 'f) *itself*
assumes *rel-S-neg-distr-cond' Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2*
Contra2' tytok-S'
shows *rel-S-neg-distr-cond Co1 Co1' Co2 Co2' Contra1 Contra1' Contra2 Contra2'* *tytok-S*
{proof}

end

5.3 Subtypes of BNFs without co- and contravariance

If all variables are live, **lift-bnf**'s requirement *P-zip-closed* is equivalent to our closedness under zippings, and Popescu's weaker condition is equivalent to negative subdistributivity restricted to the subset.

bnf-axiomatization $'a\ H$

consts $Q :: 'a\ H \Rightarrow bool$

axiomatization where

$Q\text{-map}: \bigwedge x\ l. Q\ x \Longrightarrow Q\ (\text{map-}H\ l\ x)$

lemma $Q\text{-rel-}H\text{-zipping}$:

fixes $x :: 'a\ H$ **and** $y :: 'c\ H$ **and** $z :: ('a \times 'c)\ H$

assumes $Q\text{-zip}: \bigwedge z :: ('a \times 'c)\ H. \llbracket Q\ (\text{map-}H\ \text{fst}\ z); Q\ (\text{map-}H\ \text{snd}\ z) \rrbracket \Longrightarrow Q\ z$

and $Q\ x$ **and** $Q\ y$ **and** $\text{rel-}H\ L\ x\ y$

and $\text{related}: \text{rel-}H\ (\lambda x\ (x', y). x' = x \wedge L\ x\ y)\ x\ z\ \ \ \ \ \text{rel-}H\ (\lambda(x, y')\ y. y' = y \wedge L\ x\ y)\ z\ y$

shows $Q\ z$

$\langle\text{proof}\rangle$

lemma $Q\text{-zip}$:

fixes $z :: ('a \times 'c)\ H$

assumes $Q\text{-rel-}H\text{-zipping}: \bigwedge(L :: 'a \Rightarrow 'c \Rightarrow -)\ x\ y\ z.$

$\llbracket Q\ x; Q\ y; \text{rel-}H\ L\ x\ y; \text{rel-}H\ (\lambda x\ (x', y). x' = x \wedge L\ x\ y)\ x\ z;$

$\text{rel-}H\ (\lambda(x, y')\ y. y' = y \wedge L\ x\ y)\ z\ y \rrbracket \Longrightarrow Q\ z$

and $Q\ (\text{map-}H\ \text{fst}\ z)$ **and** $Q\ (\text{map-}H\ \text{snd}\ z)$

shows $Q\ z$

$\langle\text{proof}\rangle$

lemma $Q\text{-neg-distr}$:

fixes $x :: 'a\ H$ **and** $y :: 'c\ H$

assumes $Q\text{-zip-weak}: \bigwedge z :: ('a \times 'c)\ H. \llbracket Q\ (\text{map-}H\ \text{fst}\ z); Q\ (\text{map-}H\ \text{snd}\ z) \rrbracket \Longrightarrow$

$\exists z'. Q\ z' \wedge \text{set-}H\ z' \subseteq \text{set-}H\ z \wedge \text{map-}H\ \text{fst}\ z' = \text{map-}H\ \text{fst}\ z \wedge \text{map-}H\ \text{snd}\ z' = \text{map-}H\ \text{snd}\ z$

and $Q\ x$ **and** $Q\ y$ **and** $\text{related}: \text{rel-}H\ (L\ OO\ L')\ x\ y$

shows $(\text{rel-}H\ L\ OO\ \text{eq-onp}\ Q\ OO\ \text{rel-}H\ L')\ x\ y$

$\langle\text{proof}\rangle$

lemma $Q\text{-zip-weak}$:

fixes $z :: ('a \times 'c)\ H$

assumes $Q\text{-neg-distr}: \bigwedge(L :: 'a \Rightarrow ('a \times 'c) \Rightarrow -)\ (L' :: ('a \times 'c) \Rightarrow 'c \Rightarrow bool)\ x\ y.$

$\llbracket Q\ x; Q\ y; \text{rel-}H\ (L\ OO\ L')\ x\ y \rrbracket \Longrightarrow (\text{rel-}H\ L\ OO\ \text{eq-onp}\ Q\ OO\ \text{rel-}H\ L')\ x\ y$

y

and $Q\ (\text{map-}H\ \text{fst}\ z)$ **and** $Q\ (\text{map-}H\ \text{snd}\ z)$

obtains z' **where** $Q\ z'$ **and** $\text{set-}H\ z' \subseteq \text{set-}H\ z$

and $\text{map-H fst } z' = \text{map-H fst } z$ **and** $\text{map-H snd } z' = \text{map-H snd } z$
 ⟨*proof*⟩

end

6 Quotient preservation

theory *Quotient-Preservation* **imports**

Axiomatised-BNF-CC

begin

lemma *G-Quotient*:

fixes $T-l1 :: 'l1 \Rightarrow 'l1' \Rightarrow \text{bool}$ **and** $T-l2 :: 'l2 \Rightarrow 'l2' \Rightarrow \text{bool}$

and $\text{tytok} :: ('l1 \times 'l1' \times 'l1 \times 'l2 \times 'l2' \times 'l2 \times 'f)$ *itself*

assumes *Quotient R-l1 Abs-l1 Rep-l1 T-l1* **and** *Quotient R-l2 Abs-l2 Rep-l2 T-l2*

and *Quotient R-co1 Abs-co1 Rep-co1 T-co1* **and** *Quotient R-co2 Abs-co2*

Rep-co2 T-co2

and *Quotient R-contra1 Abs-contra1 Rep-contra1 T-contra1*

and *Quotient R-contra2 Abs-contra2 Rep-contra2 T-contra2*

and *rel-G-pos-distr-cond T-co1 T-co1⁻¹⁻¹ T-co2 T-co2⁻¹⁻¹ T-contra1 T-contra1⁻¹⁻¹*
T-contra2 T-contra2⁻¹⁻¹

tytok

shows *Quotient (rel-G R-l1 R-l2 R-co1 R-co2 R-contra1 R-contra2)*

(map-G Abs-l1 Abs-l2 Abs-co1 Abs-co2 Rep-contra1 Rep-contra2)

(map-G Rep-l1 Rep-l2 Rep-co1 Rep-co2 Abs-contra1 Abs-contra2)

(rel-G T-l1 T-l2 T-co1 T-co2 T-contra1 T-contra2 :: ($\neg, \neg, \neg, \neg, \neg, \neg, 'f$) $G \Rightarrow \neg$)

⟨*proof*⟩

end

theory *Operation-Examples* **imports**

Composition

Fixpoints

Subtypes

Quotient-Preservation

begin

end

7 Concrete BNF_{CCS}

theory *Concrete-Examples* **imports**

Preliminaries

HOL-Library.Rewrite

HOL-Library.Cardinality

begin

context includes *lifting-syntax*
begin

7.1 Function space

lemma *rel-fun-mono*: $(A \text{====>} B) \leq (A' \text{====>} B')$ **if** $A' \leq A$ $B \leq B'$
 $\langle \text{proof} \rangle$

lemma *rel-fun-eq*: $((=) \text{====>} (=)) = (=)$ $\langle \text{proof} \rangle$

lemma *rel-fun-conversep*: $(A^{-1-1} \text{====>} B^{-1-1}) = (A \text{====>} B)^{-1-1}$ $\langle \text{proof} \rangle$

lemma *map-fun-id0*: $(id \text{---->} id) = id$ $\langle \text{proof} \rangle$

lemma *map-fun-comp*: $(f \text{---->} g) \circ (f' \text{---->} g') = ((f' \circ f) \text{---->} (g \circ g'))$
 $\langle \text{proof} \rangle$

lemma *map-fun-parametric*: $((A \text{====>} A') \text{====>} (B \text{====>} B')) \text{====>} (A' \text{====>} B) \text{====>} (A \text{====>} B')$ $(\text{---->}) (\text{---->})$
 $\langle \text{proof} \rangle$

definition *rel-fun-pos-distr-cond* :: $('a \Rightarrow 'a' \Rightarrow \text{bool}) \Rightarrow ('a' \Rightarrow 'a'' \Rightarrow \text{bool}) \Rightarrow ('b \times 'b' \times 'b'') \text{ itself} \Rightarrow \text{bool}$ **where**
rel-fun-pos-distr-cond $A A' - \longleftrightarrow (\forall (B :: 'b \Rightarrow 'b' \Rightarrow \text{bool}) (B' :: 'b' \Rightarrow 'b'' \Rightarrow \text{bool}) \Rightarrow \text{bool})$.
 $(A \text{====>} B) \text{ OO } (A' \text{====>} B') \leq (A \text{ OO } A') \text{====>} (B \text{ OO } B')$

definition *rel-fun-neg-distr-cond* :: $('a \Rightarrow 'a' \Rightarrow \text{bool}) \Rightarrow ('a' \Rightarrow 'a'' \Rightarrow \text{bool}) \Rightarrow ('b \times 'b' \times 'b'') \text{ itself} \Rightarrow \text{bool}$ **where**
rel-fun-neg-distr-cond $A A' - \longleftrightarrow (\forall (B :: 'b \Rightarrow 'b' \Rightarrow \text{bool}) (B' :: 'b' \Rightarrow 'b'' \Rightarrow \text{bool}) \Rightarrow \text{bool})$.
 $(A \text{ OO } A') \text{====>} (B \text{ OO } B') \leq (A \text{====>} B) \text{ OO } (A' \text{====>} B')$

lemmas

rel-fun-pos-distr = *rel-fun-pos-distr-cond-def*[*THEN iffD1*, *rule-format*] **and**
rel-fun-neg-distr = *rel-fun-neg-distr-cond-def*[*THEN iffD1*, *rule-format*]

lemma *rel-fun-pos-distr-iff* [*simp*]: *rel-fun-pos-distr-cond* $A A'$ *tytok* = *True*
 $\langle \text{proof} \rangle$

lemma *rel-fun-neg-distr-imp*: $\llbracket \text{left-unique } A; \text{right-total } A; \text{right-unique } A'; \text{left-total } A' \rrbracket \Longrightarrow$
rel-fun-neg-distr-cond $A A'$ *tytok*
 $\langle \text{proof} \rangle$

lemma *rel-fun-pos-distr-cond-eq*: *rel-fun-pos-distr-cond* $(=) (=)$ *tytok*
 $\langle \text{proof} \rangle$

lemma *rel-fun-neg-distr-cond-eq*: *rel-fun-neg-distr-cond* $(=) (=)$ *tytok*

<proof>

thm *fun.set-map fun.map-cong0 fun.rel-mono-strong*

7.2 Covariant powerset

lemma *rel-set-mono*: $A \leq A' \implies \text{rel-set } A \leq \text{rel-set } A'$ *<proof>*

lemma *rel-set-eq*: $\text{rel-set } (=) = (=)$ *<proof>*

lemma *rel-set-conversep*: $\text{rel-set } A^{-1-1} = (\text{rel-set } A)^{-1-1}$ *<proof>*

lemma *map-set-id0*: $\text{image id} = \text{id}$ *<proof>*

lemma *map-set-comp*: $\text{image } f \circ \text{image } g = \text{image } (f \circ g)$ *<proof>*

lemma *map-set-parametric*: **includes** *lifting-syntax* **shows**
 $((A \text{====>} B) \text{====>} \text{rel-set } A \text{====>} \text{rel-set } B)$ *image image*
<proof>

definition *rel-set-pos-distr-cond* :: $('a \Rightarrow 'a' \Rightarrow \text{bool}) \Rightarrow ('a' \Rightarrow 'a'' \Rightarrow \text{bool}) \Rightarrow \text{bool}$ **where**

rel-set-pos-distr-cond $A A' \longleftrightarrow \text{rel-set } A \text{ OO rel-set } A' \leq \text{rel-set } (A \text{ OO } A')$

definition *rel-set-neg-distr-cond* :: $('a \Rightarrow 'a' \Rightarrow \text{bool}) \Rightarrow ('a' \Rightarrow 'a'' \Rightarrow \text{bool}) \Rightarrow \text{bool}$ **where**

rel-set-neg-distr-cond $A A' \longleftrightarrow \text{rel-set } (A \text{ OO } A') \leq \text{rel-set } A \text{ OO rel-set } A'$

lemmas

rel-set-pos-distr = *rel-set-pos-distr-cond-def*[*THEN iffD1*, *rule-format*] **and**
rel-set-neg-distr = *rel-set-neg-distr-cond-def*[*THEN iffD1*, *rule-format*]

lemma *rel-set-pos-distr-iff* [*simp*]: $\text{rel-set-pos-distr-cond } A A' = \text{True}$
<proof>

lemma *rel-set-neg-distr-iff* [*simp*]: $\text{rel-set-neg-distr-cond } A A' = \text{True}$
<proof>

lemma *rel-set-pos-distr-eq*: $\text{rel-set-pos-distr-cond } (=) (=)$
<proof>

lemma *rel-set-neg-distr-eq*: $\text{rel-set-neg-distr-cond } (=) (=)$
<proof>

7.3 Bounded sets

We define bounded sets as a subtype, with an additional fixed parameter which controls the bound. Using the BNF_{CC} structure on the covariant powerset functor, it suffices to show the preconditions for the closedness of

BNF_{CC} under subtypes.

typedef ('a, 'k) bset = {A :: 'a set. finite A ∧ card A ≤ CARD('k)}
⟨proof⟩

setup-lifting type-definition-bset

lemma bset-map-closed:
fixes f A
defines B ≡ image f A
assumes finite A ∧ card A ≤ CARD('k)
shows finite B ∧ card B ≤ CARD('k)
⟨proof⟩

lift-definition map-bset :: ('a ⇒ 'b) ⇒ ('a, 'k) bset ⇒ ('b, 'k) bset **is** image
⟨proof⟩

lift-definition rel-bset :: ('a ⇒ 'b ⇒ bool) ⇒ ('a, 'k) bset ⇒ ('b, 'k) bset ⇒ bool
is rel-set ⟨proof⟩

definition neg-distr-cond-bset :: ('a ⇒ 'b ⇒ bool) ⇒ ('b ⇒ 'c ⇒ bool) ⇒ 'k itself
⇒ bool **where**
neg-distr-cond-bset C C' - ↔ rel-bset (C OO C') ≤ rel-bset C OO (rel-bset C')
:: (-, 'k) bset ⇒ -)

lemma right-unique-rel-set-lemma:
assumes right-unique R **and** rel-set R X Y
obtains f **where** Y = image f X **and** ∀ x∈X. R x (f x)
⟨proof⟩

lemma left-unique-rel-set-lemma:
assumes left-unique R **and** rel-set R Y X
obtains f **where** Y = image f X **and** ∀ x∈X. R (f x) x
⟨proof⟩

lemma neg-distr-cond-bset-right-unique:
right-unique C ⇒ neg-distr-cond-bset C D tytok
⟨proof⟩

lemma neg-distr-cond-bset-left-unique:
left-unique D ⇒ neg-distr-cond-bset C D tytok
⟨proof⟩

lemma neg-distr-cond-bset-eq: neg-distr-cond-bset (=) (=) tytok
⟨proof⟩

7.4 Contravariant powerset (sets as predicates)

type-synonym 'a pred = 'a ⇒ bool

definition *map-pred* :: ('b ⇒ 'a) ⇒ 'a pred ⇒ 'b pred **where**
map-pred f = (f ----> id)

definition *rel-pred* :: ('a ⇒ 'b ⇒ bool) ⇒ 'a pred ⇒ 'b pred ⇒ bool **where**
rel-pred R = (R ===> (↔))

lemma *rel-pred-mono*: $A' \leq A \implies \text{rel-pred } A \leq \text{rel-pred } A'$ *<proof>*

lemma *rel-pred-eq*: *rel-pred* (=) = (=)
<proof>

lemma *rel-pred-conversep*: *rel-pred* $A^{-1-1} = (\text{rel-pred } A)^{-1-1}$
<proof>

lemma *map-pred-id0*: *map-pred* id = id
<proof>

lemma *map-pred-comp*: *map-pred* f ◦ *map-pred* g = *map-pred* (g ◦ f)
<proof>

lemma *map-pred-parametric*: ((A' ===> A) ===> *rel-pred* A ===> *rel-pred* A') *map-pred* *map-pred*
<proof>

definition *rel-pred-pos-distr-cond* :: ('a ⇒ 'a' ⇒ bool) ⇒ ('a' ⇒ 'a'' ⇒ bool) ⇒ bool **where**
rel-pred-pos-distr-cond A B ↔ *rel-pred* A OO *rel-pred* B ≤ *rel-pred* (A OO B)

definition *rel-pred-neg-distr-cond* :: ('a ⇒ 'a' ⇒ bool) ⇒ ('a' ⇒ 'a'' ⇒ bool) ⇒ bool **where**
rel-pred-neg-distr-cond A B ↔ *rel-pred* (A OO B) ≤ *rel-pred* A OO *rel-pred* B

lemmas

rel-pred-pos-distr = *rel-pred-pos-distr-cond-def*[*THEN iffD1*, *rule-format*] **and**
rel-pred-neg-distr = *rel-pred-neg-distr-cond-def*[*THEN iffD1*, *rule-format*]

lemma *rel-pred-pos-distr-iff* [*simp*]: *rel-pred-pos-distr-cond* A B = True
<proof>

lemma *rel-pred-pos-distr-cond-eq*: *rel-pred-pos-distr-cond* (=) (=)
<proof>

lemma *neg-fun-distr3*:

assumes 1: *left-unique* R *right-total* R

and 2: *right-unique* S *left-total* S

shows *rel-fun* (R OO R') (S OO S') ≤ *rel-fun* R S OO *rel-fun* R' S'

<proof>

As there are no live variables, we can get a weaker condition than if we

derived it from (\implies)'s condition!

lemma *rel-pred-neg-distr-imp*:

right-unique B \wedge left-total B \vee left-unique A \wedge right-total A \implies rel-pred-neg-distr-cond A B

<proof>

lemma *rel-pred-neg-distr-cond-eq*: *rel-pred-neg-distr-cond (=) (=)*

<proof>

lemma *left-unique-rel-pred*: *left-total A \implies left-unique (rel-pred A)*

<proof>

lemma *right-unique-rel-pred*: *right-total A \implies right-unique (rel-pred A)*

<proof>

lemma *left-total-rel-pred*: *left-unique A \implies left-total (rel-pred A)*

<proof>

lemma *right-total-rel-pred*: *right-unique A \implies right-total (rel-pred A)*

<proof>

end

7.5 Filter

Similarly to bounded sets, we exploit the definition of filters as a subtype in order to lift the BNF_{CC} operations. Here we use that the *is-filter* predicate is closed under zippings.

lemma *map-filter-closed*:

includes *lifting-syntax*

assumes *is-filter F*

shows *is-filter (((f \dashrightarrow id) \dashrightarrow id) F)*

<proof>

definition *rel-pred2-neg-distr-cond* :: *('a \Rightarrow 'a' \Rightarrow bool) \Rightarrow ('a' \Rightarrow 'a'' \Rightarrow bool) \Rightarrow bool* **where**

rel-pred2-neg-distr-cond A B \longleftrightarrow

rel-pred (rel-pred (A OO B)) \leq rel-pred (rel-pred A) OO rel-pred (rel-pred B)

consts *rel-pred2-witness* :: *('a \Rightarrow 'a' \Rightarrow bool) \Rightarrow ('a' \Rightarrow 'a'' \Rightarrow bool) \Rightarrow*

((('a \Rightarrow bool) \Rightarrow bool) \times (('a'' \Rightarrow bool) \Rightarrow bool) \Rightarrow ('a' \Rightarrow bool) \Rightarrow bool

specification (*rel-pred2-witness*)

rel-pred2-witness1: $\bigwedge K K' x y. \llbracket \text{rel-pred2-neg-distr-cond } K K'; \text{rel-pred (rel-pred (K OO K')) } x y \rrbracket \implies$

rel-pred (rel-pred K) x (rel-pred2-witness K K' (x, y))

rel-pred2-witness2: $\bigwedge K K' x y. \llbracket \text{rel-pred2-neg-distr-cond } K K'; \text{rel-pred (rel-pred$

$(K \text{ OO } K') \ x \ y \] \implies$
 $\text{rel-pred } (\text{rel-pred } K') \ (\text{rel-pred2-witness } K \ K' \ (x, y)) \ y$
 $\langle \text{proof} \rangle$

lemmas $\text{rel-pred2-witness} = \text{rel-pred2-witness1 } \text{rel-pred2-witness2}$

context includes *lifting-syntax*
begin

definition $\text{rel-filter-neg-distr-cond}' :: ('a \Rightarrow 'b \Rightarrow \text{bool}) \Rightarrow ('b \Rightarrow 'c \Rightarrow \text{bool}) \Rightarrow \text{bool}$
where

$\text{rel-filter-neg-distr-cond}' \ C \ C' \longleftrightarrow \text{left-total } C \wedge \text{right-unique } C \vee \text{right-total } C'$
 $\wedge \text{left-unique } C'$

lemma $\text{rel-filter-neg-distr-cond}'\text{-stronger}$:

assumes $\text{rel-filter-neg-distr-cond}' \ C \ C'$

shows $\text{rel-pred2-neg-distr-cond } C \ C'$

$\langle \text{proof} \rangle$

lemma $\text{rel-filter-neg-distr-cond}'\text{-eq}$: $\text{rel-filter-neg-distr-cond}' \ (=) \ (=)$

$\langle \text{proof} \rangle$

lemma $\text{is-filter-rel-witness}$:

assumes F : $\text{is-filter } F$ **and** G : $\text{is-filter } G$

and FG : $\text{rel-pred } (\text{rel-pred } (C \text{ OO } C')) \ F \ G$

and cond : $\text{rel-filter-neg-distr-cond}' \ C \ C'$

shows $\text{is-filter } (\text{rel-pred2-witness } C \ C' \ (F, G))$

$\langle \text{proof} \rangle$

end

The following example shows that filters do not satisfy **lift-bnf**'s condition.

experiment begin

unbundle *lifting-syntax*

definition $\text{raw-filtermap } f = ((f \text{ ----} > \text{id}) \text{ ----} > \text{id})$

lemma $\text{raw-filtermap-apply}$: $\text{raw-filtermap } f \ F = (\lambda P. F \ (\lambda x. P \ (f \ x)))$

$\langle \text{proof} \rangle$

lemma $\text{filtermap } f = \text{Abs-filter} \circ \text{raw-filtermap } f \circ \text{Rep-filter}$

$\langle \text{proof} \rangle$

definition Z **where**

$Z = \{ \{(False, False), (False, True)\}, \{(False, False), (True, False)\},$
 $\{(False, False), (False, True), (True, False), (True, True)\} \}$

abbreviation $Z' \equiv (\lambda P. \text{Collect } P \in Z)$

lemma *is-filter* (raw-filtermap fst Z')
⟨proof⟩

lemma *is-filter* (raw-filtermap snd Z')
⟨proof⟩

lemma \neg *is-filter* Z'
⟨proof⟩

end

7.6 Almost-everywhere equal sequences

inductive *aeseq-eq* :: (nat \Rightarrow 'a) \Rightarrow (nat \Rightarrow 'a) \Rightarrow bool **for** *f g* **where**
aeseq-eq f g **if** *finite* {n. f n \neq g n}

lemma *equivp-aeseq-eq*: *equivp aeseq-eq*
⟨proof⟩

quotient-type 'a *aeseq* = nat \Rightarrow 'a / *aeseq-eq* ⟨proof⟩

lift-definition *map-aeseq* :: ('a \Rightarrow 'b) \Rightarrow 'a *aeseq* \Rightarrow 'b *aeseq* **is** (○)
⟨proof⟩

lemma *map-aeseq-id*: *map-aeseq id x* = *x*
⟨proof⟩

lemma *map-aeseq-comp*: *map-aeseq f (map-aeseq g x)* = *map-aeseq (f ○ g) x*
⟨proof⟩

lift-definition *rel-aeseq* :: ('a \Rightarrow 'b \Rightarrow bool) \Rightarrow 'a *aeseq* \Rightarrow 'b *aeseq* \Rightarrow bool **is**
 $\lambda R f g. \text{finite } \{n. \neg R (f n) (g n)\}$
⟨proof⟩

lemma *rel-aeseq-mono*: $R \leq S \implies \text{rel-aeseq } R \leq \text{rel-aeseq } S$
⟨proof⟩

lemma *rel-aeseq-eq*: *rel-aeseq* (=) = (=)
⟨proof⟩

lemma *rel-aeseq-conversep*: *rel-aeseq* R^{-1-1} = (*rel-aeseq* *R*)⁻¹⁻¹
⟨proof⟩

lemma *map-aeseq-parametric*: **includes** *lifting-syntax* **shows**
((*A* ==== *B*) \implies *rel-aeseq A* ==== *rel-aeseq B*) *map-aeseq map-aeseq*
⟨proof⟩

lemma *rel-aeseq-distr*: *rel-aeseq (R OO S)* = *rel-aeseq R OO rel-aeseq S*
⟨proof⟩

end

8 Example: deterministic discrete system

```
theory DDS imports
  Concrete-Examples
  HOL-Library.Rewrite
  HOL-Library.FSet
begin
```

```
unbundle lifting-syntax
```

8.1 Definition and generalised mapper and relator

```
codatatype ('a, 'b) dds = DDS (run: 'a  $\Rightarrow$  'b  $\times$  ('a, 'b) dds)
  for map: map-dds'
      rel: rel-dds'
```

```
primcorec map-dds :: ('a'  $\Rightarrow$  'a)  $\Rightarrow$  ('b  $\Rightarrow$  'b')  $\Rightarrow$  ('a, 'b) dds  $\Rightarrow$  ('a', 'b') dds
where
```

```
  run (map-dds f g S) = ( $\lambda a$ . map-prod g (map-dds f g) (run S (f a)))
```

```
lemma map-dds-id: map-dds id id S = S
  <proof>
```

```
lemma map-dds-comp: map-dds f g (map-dds f' g' S) = map-dds (f'  $\circ$  f) (g  $\circ$  g') S
  <proof>
```

```
coinductive rel-dds :: ('a  $\Rightarrow$  'a'  $\Rightarrow$  bool)  $\Rightarrow$  ('b  $\Rightarrow$  'b'  $\Rightarrow$  bool)  $\Rightarrow$  ('a, 'b) dds  $\Rightarrow$ 
  ('a', 'b') dds  $\Rightarrow$  bool
  for A B where
    rel-dds A B S S' if rel-fun A (rel-prod B (rel-dds A B)) (run S) (run S')
```

```
lemma rel-dds'-rel-dds: rel-dds' B = rel-dds (=) B
  <proof>
```

```
lemma rel-dds-eq [relator-eq]: rel-dds (=) (=) = (=)
  <proof>
```

```
lemma rel-dds-mono [relator-mono]: rel-dds A B  $\leq$  rel-dds A' B' if A'  $\leq$  A B
   $\leq$  B'
  <proof>
```

```
lemma rel-dds-conversep: rel-dds A-1-1 B-1-1 = (rel-dds A B)-1-1
  <proof>
```

```
lemma DDS-parametric [transfer-rule]:
```

$((A \text{====>} \text{rel-prod } B \text{ (rel-dds } A \text{ } B)) \text{====>} \text{rel-dds } A \text{ } B) \text{ DDS DDS}$
 $\langle \text{proof} \rangle$

lemma *run-parametric* [*transfer-rule*]:
 $(\text{rel-dds } A \text{ } B \text{====>} A \text{====>} \text{rel-prod } B \text{ (rel-dds } A \text{ } B)) \text{ run run}$
 $\langle \text{proof} \rangle$

lemma *corec-dds-parametric* [*transfer-rule*]:
 $((S \text{====>} A \text{====>} \text{rel-prod } B \text{ (rel-sum (rel-dds } A \text{ } B) S)) \text{====>} S \text{====>} \text{rel-dds } A \text{ } B) \text{ corec-dds corec-dds}$
 $\langle \text{proof} \rangle$

lemma *map-dds-parametric* [*transfer-rule*]:
 $((A' \text{====>} A) \text{====>} (B \text{====>} B')) \text{====>} \text{rel-dds } A \text{ } B \text{====>} \text{rel-dds } A' \text{ } B'$
 map-dds map-dds
 $\langle \text{proof} \rangle$

lemmas *map-dds-rel-cong* = *map-dds-parametric*[*unfolded rel-fun-def*, *rule-format*, *rotated -1*]

lemma *rel-dds-Grp*:
 $\text{rel-dds (Grp UNIV } f)^{-1-1} \text{ (Grp UNIV } g) = \text{Grp UNIV (map-dds } f \text{ } g)$
 $\langle \text{proof} \rangle$

lemma *rel-dds-pos-distr* [*relator-distr*]:
 $\text{rel-dds } A \text{ } B \text{ OO rel-dds } C \text{ } D \leq \text{rel-dds (} A \text{ OO } C) \text{ (} B \text{ OO } D)$
 $\langle \text{proof} \rangle$

lemma *Quotient-dds* [*quot-map*]:
assumes *Quotient* *R1* *Abs1* *Rep1* *T1* **and** *Quotient* *R2* *Abs2* *Rep2* *T2*
shows *Quotient* $(\text{rel-dds } R1 \text{ } R2) \text{ (map-dds } Rep1 \text{ } Abs2) \text{ (map-dds } Abs1 \text{ } Rep2)$
 $(\text{rel-dds } T1 \text{ } T2)$
 $\langle \text{proof} \rangle$

This is just the co-iterator.

primcorec *dds-of* :: $(\text{'s} \Rightarrow \text{'a} \Rightarrow (\text{'b} \times \text{'s})) \Rightarrow \text{'s} \Rightarrow (\text{'a}, \text{'b}) \text{ dds}$ **where**
 $\text{run (dds-of } f \text{ } s) = \text{map-prod id (dds-of } f) \circ f \text{ } s$

lemma *dds-of-parametric* [*transfer-rule*]:
 $((S \text{====>} A \text{====>} \text{rel-prod } B \text{ } S) \text{====>} S \text{====>} \text{rel-dds } A \text{ } B) \text{ dds-of dds-of}$
 $\langle \text{proof} \rangle$

8.2 Evenness of partial sums

definition *even-psum* :: $(\text{int}, \text{bool}) \text{ dds}$ **where**
 $\text{even-psum} = \text{dds-of } (\lambda \text{psum } n. (\text{even (psum} + n), \text{psum} + n)) \text{ } 0$

definition *even-psum-nat* :: $(\text{nat}, \text{bool}) \text{ dds}$ **where**
 $\text{even-psum-nat} = \text{map-dds int id even-psum}$

8.3 Composition

primcorec *compose* :: ('a, 'b) dds ⇒ ('b, 'c) dds ⇒ ('a, 'c) dds (**infixl** · 120)
where
run (S1 · S2) = (λa. let (b, S1') = run S1 a; (c, S2') = run S2 b in (c, S1' · S2'))

lemma *compose-parametric* [*transfer-rule*]:
 (rel-dds A B ===> rel-dds B C ===> rel-dds A C) (·) (·)
 ⟨*proof*⟩

For the following lemma, a direct proof by induction is easy as the inner functor of the *dds* codatatype is fairly simple.

lemma *map-dds f g S1 · S2 = map-dds f id (S1 · map-dds g id S2)*
 ⟨*proof*⟩

However, we can also follow the systematic route via parametricity:

lemma *compose-map1*: *map-dds f g S1 · S2 = map-dds f id (S1 · map-dds g id S2)*
for S1 :: ('a, 'b) dds **and** S2 :: ('b, 'c) dds
 ⟨*proof*⟩

lemma *compose-map2*: *S1 · map-dds f g S2 = map-dds id g (map-dds id f S1 · S2)*
for S1 :: ('a, 'b) dds **and** S2 :: ('b, 'c) dds
 ⟨*proof*⟩

primcorec *parallel* :: ('a, 'b) dds ⇒ ('c, 'd) dds ⇒ ('a + 'c, 'b + 'd) dds (**infixr** || 130) **where**
run (S1 || S2) = (λx. case x of
 | Inl a ⇒ let (b, S1') = run S1 a in (Inl b, S1' || S2)
 | Inr c ⇒ let (d, S2') = run S2 c in (Inr d, S1 || S2'))

lemma *parallel-parametric* [*transfer-rule*]:
 (rel-dds A B ===> rel-dds C D ===> rel-dds (rel-sum A C) (rel-sum B D))
 (||) (||)
 ⟨*proof*⟩

lemma *map-parallel*:
map-dds f h S1 || map-dds g k S2 = map-dds (map-sum f g) (map-sum h k) (S1 || S2)
 ⟨*proof*⟩

8.4 Graph traversal: refinement and quotients

lemma *finite-Image*:
finite A ⇒ finite (R “ A) ↔ (∀ x∈A. finite {y. (x, y) ∈ R})
 ⟨*proof*⟩

context includes *fset.lifting* **begin**
lift-definition *fImage* :: ('a × 'b) fset ⇒ 'a fset ⇒ 'b fset **is Image parametric**
Image-parametric
 ⟨*proof*⟩

lemmas *fImage-iff* = *Image-iff*[*Transfer.transferred*]
lemmas *fImageI* [*intro*] = *ImageI*[*Transfer.transferred*]
lemmas *fImageE* [*elim!*] = *ImageE*[*Transfer.transferred*]
lemmas *rev-fImageI* = *rev-ImageI*[*Transfer.transferred*]
lemmas *fImage-mono* = *Image-mono*[*Transfer.transferred*]

lifting-update *fset.lifting*
lifting-forget *fset.lifting*
end

type-synonym 'a graph = ('a × 'a) fset

definition *traverse* :: 'a graph ⇒ ('a fset, 'a fset) dds **where**
traverse E = dds-of (λ*visited* A. ((*fImage* E A) |-| *visited*, *visited* |∪| A)) {||}

type-synonym 'a graph' = ('a × 'a) list

definition *traverse-impl* :: 'a graph' ⇒ ('a list, 'a list) dds **where**
traverse-impl E =
 dds-of (λ*visited* A. (map snd [(*x*, *y*) ← E . *x* ∈ set A ∧ *y* ∉ |*visited*],
visited |∪| *fset-of-list* A)) {||}

definition *list-fset-rel* :: 'a list ⇒ 'a fset ⇒ bool **where**
list-fset-rel *xs* A ↔ *fset-of-list* *xs* = A

lemma *traverse-refinement*: — This is the refinement lemma.
 (*list-fset-rel* ==> *rel-dds* *list-fset-rel* *list-fset-rel*) *traverse-impl* *traverse*
 ⟨*proof*⟩

lemma *fset-of-list-parametric* [*transfer-rule*]:
 (*list-all2* A ==> *rel-fset* A) *fset-of-list* *fset-of-list*
including *fset.lifting* ⟨*proof*⟩

lemma *traverse-impl-parametric* [*transfer-rule*]:
assumes [*transfer-rule*]: *bi-unique* A
shows (*list-all2* (*rel-prod* A A) ==> *rel-dds* (*list-all2* A) (*list-all2* A)) *traverse-impl* *traverse-impl*
 ⟨*proof*⟩

By constructing finite sets as a quotient of lists, we can synthesise an abstract version of *traverse-impl* automatically, together with a polymorphic refinement lemma.

quotient-type 'a fset' = 'a list / *vimage2p* set set (=)
 ⟨*proof*⟩

lift-definition $traverse'' :: ('a \times 'a) fset' \Rightarrow ('a fset', 'a fset') dds$
is $traverse-impl :: 'a graph' \Rightarrow -$ **parametric** $traverse-impl-parametric$
 $\langle proof \rangle$

8.5 Generalised rewriting

definition $accumulate :: ('a fset, 'a fset) dds$ **where**
 $accumulate = dds-of (\lambda A X. (A \mid\cup\mid X, A \mid\cup\mid X)) \{\mid\mid\}$

lemma $accumulate-mono: rel-dds (\mid\subseteq\mid) (\mid\subseteq\mid) accumulate accumulate$
 $\langle proof \rangle$

lemma $traverse-mono: ((\mid\subseteq\mid) ==> rel-dds (=) (\mid\subseteq\mid)) traverse traverse$
 $\langle proof \rangle$

lemma
assumes $G \mid\subseteq\mid H$
shows $rel-dds (=) (\mid\subseteq\mid) (traverse G \cdot accumulate) (traverse H \cdot accumulate)$
 $\langle proof \rangle$

definition $seen :: ('a fset, 'a fset) dds$ **where**
 $seen = dds-of (\lambda S X. (S \mid\cap\mid X, S \mid\cup\mid X)) \{\mid\mid\}$

lemma $seen-mono: rel-dds (\mid\subseteq\mid) (\mid\subseteq\mid) seen seen$
 $\langle proof \rangle$

lemma
assumes $G \mid\subseteq\mid H$
shows $rel-dds (=) (\mid\subseteq\mid) (traverse G \cdot seen) (traverse H \cdot seen)$
 $\langle proof \rangle$

end

References

- [1] J. C. Blanchette, A. Popescu, and D. Traytel. Operations on bounded natural functors. *Archive of Formal Proofs*, Dec. 2017. http://isa-afp.org/entries/BNF_Operations.html, Formal proof development.
- [2] A. Lochbihler and J. Schneider. Relational parametricity and quotient preservation for modular (co)datatypes. In J. Avigad and A. Mahboubi, editors, *Interactive Theorem Proving (ITP 2018)*, LNCS. Springer, 2018.