Transport via Partial Galois Connections and Equivalences

Kevin Kappelmann

October 13, 2023

Abstract

This entry contains the accompanying formalisation of the paper "Transport via Partial Galois Connections and Equivalences" (APLAS 2023) [2]. It contains a theoretical framework to transport programs via equivalences, subsuming the theory of Isabelle's Lifting package [1]. It also contains a prototype to automate transports using this framework in Isabelle/HOL, but this prototype is not yet ready for production. Finally, it contains a library on top of Isabelle/HOL's axioms, including various relativised concepts on orders, functions, binary relations, and Galois connections and equivalences.

Contents

Chapter 1

HOL-Basics

theory *HOL-Basics-Base*
  imports
    HOL.HOL
begin

end

1.1 Binary Relations

1.1.1 Basic Functions

theory *Binary-Relation-Functions*
  imports
    HOL-Basics-Base
begin

Summary Basic functions on binary relations.

definition *rel-comp R S x y* ≡ ∃ *z*. *R x z* ∧ *S z y*

bundle *rel-comp-syntax* **begin notation** *rel-comp* (**infixl** ∘∘ *55*) **end**
bundle *no-rel-comp-syntax* **begin no-notation** *rel-comp* (**infixl** ∘∘ *55*) **end**
unbundle *rel-comp-syntax*

lemma *rel-compI* [*intro*]:
  assumes *R x y*
  and *S y z*
  shows (*R* ∘∘ *S*) *x z*
  ⟨*proof*⟩

lemma *rel-compE* [*elim*]:
  assumes (*R* ∘∘ *S*) *x y*
  obtains *z* **where** *R x z S z y*
  ⟨*proof*⟩

lemma *rel-comp-assoc*: $R \circ\circ (S \circ\circ T) = (R \circ\circ S) \circ\circ T$
⟨*proof*⟩

definition *rel-inv* :: $('a \Rightarrow 'b \Rightarrow bool) \Rightarrow 'b \Rightarrow 'a \Rightarrow bool$
where *rel-inv* $R\ x\ y \equiv R\ y\ x$

bundle *rel-inv-syntax* **begin notation** *rel-inv* $((\text{-}^{-1})\ [1000])$ **end**
bundle *no-rel-inv-syntax* **begin no-notation** *rel-inv* $((\text{-}^{-1})\ [1000])$ **end**
unbundle *rel-inv-syntax*

lemma *rel-invI* [*intro*]:
assumes $R\ x\ y$
shows $R^{-1}\ y\ x$
⟨*proof*⟩

lemma *rel-invD* [*dest*]:
assumes $R^{-1}\ x\ y$
shows $R\ y\ x$
⟨*proof*⟩

lemma *rel-inv-iff-rel* [*simp*]: $R^{-1}\ x\ y \longleftrightarrow R\ y\ x$
⟨*proof*⟩

lemma *rel-inv-comp-eq* [*simp*]: $(R \circ\circ S)^{-1} = S^{-1} \circ\circ R^{-1}$
⟨*proof*⟩

lemma *rel-inv-inv-eq-self* [*simp*]: $R^{-1-1} = R$
⟨*proof*⟩

lemma *rel-inv-eq-iff-eq* [*iff*]: $R^{-1} = S^{-1} \longleftrightarrow R = S$
⟨*proof*⟩

definition *in-dom* $R\ x \equiv \exists\, y.\ R\ x\ y$

lemma *in-domI* [*intro*]:
assumes $R\ x\ y$
shows *in-dom* $R\ x$
⟨*proof*⟩

lemma *in-domE* [*elim*]:
assumes *in-dom* $R\ x$
obtains y **where** $R\ x\ y$
⟨*proof*⟩

lemma *in-dom-if-in-dom-rel-comp*:
assumes *in-dom* $(R \circ\circ S)\ x$
shows *in-dom* $R\ x$
⟨*proof*⟩

definition *in-codom* $R\ y \equiv \exists x.\ R\ x\ y$

lemma *in-codomI* [*intro*]:
assumes $R\ x\ y$
shows *in-codom* $R\ y$
⟨*proof*⟩

lemma *in-codomE* [*elim*]:
assumes *in-codom* $R\ y$
obtains x **where** $R\ x\ y$
⟨*proof*⟩

lemma *in-codom-if-in-codom-rel-comp*:
assumes *in-codom* $(R \circ\circ S)\ y$
shows *in-codom* $S\ y$
⟨*proof*⟩

lemma *in-codom-rel-inv-eq-in-dom* [*simp*]: *in-codom* (R^{-1}) = *in-dom* R
⟨*proof*⟩

lemma *in-dom-rel-inv-eq-in-codom* [*simp*]: *in-dom* (R^{-1}) = *in-codom* R
⟨*proof*⟩

definition *in-field* $R\ x \equiv$ *in-dom* $R\ x \vee$ *in-codom* $R\ x$

lemma *in-field-if-in-dom*:
assumes *in-dom* $R\ x$
shows *in-field* $R\ x$
⟨*proof*⟩

lemma *in-field-if-in-codom*:
assumes *in-codom* $R\ x$
shows *in-field* $R\ x$
⟨*proof*⟩

lemma *in-fieldE* [*elim*]:
assumes *in-field* $R\ x$
obtains (*in-dom*) x' **where** $R\ x\ x'$ | (*in-codom*) x' **where** $R\ x'\ x$
⟨*proof*⟩

lemma *in-fieldE′*:
assumes *in-field* $R\ x$
obtains (*in-dom*) *in-dom* $R\ x$ | (*in-codom*) *in-codom* $R\ x$
⟨*proof*⟩

lemma *in-fieldI* [*intro*]:
assumes $R\ x\ y$
shows *in-field* $R\ x$ *in-field* $R\ y$
⟨*proof*⟩

lemma *in-field-iff-in-dom-or-in-codom*:
  in-field L x $\longleftrightarrow$ *in-dom* L x $\vee$ *in-codom* L x
  ⟨*proof*⟩

lemma *in-field-rel-inv-eq* [*simp*]: *in-field* R^{-1} = *in-field* R
  ⟨*proof*⟩

lemma *in-field-compE* [*elim*]:
  assumes *in-field* (R ∘∘ S) x
  obtains (*in-dom*) *in-dom* R x | (*in-codom*) *in-codom* S x
  ⟨*proof*⟩

lemma *in-field-eq-in-dom-if-in-codom-eq-in-dom*:
  assumes *in-codom* R = *in-dom* R
  shows *in-field* R = *in-dom* R
  ⟨*proof*⟩

definition *rel-if* B R x y $\equiv$ B $\longrightarrow$ R x y

bundle *rel-if-syntax* **begin notation** (**output**) *rel-if* (**infixl** $\longrightarrow$ *50*) **end**
bundle *no-rel-if-syntax* **begin no-notation** (**output**) *rel-if* (**infixl** $\longrightarrow$ *50*) **end**
unbundle *rel-if-syntax*

lemma *rel-if-eq-rel-if-pred* [*simp*]:
  assumes B
  shows (*rel-if* B R) = R
  ⟨*proof*⟩

lemma *rel-if-eq-top-if-not-pred* [*simp*]:
  assumes $\neg B$
  shows (*rel-if* B R) = (λ- -. *True*)
  ⟨*proof*⟩

lemma *rel-if-if-impI* [*intro*]:
  assumes $B \Longrightarrow R$ x y
  shows (*rel-if* B R) x y
  ⟨*proof*⟩

lemma *rel-ifE* [*elim*]:
  assumes (*rel-if* B R) x y
  obtains $\neg B$ | B R x y
  ⟨*proof*⟩

lemma *rel-ifD*:
  assumes (*rel-if* B R) x y
  and B
  shows R x y
  ⟨*proof*⟩

consts *restrict-left* :: $('a \Rightarrow 'b \Rightarrow bool) \Rightarrow 'c \Rightarrow 'a \Rightarrow 'b \Rightarrow bool$

definition *restrict-right* $R\ P \equiv (restrict\text{-}left\ R^{-1}\ P)^{-1}$

overloading
  restrict-left-pred $\equiv$ *restrict-left* :: $('a \Rightarrow 'b \Rightarrow bool) \Rightarrow ('a \Rightarrow bool) \Rightarrow 'a \Rightarrow 'b \Rightarrow bool$
begin
  definition *restrict-left-pred* $R\ P\ x\ y \equiv P\ x \wedge R\ x\ y$
end

bundle *restrict-syntax*
begin
notation *restrict-left* ((-)↾(-) [*1000*])
notation *restrict-right* ((-)↿(-) [*1000*])
end
bundle *no-restrict-syntax*
begin
no-notation *restrict-left* ((-)↾(-) [*1000*])
no-notation *restrict-right* ((-)↿(-) [*1000*])
end
unbundle *restrict-syntax*

lemma *restrict-leftI* [*intro*]:
  assumes $R\ x\ y$
  and $P\ x$
  shows $R{\restriction}_P\ x\ y$
  ⟨*proof*⟩

lemma *restrict-leftE* [*elim*]:
  assumes $R{\restriction}_P\ x\ y$
  obtains $P\ x\ R\ x\ y$
  ⟨*proof*⟩

lemma *restrict-right-eq*: $R{\upharpoonleft}_P = ((R^{-1}){\restriction}_P)^{-1}$
  ⟨*proof*⟩

lemma *rel-inv-restrict-right-rel-inv-eq-restrict-left* [*simp*]: $((R^{-1}){\upharpoonleft}_P)^{-1} = R{\restriction}_P$
  ⟨*proof*⟩

lemma *restrict-right-iff-restrict-left*: $R{\upharpoonleft}_P\ x\ y = (R^{-1}){\restriction}_P\ y\ x$
  ⟨*proof*⟩

lemma *restrict-rightI* [*intro*]:
  assumes $R\ x\ y$
  and $P\ y$
  shows $R{\upharpoonleft}_P\ x\ y$
  ⟨*proof*⟩

lemma *restrict-rightE* [*elim*]:
  assumes $R{\upharpoonleft}_{P}\ x\ y$
  obtains $P\ y\ R\ x\ y$
  ⟨*proof*⟩

lemma *rel-inv-restrict-left-inv-restrict-left-eq*:
  fixes $R :: 'a \Rightarrow 'b \Rightarrow bool$ **and** $P :: 'a \Rightarrow bool$ **and** $Q :: 'b \Rightarrow bool$
  shows $(((R{\upharpoonright}_{P})^{-1}){\upharpoonright}_{Q})^{-1} = (((R^{-1}){\upharpoonright}_{Q})^{-1}){\upharpoonright}_{P}$
  ⟨*proof*⟩

lemma *restrict-left-right-eq-restrict-right-left*:
  fixes $R :: 'a \Rightarrow 'b \Rightarrow bool$ **and** $P :: 'a \Rightarrow bool$ **and** $Q :: 'b \Rightarrow bool$
  shows $R{\upharpoonright}_{P}{\upharpoonleft}_{Q} = R{\upharpoonleft}_{Q}{\upharpoonright}_{P}$
  ⟨*proof*⟩

lemma *in-dom-restrict-leftI* [*intro*]:
  assumes $R\ x\ y$
  and $P\ x$
  shows *in-dom* $R{\upharpoonright}_{P}\ x$
  ⟨*proof*⟩

lemma *in-dom-restrict-left-if-in-dom*:
  assumes *in-dom* $R\ x$
  and $P\ x$
  shows *in-dom* $R{\upharpoonright}_{P}\ x$
  ⟨*proof*⟩

lemma *in-dom-restrict-leftE* [*elim*]:
  assumes *in-dom* $R{\upharpoonright}_{P}\ x$
  obtains y **where** $P\ x\ R\ x\ y$
  ⟨*proof*⟩

lemma *in-codom-restrict-leftI* [*intro*]:
  assumes $R\ x\ y$
  and $P\ x$
  shows *in-codom* $R{\upharpoonright}_{P}\ y$
  ⟨*proof*⟩

lemma *in-codom-restrict-leftE* [*elim*]:
  assumes *in-codom* $R{\upharpoonright}_{P}\ y$
  obtains x **where** $P\ x\ R\ x\ y$
  ⟨*proof*⟩

definition *rel-bimap* $f\ g\ (R :: 'a \Rightarrow 'b \Rightarrow bool)\ x\ y \equiv R\ (f\ x)\ (g\ y)$

lemma *rel-bimap-eq* [*simp*]: *rel-bimap* $f\ g\ R\ x\ y = R\ (f\ x)\ (g\ y)$
  ⟨*proof*⟩

definition *rel-map f R* ≡ *rel-bimap f f R*

lemma *rel-bimap-self-eq-rel-map* [*simp*]: *rel-bimap f f R* = *rel-map f R*
⟨*proof*⟩

lemma *rel-map-eq* [*simp*]: *rel-map f R x y* = *R* (*f x*) (*f y*)
⟨*proof*⟩

end

1.1.2 Order

theory *Binary-Relations-Order-Base*
imports
Binary-Relation-Functions
HOL.Orderings
begin

lemma *le-relI* [*intro*]:
assumes $\bigwedge x\ y.\ R\ x\ y \Longrightarrow S\ x\ y$
shows $R \le S$
⟨*proof*⟩

lemma *le-relD* [*dest*]:
assumes $R \le S$
and *R x y*
shows *S x y*
⟨*proof*⟩

lemma *le-relE*:
assumes $R \le S$
and *R x y*
obtains *S x y*
⟨*proof*⟩

lemma *rel-inv-le-rel-inv-iff* [*iff*]: $R^{-1} \le S^{-1} \longleftrightarrow R \le S$
⟨*proof*⟩

lemma *restrict-left-top-eq* [*simp*]: $(R :: {}'a \Rightarrow \text{-})\restriction_{(\top :: {}'a \Rightarrow bool)} = R$
⟨*proof*⟩

end

1.1.3 Lattice

theory *Binary-Relations-Lattice*
imports

Binary-Relations-Order-Base
HOL.Boolean-Algebras
begin

Summary Basic results about the lattice structure on binary relations.

lemma *rel-infI* [*intro*]:
assumes $R\ x\ y$
and $S\ x\ y$
shows $(R \sqcap S)\ x\ y$
⟨*proof*⟩

lemma *rel-infE* [*elim*]:
assumes $(R \sqcap S)\ x\ y$
obtains $R\ x\ y\ S\ x\ y$
⟨*proof*⟩

lemma *rel-infD*:
assumes $(R \sqcap S)\ x\ y$
shows $R\ x\ y$ **and** $S\ x\ y$
⟨*proof*⟩

lemma *in-dom-rel-infI* [*intro*]:
assumes $R\ x\ y$
and $S\ x\ y$
shows *in-dom* $(R \sqcap S)\ x$
⟨*proof*⟩

lemma *in-dom-rel-infE* [*elim*]:
assumes *in-dom* $(R \sqcap S)\ x$
obtains y **where** $R\ x\ y\ S\ x\ y$
⟨*proof*⟩

lemma *in-codom-rel-infI* [*intro*]:
assumes $R\ x\ y$
and $S\ x\ y$
shows *in-codom* $(R \sqcap S)\ y$
⟨*proof*⟩

lemma *in-codom-rel-infE* [*elim*]:
assumes *in-codom* $(R \sqcap S)\ y$
obtains x **where** $R\ x\ y\ S\ x\ y$
⟨*proof*⟩

lemma *in-field-eq-in-dom-sup-in-codom*: *in-field* $L =$ (*in-dom* $L \sqcup$ *in-codom* L)
⟨*proof*⟩

lemma *in-dom-restrict-left-eq* [*simp*]: *in-dom* $R{\restriction}_P =$ (*in-dom* $R \sqcap P$)
⟨*proof*⟩

lemma *in-codom-restrict-left-eq* [*simp*]: *in-codom* $R\upharpoonleft_{P}$ = (*in-codom* $R \sqcap P$)
  ⟨*proof*⟩

lemma *restrict-left-restrict-left-eq* [*simp*]:
  fixes R :: $'a \Rightarrow$ - **and** P Q :: $'a \Rightarrow$ *bool*
  shows $R\upharpoonright_{P}\upharpoonright_{Q} = R\upharpoonright_{P} \sqcap R\upharpoonright_{Q}$
  ⟨*proof*⟩

lemma *restrict-left-restrict-right-eq* [*simp*]:
  fixes R :: $'a \Rightarrow 'b \Rightarrow$ *bool* **and** P :: $'a \Rightarrow$ *bool* **and** Q :: $'b \Rightarrow$ *bool*
  shows $R\upharpoonright_{P}\upharpoonleft_{Q} = R\upharpoonright_{P} \sqcap R\upharpoonleft_{Q}$
  ⟨*proof*⟩

lemma *restrict-right-restrict-left-eq* [*simp*]:
  fixes R :: $'a \Rightarrow 'b \Rightarrow$ *bool* **and** P :: $'b \Rightarrow$ *bool* **and** Q :: $'a \Rightarrow$ *bool*
  shows $R\upharpoonleft_{P}\upharpoonright_{Q} = R\upharpoonleft_{P} \sqcap R\upharpoonright_{Q}$
  ⟨*proof*⟩

lemma *restrict-right-restrict-right-eq* [*simp*]:
  fixes R :: $'a \Rightarrow 'b \Rightarrow$ *bool* **and** P Q :: $'b \Rightarrow$ *bool*
  shows $R\upharpoonleft_{P}\upharpoonleft_{Q} = R\upharpoonleft_{P} \sqcap R\upharpoonleft_{Q}$
  ⟨*proof*⟩

lemma *restrict-left-sup-eq* [*simp*]: $(R :: 'a \Rightarrow \text{-})\upharpoonright_{((P :: 'a \Rightarrow bool) \sqcup Q)} = R\upharpoonright_{P} \sqcup R\upharpoonright_{Q}$
  ⟨*proof*⟩

lemma *restrict-left-inf-eq* [*simp*]: $(R :: 'a \Rightarrow \text{-})\upharpoonright_{((P :: 'a \Rightarrow bool) \sqcap Q)} = R\upharpoonright_{P} \sqcap R\upharpoonright_{Q}$
  ⟨*proof*⟩

lemma *inf-rel-bimap-and-eq-restrict-left-restrict-right*:
  $R \sqcap$ (*rel-bimap* P Q $(\wedge)$) = $R\upharpoonright_{P}\upharpoonleft_{Q}$
  ⟨*proof*⟩

end

1.2 Functions

1.2.1 Basic Functions

theory *Functions-Base*
  imports *HOL-Basics-Base*
begin

definition *id* $x \equiv x$

lemma *id-eq-self* [*simp*]: *id x* = *x*
⟨*proof*⟩

definition *comp f g x* ≡ *f* (*g x*)

bundle *comp-syntax* **begin notation** *comp* (**infixl** ∘ *55*) **end**
bundle *no-comp-syntax* **begin no-notation** *comp* (**infixl** ∘ *55*) **end**
unbundle *comp-syntax*

lemma *comp-eq* [*simp*]: (*f* ∘ *g*) *x* = *f* (*g x*)
⟨*proof*⟩

lemma *id-comp-eq* [*simp*]: *id* ∘ *f* = *f*
⟨*proof*⟩

lemma *comp-id-eq* [*simp*]: *f* ∘ *id* = *f*
⟨*proof*⟩

definition *dep-fun-map f g h x* ≡ *g x* (*f x*) (*h* (*f x*))

abbreviation *fun-map f g h* ≡ *dep-fun-map f* (λ- -. *g*) *h*

bundle *dep-fun-map-syntax* **begin**
syntax
-fun-map :: ('*a* ⇒ '*b*) ⇒ ('*c* ⇒ '*d*) ⇒ ('*b* ⇒ '*c*) ⇒
('*a* ⇒ '*d*) ((-) → (-) [*41*, *40*] *40*)
-dep-fun-map :: *idt* ⇒ ('*a* ⇒ '*b*) ⇒ ('*c* ⇒ '*d*) ⇒ ('*b* ⇒ '*c*) ⇒
('*a* ⇒ '*d*) ([-/ : / -] → (-) [*41*, *41*, *40*] *40*)
end
bundle *no-dep-fun-map-syntax* **begin**
no-syntax
-fun-map :: ('*a* ⇒ '*b*) ⇒ ('*c* ⇒ '*d*) ⇒ ('*b* ⇒ '*c*) ⇒
('*a* ⇒ '*d*) ((-) → (-) [*41*, *40*] *40*)
-dep-fun-map :: *idt* ⇒ ('*a* ⇒ '*b*) ⇒ ('*c* ⇒ '*d*) ⇒ ('*b* ⇒ '*c*) ⇒
('*a* ⇒ '*d*) ([-/ : / -] → (-) [*41*, *41*, *40*] *40*)
end
unbundle *dep-fun-map-syntax*
translations
f → *g* ⇌ *CONST fun-map f g*
[*x* : *f*] → *g* ⇌ *CONST dep-fun-map f* (λ*x*. *g*)

lemma *dep-fun-map-eq* [*simp*]: ([*x* : *f*] → *g x*) *h x* = *g x* (*f x*) (*h* (*f x*))
⟨*proof*⟩

lemma *fun-map-eq-comp* [*simp*]: (*f* → *g*) *h* = *g* ∘ *h* ∘ *f*
⟨*proof*⟩

lemma *fun-map-eq* [*simp*]: (*f* → *g*) *h x* = *g* (*h* (*f x*))
⟨*proof*⟩

lemma *fun-map-id-eq-comp* [*simp*]: *fun-map id* = (∘)
  ⟨*proof*⟩

lemma *fun-map-id-eq-comp′* [*simp*]: (*f* → *id*) *h* = *h* ∘ *f*
  ⟨*proof*⟩

end

1.2.2 Lattice Syntax

theory *HOL-Syntax-Bundles-Lattices*
  imports
    HOL.Lattices
begin

bundle *lattice-syntax* — copied from theory Main
begin
notation
  bot (⊥)
  and *top* (⊤)
  and *inf* (**infixl** ⊓ *70*)
  and *sup* (**infixl** ⊔ *65*)
end
bundle *no-lattice-syntax*
begin
no-notation
  bot (⊥)
  and *top* (⊤)
  and *inf* (**infixl** ⊓ *70*)
  and *sup* (**infixl** ⊔ *65*)
end

unbundle *lattice-syntax*

end

1.2.3 Lattice

theory *Predicates-Lattice*
  imports
    HOL-Syntax-Bundles-Lattices
    HOL.Boolean-Algebras
begin

lemma *inf-predI* [*intro*]:
  assumes *P x*

and $Q\ x$
shows $(P \sqcap Q)\ x$
⟨*proof*⟩

lemma *inf-predE* [*elim*]:
assumes $(P \sqcap Q)\ x$
obtains $P\ x\ Q\ x$
⟨*proof*⟩

lemma *inf-predD*:
assumes $(P \sqcap Q)\ x$
shows $P\ x$ **and** $Q\ x$
⟨*proof*⟩

end

1.2.4 Relators

theory *Function-Relators*
imports
Binary-Relation-Functions
Functions-Base
Predicates-Lattice
begin

Summary Introduces the concept of function relators. The slogan of function relators is "related functions map related inputs to related outputs".

definition *Dep-Fun-Rel-rel* $R\ S\ f\ g \equiv \forall x\ y.\ R\ x\ y \longrightarrow S\ x\ y\ (f\ x)\ (g\ y)$

abbreviation *Fun-Rel-rel* $R\ S \equiv$ *Dep-Fun-Rel-rel* $R\ (\lambda\text{- -}.\ S)$

definition *Dep-Fun-Rel-pred* $P\ R\ f\ g \equiv \forall x.\ P\ x \longrightarrow R\ x\ (f\ x)\ (g\ x)$

abbreviation *Fun-Rel-pred* $P\ R \equiv$ *Dep-Fun-Rel-pred* $P\ (\lambda\text{-}.\ R)$

bundle *Dep-Fun-Rel-syntax* **begin**
syntax
-Fun-Rel-rel :: $('a \Rightarrow 'b \Rightarrow bool) \Rightarrow ('c \Rightarrow 'd \Rightarrow bool) \Rightarrow ('a \Rightarrow 'c) \Rightarrow$
$('b \Rightarrow 'd) \Rightarrow bool$ ((-) ⇛ (-) [41, 40] 40)
-Dep-Fun-Rel-rel :: $idt \Rightarrow idt \Rightarrow ('a \Rightarrow 'b \Rightarrow bool) \Rightarrow ('c \Rightarrow 'd \Rightarrow bool) \Rightarrow$
$('a \Rightarrow 'c) \Rightarrow ('b \Rightarrow 'd) \Rightarrow bool$ ([-/ -/ ::/ -] ⇛ (-) [41, 41, 41, 40] 40)
-Dep-Fun-Rel-rel-if :: $idt \Rightarrow idt \Rightarrow ('a \Rightarrow 'b \Rightarrow bool) \Rightarrow bool \Rightarrow ('c \Rightarrow 'd \Rightarrow bool)$
$\Rightarrow$
$('a \Rightarrow 'c) \Rightarrow ('b \Rightarrow 'd) \Rightarrow bool$ ([-/ -/ ::/ -/ |/ -] ⇛ (-) [41, 41, 41, 41, 40] 40)
-Fun-Rel-pred :: $('a \Rightarrow bool) \Rightarrow ('b \Rightarrow 'c \Rightarrow bool) \Rightarrow ('a \Rightarrow 'b) \Rightarrow$
$('a \Rightarrow 'c) \Rightarrow bool$ ([-] ⇛ (-) [41, 40] 40)
-Dep-Fun-Rel-pred :: $idt \Rightarrow ('a \Rightarrow bool) \Rightarrow ('b \Rightarrow 'c \Rightarrow bool) \Rightarrow$
$('a \Rightarrow 'b) \Rightarrow ('a \Rightarrow 'c) \Rightarrow bool$ ([-/ ::/ -] ⇛ (-) [41, 41, 40] 40)

-Dep-Fun-Rel-pred-if :: *idt* ⇒ (*'a* ⇒ *bool*) ⇒ *bool* ⇒ (*'b* ⇒ *'c* ⇒ *bool*) ⇒
(*'a* ⇒ *'b*) ⇒ (*'a* ⇒ *'c*) ⇒ *bool* ([-/ ::/ -/ |/ -] ⇛ (-) [*41*, *41*, *41*, *40*] *40*)
end
bundle *no-Dep-Fun-Rel-syntax* **begin**
no-syntax
-Fun-Rel-rel :: (*'a* ⇒ *'b* ⇒ *bool*) ⇒ (*'c* ⇒ *'d* ⇒ *bool*) ⇒ (*'a* ⇒ *'c*) ⇒
(*'b* ⇒ *'d*) ⇒ *bool* ((-) ⇛ (-) [*41*, *40*] *40*)
-Dep-Fun-Rel-rel :: *idt* ⇒ *idt* ⇒ (*'a* ⇒ *'b* ⇒ *bool*) ⇒ (*'c* ⇒ *'d* ⇒ *bool*) ⇒
(*'a* ⇒ *'c*) ⇒ (*'b* ⇒ *'d*) ⇒ *bool* ([-/ -/ ::/ -] ⇛ (-) [*41*, *41*, *41*, *40*] *40*)
-Dep-Fun-Rel-rel-if :: *idt* ⇒ *idt* ⇒ (*'a* ⇒ *'b* ⇒ *bool*) ⇒ *bool* ⇒ (*'c* ⇒ *'d* ⇒ *bool*)
⇒
(*'a* ⇒ *'c*) ⇒ (*'b* ⇒ *'d*) ⇒ *bool* ([-/ -/ ::/ -/ |/ -] ⇛ (-) [*41*, *41*, *41*, *41*, *40*] *40*)
-Fun-Rel-pred :: (*'a* ⇒ *bool*) ⇒ (*'b* ⇒ *'c* ⇒ *bool*) ⇒ (*'a* ⇒ *'b*) ⇒
(*'a* ⇒ *'c*) ⇒ *bool* ([-] ⇛ (-) [*41*, *40*] *40*)
-Dep-Fun-Rel-pred :: *idt* ⇒ (*'a* ⇒ *bool*) ⇒ (*'b* ⇒ *'c* ⇒ *bool*) ⇒
(*'a* ⇒ *'b*) ⇒ (*'a* ⇒ *'c*) ⇒ *bool* ([-/ ::/ -] ⇛ (-) [*41*, *41*, *40*] *40*)
-Dep-Fun-Rel-pred-if :: *idt* ⇒ (*'a* ⇒ *bool*) ⇒ *bool* ⇒ (*'b* ⇒ *'c* ⇒ *bool*) ⇒
(*'a* ⇒ *'b*) ⇒ (*'a* ⇒ *'c*) ⇒ *bool* ([-/ ::/ -/ |/ -] ⇛ (-) [*41*, *41*, *41*, *40*] *40*)
end
unbundle *Dep-Fun-Rel-syntax*
translations
R ⇛ *S* ⇌ *CONST Fun-Rel-rel R S*
[*x y* :: *R*] ⇛ *S* ⇌ *CONST Dep-Fun-Rel-rel R* (λ*x y*. *S*)
[*x y* :: *R* | *B*] ⇛ *S* ⇌ *CONST Dep-Fun-Rel-rel R* (λ*x y*. *CONST rel-if B S*)
[*P*] ⇛ *R* ⇌ *CONST Fun-Rel-pred P R*
[*x* :: *P*] ⇛ *R* ⇌ *CONST Dep-Fun-Rel-pred P* (λ*x*. *R*)
[*x* :: *P* | *B*] ⇛ *R* ⇌ *CONST Dep-Fun-Rel-pred P* (λ*x*. *CONST rel-if B R*)

lemma *Dep-Fun-Rel-relI* [*intro*]:
assumes ⋀*x y*. *R x y* ⟹ *S x y* (*f x*) (*g y*)
shows ([*x y* :: *R*] ⇛ *S x y*) *f g*
⟨*proof*⟩

lemma *Dep-Fun-Rel-relD*:
assumes ([*x y* :: *R*] ⇛ *S x y*) *f g*
and *R x y*
shows *S x y* (*f x*) (*g y*)
⟨*proof*⟩

lemma *Dep-Fun-Rel-relE* [*elim*]:
assumes ([*x y* :: *R*] ⇛ *S x y*) *f g*
and *R x y*
obtains *S x y* (*f x*) (*g y*)
⟨*proof*⟩

lemma *Dep-Fun-Rel-predI* [*intro*]:
assumes ⋀*x*. *P x* ⟹ *R x* (*f x*) (*g x*)
shows ([*x* :: *P*] ⇛ *R x*) *f g*
⟨*proof*⟩

lemma *Dep-Fun-Rel-predD*:
assumes $([x :: P] \Rrightarrow R\ x)\ f\ g$
and $P\ x$
shows $R\ x\ (f\ x)\ (g\ x)$
⟨*proof*⟩

lemma *Dep-Fun-Rel-predE* [*elim*]:
assumes $([x :: P] \Rrightarrow R\ x)\ f\ g$
and $P\ x$
obtains $R\ x\ (f\ x)\ (g\ x)$
⟨*proof*⟩

lemma *rel-inv-Dep-Fun-Rel-rel-eq* [*simp*]:
$([x\ y :: R] \Rrightarrow S\ x\ y)^{-1} = ([y\ x :: R^{-1}] \Rrightarrow (S\ x\ y)^{-1})$
⟨*proof*⟩

lemma *rel-inv-Dep-Fun-Rel-pred-eq* [*simp*]:
$([x :: P] \Rrightarrow R\ x)^{-1} = ([x :: P] \Rrightarrow (R\ x)^{-1})$
⟨*proof*⟩

lemma *Dep-Fun-Rel-pred-eq-Dep-Fun-Rel-rel*:
$([x :: P] \Rrightarrow R\ x) = ([x\ \text{-} :: (((\sqcap)\ P) \circ (=))] \Rrightarrow R\ x)$
⟨*proof*⟩

lemma *Fun-Rel-eq-eq-eq* [*simp*]: $((=) \Rrightarrow (=)) = (=)$
⟨*proof*⟩

Composition

lemma *Dep-Fun-Rel-rel-compI*:
assumes *Dep-Fun-Rel1*: $([x\ y :: R] \Rrightarrow S\ x\ y)\ f\ g$
and *Dep-Fun-Rel2*: $\bigwedge x\ y.\ R\ x\ y \Longrightarrow ([x'\ y' :: T\ x\ y] \Rrightarrow U\ x\ y\ x'\ y')\ f'\ g'$
and *le*: $\bigwedge x\ y.\ R\ x\ y \Longrightarrow S\ x\ y\ (f\ x)\ (g\ y) \Longrightarrow T\ x\ y\ (f\ x)\ (g\ y)$
shows $([x\ y :: R] \Rrightarrow U\ x\ y\ (f\ x)\ (g\ y))\ (f' \circ f)\ (g' \circ g)$
⟨*proof*⟩

corollary *Dep-Fun-Rel-rel-compI'*:
assumes $([x\ y :: R] \Rrightarrow S\ x\ y)\ f\ g$
and $\bigwedge x\ y.\ R\ x\ y \Longrightarrow ([x'\ y' :: S\ x\ y] \Rrightarrow T\ x\ y\ x'\ y')\ f'\ g'$
shows $([x\ y :: R] \Rrightarrow T\ x\ y\ (f\ x)\ (g\ y))\ (f' \circ f)\ (g' \circ g)$
⟨*proof*⟩

lemma *Dep-Fun-Rel-pred-comp-Dep-Fun-Rel-rel-compI*:
assumes *Dep-Fun-Rel1*: $([x :: P] \Rrightarrow R\ x)\ f\ g$
and *Dep-Fun-Rel2*: $\bigwedge x.\ P\ x \Longrightarrow ([x'\ y' :: S\ x] \Rrightarrow T\ x\ x'\ y')\ f'\ g'$
and *le*: $\bigwedge x.\ P\ x \Longrightarrow R\ x\ (f\ x)\ (g\ x) \Longrightarrow S\ x\ (f\ x)\ (g\ x)$
shows $([x :: P] \Rrightarrow T\ x\ (f\ x)\ (g\ x))\ (f' \circ f)\ (g' \circ g)$
⟨*proof*⟩

corollary *Dep-Fun-Rel-pred-comp-Dep-Fun-Rel-rel-compI'*:

assumes $([x :: P] \Rrightarrow R\ x)\ f\ g$
and $\bigwedge x.\ P\ x \Longrightarrow ([x'\ y' :: R\ x] \Rrightarrow S\ x\ x'\ y')\ f'\ g'$
shows $([x :: P] \Rrightarrow S\ x\ (f\ x)\ (g\ x))\ (f' \circ f)\ (g' \circ g)$
$\langle proof \rangle$

Restrictions **lemma** *restrict-left-Dep-Fun-Rel-rel-restrict-left-eq*:
fixes $R :: 'a1 \Rightarrow 'a2 \Rightarrow bool$
and $S :: 'a1 \Rightarrow 'a2 \Rightarrow 'b1 \Rightarrow 'b2 \Rightarrow bool$
and $P :: 'a1 \Rightarrow 'a2 \Rightarrow 'b1 \Rightarrow bool$
assumes $\bigwedge f\ x\ y.\ Q\ f \Longrightarrow R\ x\ y \Longrightarrow P\ x\ y\ (f\ x)$
shows $([x\ y :: R] \Rrightarrow (S\ x\ y){\restriction}_{P\ x\ y}){\restriction}_{Q} = ([x\ y :: R] \Rrightarrow S\ x\ y){\restriction}_{Q}$
$\langle proof \rangle$

lemma *restrict-right-Dep-Fun-Rel-rel-restrict-right-eq*:
fixes $R :: 'a1 \Rightarrow 'a2 \Rightarrow bool$
and $S :: 'a1 \Rightarrow 'a2 \Rightarrow 'b1 \Rightarrow 'b2 \Rightarrow bool$
and $P :: 'a1 \Rightarrow 'a2 \Rightarrow 'b2 \Rightarrow bool$
assumes $\bigwedge f\ x\ y.\ Q\ f \Longrightarrow R\ x\ y \Longrightarrow P\ x\ y\ (f\ y)$
shows $(([x\ y :: R] \Rrightarrow (S\ x\ y){\upharpoonleft}_{P\ x\ y}){\upharpoonleft}_{Q}) = (([x\ y :: R] \Rrightarrow S\ x\ y){\upharpoonleft}_{Q})$
$\langle proof \rangle$

end

1.2.5 Orders

theory *Predicates-Order*
imports
HOL.Orderings
begin

lemma *le-predI* [*intro*]:
assumes $\bigwedge x.\ P\ x \Longrightarrow Q\ x$
shows $P \le Q$
$\langle proof \rangle$

lemma *le-predD* [*dest*]:
assumes $P \le Q$
and $P\ x$
shows $Q\ x$
$\langle proof \rangle$

lemma *le-predE*:
assumes $P \le Q$
and $P\ x$
obtains $Q\ x$
$\langle proof \rangle$

end

1.3 Predicates

theory *Predicates*
  imports
    Functions-Base
    Predicates-Order
    Predicates-Lattice
begin

Summary Basic concepts on predicates.

definition *pred-map* f $(P :: 'a \Rightarrow bool)$ $x \equiv P\ (f\ x)$

lemma *pred-map-eq* [*simp*]: *pred-map* $f\ P\ x = P\ (f\ x)$
  ⟨*proof*⟩

lemma *comp-eq-pred-map* [*simp*]: $P \circ f =$ *pred-map* $f\ P$
  ⟨*proof*⟩

end

Monotonicity

theory *Functions-Monotone*
  imports
    Binary-Relations-Order-Base
    Function-Relators
    Predicates
begin

Summary Introduces the concept of monotone functions. A function is monotone if it is related to itself - see *Dep-Fun-Rel-rel*.

declare *le-funI*[*intro*]
declare *le-funE*[*elim*]

definition *dep-mono-wrt-rel* $R\ S\ f \equiv ([x\ y :: R] \Rrightarrow S\ x\ y)\ f\ f$

abbreviation *mono-wrt-rel* $R\ S \equiv$ *dep-mono-wrt-rel* $R\ (\lambda\text{- -}.\ S)$

definition *dep-mono-wrt-pred* $P\ Q\ f \equiv ([x :: P] \Rrightarrow (\lambda\text{-}.\ Q\ x))\ f\ f$

abbreviation *mono-wrt-pred* $P\ Q \equiv$ *dep-mono-wrt-pred* $P\ (\lambda\text{-}.\ Q)$

bundle *dep-mono-wrt-syntax* **begin**
syntax

-mono-wrt-rel :: (*'a* ⇒ *'a* ⇒ *bool*) ⇒ (*'b* ⇒ *'b* ⇒ *bool*) ⇒ (*'a* ⇒ *'b*) ⇒
bool ((-) $\Rrightarrow_m$ (-) [*41*, *40*] *40*)
-dep-mono-wrt-rel :: *idt* ⇒ *idt* ⇒ (*'a* ⇒ *'a* ⇒ *bool*) ⇒ (*'b* ⇒ *'b* ⇒ *bool*) ⇒
(*'a* ⇒ *'b*) ⇒ *bool* ([-/ -/ ::/ -] $\Rrightarrow_m$ (-) [*41*, *41*, *41*, *40*] *40*)
-dep-mono-wrt-rel-if :: *idt* ⇒ *idt* ⇒ (*'a* ⇒ *'a* ⇒ *bool*) ⇒ *bool* ⇒ (*'b* ⇒ *'b* ⇒ *bool*)
⇒
(*'a* ⇒ *'b*) ⇒ *bool* ([-/ -/ ::/ -/ |/ -] $\Rrightarrow_m$ (-) [*41*, *41*, *41*, *41*, *40*] *40*)
-mono-wrt-pred :: (*'a* ⇒ *bool*) ⇒ (*'b* ⇒ *'b* ⇒ *bool*) ⇒ (*'a* ⇒ *'b*) ⇒
bool ([-] $\Rrightarrow_m$ (-) [*41*, *40*] *40*)
-dep-mono-wrt-pred :: *idt* ⇒ (*'a* ⇒ *bool*) ⇒ (*'b* ⇒ *'b* ⇒ *bool*) ⇒
(*'a* ⇒ *'b*) ⇒ *bool* ([-/ ::/ -] $\Rrightarrow_m$ (-) [*41*, *41*, *40*] *40*)

end
bundle *no-dep-mono-wrt-syntax* **begin**
no-syntax
-mono-wrt-rel :: (*'a* ⇒ *'a* ⇒ *bool*) ⇒ (*'b* ⇒ *'b* ⇒ *bool*) ⇒ (*'a* ⇒ *'b*) ⇒
bool ((-) $\Rrightarrow_m$ (-) [*41*, *40*] *40*)
-dep-mono-wrt-rel :: *idt* ⇒ *idt* ⇒ (*'a* ⇒ *'a* ⇒ *bool*) ⇒ (*'b* ⇒ *'b* ⇒ *bool*) ⇒
(*'a* ⇒ *'b*) ⇒ *bool* ([-/ -/ ::/ -] $\Rrightarrow_m$ (-) [*41*, *41*, *41*, *40*] *40*)
-dep-mono-wrt-rel-if :: *idt* ⇒ *idt* ⇒ (*'a* ⇒ *'a* ⇒ *bool*) ⇒ *bool* ⇒ (*'b* ⇒ *'b* ⇒ *bool*)
⇒
(*'a* ⇒ *'b*) ⇒ *bool* ([-/ -/ ::/ -/ |/ -] $\Rrightarrow_m$ (-) [*41*, *41*, *41*, *41*, *40*] *40*)
-mono-wrt-pred :: (*'a* ⇒ *bool*) ⇒ (*'b* ⇒ *'b* ⇒ *bool*) ⇒ (*'a* ⇒ *'b*) ⇒
bool ([-] $\Rrightarrow_m$ (-) [*41*, *40*] *40*)
-dep-mono-wrt-pred :: *idt* ⇒ (*'a* ⇒ *bool*) ⇒ (*'b* ⇒ *'b* ⇒ *bool*) ⇒
(*'a* ⇒ *'b*) ⇒ *bool* ([-/ ::/ -] $\Rrightarrow_m$ (-) [*41*, *41*, *40*] *40*)

end
unbundle *dep-mono-wrt-syntax*
translations
R $\Rrightarrow_m$ *S* ⇌ *CONST mono-wrt-rel R S*
[*x y* :: *R*] $\Rrightarrow_m$ *S* ⇌ *CONST dep-mono-wrt-rel R* (λ*x y*. *S*)
[*x y* :: *R* | *B*] $\Rrightarrow_m$ *S* ⇌ *CONST dep-mono-wrt-rel R* (λ*x y*. *CONST rel-if B S*)
[*P*] $\Rrightarrow_m$ *Q* ⇌ *CONST mono-wrt-pred P Q*
[*x* :: *P*] $\Rrightarrow_m$ *Q* ⇌ *CONST dep-mono-wrt-pred P* (λ*x*. *Q*)

lemma *dep-mono-wrt-relI* [*intro*]:
assumes ⋀*x y*. *R x y* ⟹ *S x y* (*f x*) (*f y*)
shows ([*x y* :: *R*] $\Rrightarrow_m$ *S x y*) *f*
⟨*proof*⟩

lemma *dep-mono-wrt-relE* [*elim*]:
assumes ([*x y* :: *R*] $\Rrightarrow_m$ *S x y*) *f*
and *R x y*
obtains *S x y* (*f x*) (*f y*)

⟨*proof*⟩

lemma *dep-mono-wrt-relD*:
assumes $([x\ y :: R] \Rrightarrow_m S\ x\ y)\ f$
and $R\ x\ y$
shows $S\ x\ y\ (f\ x)\ (f\ y)$
⟨*proof*⟩

lemma *dep-mono-wrt-predI* [*intro*]:
assumes $\bigwedge x.\ P\ x \Longrightarrow Q\ x\ (f\ x)$
shows $([x :: P] \Rrightarrow_m Q\ x)\ f$
⟨*proof*⟩

lemma *dep-mono-wrt-predE* [*elim*]:
assumes $([x :: P] \Rrightarrow_m Q\ x)\ f$
and $P\ x$
obtains $Q\ x\ (f\ x)$
⟨*proof*⟩

lemma *dep-mono-wrt-predD*:
assumes $([x :: P] \Rrightarrow_m Q\ x)\ f$
and $P\ x$
shows $Q\ x\ (f\ x)$
⟨*proof*⟩

lemma *dep-mono-wrt-rel-if-Dep-Fun-Rel-rel-self*:
assumes $([x\ y :: R] \Rrightarrow S\ x\ y)\ f\ f$
shows $([x\ y :: R] \Rrightarrow_m S\ x\ y)\ f$
⟨*proof*⟩

lemma *dep-mono-wrt-pred-if-Dep-Fun-Rel-pred-self*:
assumes $([x :: P] \Rrightarrow (\lambda\text{-}.\ Q\ x))\ f\ f$
shows $([x :: P] \Rrightarrow_m Q\ x)\ f$
⟨*proof*⟩

lemma *Dep-Fun-Rel-rel-self-if-dep-mono-wrt-rel*:
assumes $([x\ y :: R] \Rrightarrow_m S\ x\ y)\ f$
shows $([x\ y :: R] \Rrightarrow S\ x\ y)\ f\ f$
⟨*proof*⟩

lemma *Dep-Fun-Rel-pred-self-if-dep-mono-wrt-pred*:
assumes $([x :: P] \Rrightarrow_m Q\ x)\ f$
shows $([x :: P] \Rrightarrow (\lambda\text{-}.\ Q\ x))\ f\ f$
⟨*proof*⟩

corollary *Dep-Fun-Rel-rel-self-iff-dep-mono-wrt-rel*:
$([x\ y :: R] \Rrightarrow S\ x\ y)\ f\ f \longleftrightarrow ([x\ y :: R] \Rrightarrow_m S\ x\ y)\ f$
⟨*proof*⟩

corollary *Dep-Fun-Rel-pred-self-iff-dep-mono-wrt-pred*:
$([x :: P] \Rrightarrow (\lambda\text{-}.\ Q\ x))\ f\ f \longleftrightarrow ([x :: P] \Rrightarrow_m\ Q\ x)\ f$
⟨*proof*⟩

lemma *dep-mono-wrt-rel-inv-eq* [*simp*]:
$([y\ x :: R^{-1}] \Rrightarrow_m (S\ x\ y)^{-1}) = ([x\ y :: R] \Rrightarrow_m S\ x\ y)$
⟨*proof*⟩

lemma *in-dom-if-rel-if-dep-mono-wrt-rel*:
assumes $([x\ y :: R] \Rrightarrow_m S\ x\ y)\ f$
and $R\ x\ y$
shows *in-dom* $(S\ x\ y)\ (f\ x)$
⟨*proof*⟩

corollary *in-dom-if-in-dom-if-mono-wrt-rel*:
assumes $(R \Rrightarrow_m S)\ f$
shows $([\mathit{in\text{-}dom}\ R] \Rrightarrow_m \mathit{in\text{-}dom}\ S)\ f$
⟨*proof*⟩

lemma *in-codom-if-rel-if-dep-mono-wrt-rel*:
assumes $([x\ y :: R] \Rrightarrow_m S\ x\ y)\ f$
and $R\ x\ y$
shows *in-codom* $(S\ x\ y)\ (f\ y)$
⟨*proof*⟩

corollary *in-codom-if-in-codom-if-mono-wrt-rel*:
assumes $(R \Rrightarrow_m S)\ f$
shows $([\mathit{in\text{-}codom}\ R] \Rrightarrow_m \mathit{in\text{-}codom}\ S)\ f$
⟨*proof*⟩

corollary *in-field-if-in-field-if-mono-wrt-rel*:
assumes $(R \Rrightarrow_m S)\ f$
shows $([\mathit{in\text{-}field}\ R] \Rrightarrow_m \mathit{in\text{-}field}\ S)\ f$
⟨*proof*⟩

lemma *le-rel-map-if-mono-wrt-rel*:
assumes $(R \Rrightarrow_m S)\ f$
shows $R \le \mathit{rel\text{-}map}\ f\ S$
⟨*proof*⟩

lemma *le-pred-map-if-mono-wrt-pred*:
assumes $([P] \Rrightarrow_m Q)\ f$
shows $P \le \mathit{pred\text{-}map}\ f\ Q$
⟨*proof*⟩

lemma *mono-wrt-rel-if-le-rel-map*:
assumes $R \le \mathit{rel\text{-}map}\ f\ S$
shows $(R \Rrightarrow_m S)\ f$
⟨*proof*⟩

lemma *mono-wrt-pred-if-le-pred-map*:
  assumes $P \leq$ *pred-map* $f\ Q$
  shows $([P] \Rrightarrow_m Q)\ f$
  ⟨*proof*⟩

corollary *mono-wrt-rel-iff-le-rel-map*: $(R \Rrightarrow_m S)\ f \longleftrightarrow R \leq$ *rel-map* $f\ S$
  ⟨*proof*⟩

corollary *mono-wrt-pred-iff-le-pred-map*: $([P] \Rrightarrow_m Q)\ f \longleftrightarrow P \leq$ *pred-map* $f\ Q$
  ⟨*proof*⟩

definition *mono* $\equiv ((\leq) \Rrightarrow_m (\leq))$

definition *antimono* $\equiv ((\leq) \Rrightarrow_m (\geq))$

lemma *monoI* [*intro*]:
  assumes $\bigwedge x\ y.\ x \leq y \Longrightarrow f\ x \leq f\ y$
  shows *mono* f
  ⟨*proof*⟩

lemma *monoE* [*elim*]:
  assumes *mono* f
  and $x \leq y$
  obtains $f\ x \leq f\ y$
  ⟨*proof*⟩

lemma *monoD*:
  assumes *mono* f
  and $x \leq y$
  shows $f\ x \leq f\ y$
  ⟨*proof*⟩

lemma *antimonoI* [*intro*]:
  assumes $\bigwedge x\ y.\ x \leq y \Longrightarrow f\ y \leq f\ x$
  shows *antimono* f
  ⟨*proof*⟩

lemma *antimonoE* [*elim*]:
  assumes *antimono* f
  and $x \leq y$
  obtains $f\ y \leq f\ x$
  ⟨*proof*⟩

lemma *antimonoD*:
  assumes *antimono* f
  and $x \leq y$
  shows $f\ y \leq f\ x$
  ⟨*proof*⟩

lemma *antimono-Dep-Fun-Rel-rel-left*: *antimono* $(\lambda R.\ [x\ y :: R] \Rrightarrow S\ x\ y)$
⟨*proof*⟩

lemma *antimono-Dep-Fun-Rel-pred-left*: *antimono* $(\lambda P.\ [x :: P] \Rrightarrow Q\ x)$
⟨*proof*⟩

lemma *antimono-dep-mono-wrt-rel-left*: *antimono* $(\lambda R.\ [x\ y :: R] \Rrightarrow_m S\ x\ y)$
⟨*proof*⟩

lemma *antimono-dep-mono-wrt-pred-left*: *antimono* $(\lambda P.\ [x :: P] \Rrightarrow_m Q\ x)$
⟨*proof*⟩

lemma *Dep-Fun-Rel-rel-if-le-left-if-Dep-Fun-Rel-rel*:
assumes $([x\ y :: R] \Rrightarrow S\ x\ y)\ f\ g$
and $T \leq R$
shows $([x\ y :: T] \Rrightarrow S\ x\ y)\ f\ g$
⟨*proof*⟩

lemma *Dep-Fun-Rel-pred-if-le-left-if-Dep-Fun-Rel-pred*:
assumes $([x :: P] \Rrightarrow Q\ x)\ f\ g$
and $T \leq P$
shows $([x :: T] \Rrightarrow Q\ x)\ f\ g$
⟨*proof*⟩

lemma *dep-mono-wrt-rel-if-le-left-if-dep-mono-wrt-rel*:
assumes $([x\ y :: R] \Rrightarrow_m S\ x\ y)\ f$
and $T \leq R$
shows $([x\ y :: T] \Rrightarrow_m S\ x\ y)\ f$
⟨*proof*⟩

lemma *dep-mono-wrt-pred-if-le-left-if-dep-mono-wrt-pred*:
assumes $([x :: P] \Rrightarrow_m Q\ x)\ f$
and $T \leq P$
shows $([x :: T] \Rrightarrow_m Q\ x)\ f$
⟨*proof*⟩

lemma *mono-Dep-Fun-Rel-rel-right*: *mono* $(\lambda S.\ [x\ y :: R] \Rrightarrow S\ x\ y)$
⟨*proof*⟩

lemma *mono-Dep-Fun-Rel-pred-right*: *mono* $(\lambda Q.\ [x :: P] \Rrightarrow Q\ x)$
⟨*proof*⟩

lemma *mono-dep-mono-wrt-rel-right*: *mono* $(\lambda S.\ [x\ y :: R] \Rrightarrow_m S\ x\ y)$
⟨*proof*⟩

lemma *mono-dep-mono-wrt-pred-right*: *mono* $(\lambda Q.\ [x :: P] \Rrightarrow_m Q\ x)$
⟨*proof*⟩

lemma *Dep-Fun-Rel-rel-if-le-right-if-Dep-Fun-Rel-rel*:
assumes $([x\ y :: R] \Rrightarrow S\ x\ y)\ f\ g$
and $\bigwedge x\ y.\ R\ x\ y \Longrightarrow S\ x\ y\ (f\ x)\ (g\ y) \Longrightarrow T\ x\ y\ (f\ x)\ (g\ y)$
shows $([x\ y :: R] \Rrightarrow T\ x\ y)\ f\ g$
⟨*proof*⟩

lemma *Dep-Fun-Rel-pred-if-le-right-if-Dep-Fun-Rel-pred*:
assumes $([x :: P] \Rrightarrow Q\ x)\ f\ g$
and $\bigwedge x.\ P\ x \Longrightarrow Q\ x\ (f\ x)\ (g\ x) \Longrightarrow T\ x\ (f\ x)\ (g\ x)$
shows $([x :: P] \Rrightarrow T\ x)\ f\ g$
⟨*proof*⟩

lemma *dep-mono-wrt-rel-if-le-right-if-dep-mono-wrt-rel*:
assumes $([x\ y :: R] \Rrightarrow_m S\ x\ y)\ f$
and $\bigwedge x\ y.\ R\ x\ y \Longrightarrow S\ x\ y\ (f\ x)\ (f\ y) \Longrightarrow T\ x\ y\ (f\ x)\ (f\ y)$
shows $([x\ y :: R] \Rrightarrow_m T\ x\ y)\ f$
⟨*proof*⟩

lemma *dep-mono-wrt-pred-if-le-right-if-dep-mono-wrt-pred*:
assumes $([x :: P] \Rrightarrow_m Q\ x)\ f$
and $\bigwedge x.\ P\ x \Longrightarrow Q\ x\ (f\ x) \Longrightarrow T\ x\ (f\ x)$
shows $([x :: P] \Rrightarrow_m T\ x)\ f$
⟨*proof*⟩

Composition **lemma** *dep-mono-wrt-rel-compI*:
assumes $([x\ y :: R] \Rrightarrow_m S\ x\ y)\ f$
and $\bigwedge x\ y.\ R\ x\ y \Longrightarrow ([x'\ y' :: T\ x\ y] \Rrightarrow_m U\ x\ y\ x'\ y')\ f'$
and $\bigwedge x\ y.\ R\ x\ y \Longrightarrow S\ x\ y\ (f\ x)\ (f\ y) \Longrightarrow T\ x\ y\ (f\ x)\ (f\ y)$
shows $([x\ y :: R] \Rrightarrow_m U\ x\ y\ (f\ x)\ (f\ y))\ (f' \circ f)$
⟨*proof*⟩

corollary *dep-mono-wrt-rel-compI'*:
assumes $([x\ y :: R] \Rrightarrow_m S\ x\ y)\ f$
and $\bigwedge x\ y.\ R\ x\ y \Longrightarrow ([x'\ y' :: S\ x\ y] \Rrightarrow_m T\ x\ y\ x'\ y')\ f'$
shows $([x\ y :: R] \Rrightarrow_m T\ x\ y\ (f\ x)\ (f\ y))\ (f' \circ f)$
⟨*proof*⟩

lemma *dep-mono-wrt-pred-comp-dep-mono-wrt-rel-compI*:
assumes $([x :: P] \Rrightarrow_m Q\ x)\ f$
and $\bigwedge x.\ P\ x \Longrightarrow ([x'\ y' :: R\ x] \Rrightarrow_m S\ x\ x'\ y')\ f'$
and $\bigwedge x.\ P\ x \Longrightarrow Q\ x\ (f\ x) \Longrightarrow R\ x\ (f\ x)\ (f\ x)$
shows $([x :: P] \Rrightarrow_m (\lambda y.\ S\ x\ (f\ x)\ (f\ x)\ y\ y))\ (f' \circ f)$
⟨*proof*⟩

lemma *dep-mono-wrt-pred-comp-dep-mono-wrt-pred-compI*:
assumes $([x :: P] \Rrightarrow_m Q\ x)\ f$
and $\bigwedge x.\ P\ x \Longrightarrow ([x' :: R\ x] \Rrightarrow_m S\ x\ x')\ f'$
and $\bigwedge x.\ P\ x \Longrightarrow Q\ x\ (f\ x) \Longrightarrow R\ x\ (f\ x)$
shows $([x :: P] \Rrightarrow_m S\ x\ (f\ x))\ (f' \circ f)$

⟨*proof*⟩

corollary *dep-mono-wrt-pred-comp-dep-mono-wrt-pred-compI'*:
assumes $([x :: P] \Rrightarrow_m Q\ x)\ f$
and $\bigwedge x.\ P\ x \Longrightarrow ([x' :: Q\ x] \Rrightarrow_m S\ x\ x')\ f'$
shows $([x :: P] \Rrightarrow_m S\ x\ (f\ x))\ (f' \circ f)$
⟨*proof*⟩

Instantiations **lemma** *mono-wrt-rel-self-id*: $(R \Rrightarrow_m R)\ id$ ⟨*proof*⟩
lemma *mono-wrt-pred-self-id*: $([P] \Rrightarrow_m P)\ id$ ⟨*proof*⟩

lemma *mono-in-dom*: *mono in-dom* ⟨*proof*⟩
lemma *mono-in-codom*: *mono in-codom* ⟨*proof*⟩
lemma *mono-in-field*: *mono in-field* ⟨*proof*⟩
lemma *mono-rel-comp1*: *mono* $(\circ\circ)$ ⟨*proof*⟩
lemma *mono-rel-comp2*: *mono* $((\circ\circ)\ x)$ ⟨*proof*⟩

end

Reflexive

theory *Binary-Relations-Reflexive*
imports
Functions-Monotone
begin

consts *reflexive-on* :: $'a \Rightarrow ('b \Rightarrow 'b \Rightarrow bool) \Rightarrow bool$

overloading
reflexive-on-pred $\equiv$ *reflexive-on* :: $('a \Rightarrow bool) \Rightarrow ('a \Rightarrow 'a \Rightarrow bool) \Rightarrow bool$
begin
definition *reflexive-on-pred* $P\ R \equiv \forall x.\ P\ x \longrightarrow R\ x\ x$
end

lemma *reflexive-onI* [*intro*]:
assumes $\bigwedge x.\ P\ x \Longrightarrow R\ x\ x$
shows *reflexive-on* $P\ R$
⟨*proof*⟩

lemma *reflexive-onD* [*dest*]:
assumes *reflexive-on* $P\ R$
and $P\ x$
shows $R\ x\ x$
⟨*proof*⟩

lemma *le-in-dom-if-reflexive-on*:
assumes *reflexive-on* $P\ R$
shows $P \le$ *in-dom* R

⟨*proof*⟩

lemma *le-in-codom-if-reflexive-on*:
assumes *reflexive-on P R*
shows $P \leq$ *in-codom R*
⟨*proof*⟩

lemma *in-codom-eq-in-dom-if-reflexive-on-in-field*:
assumes *reflexive-on* (*in-field R*) *R*
shows *in-codom R* = *in-dom R*
⟨*proof*⟩

lemma *reflexive-on-rel-inv-iff-reflexive-on* [*iff*]:
reflexive-on P R^{-1} ⟷ *reflexive-on* (*P* :: ′*a* ⇒ *bool*) (*R* :: ′*a* ⇒ -)
⟨*proof*⟩

lemma *antimono-reflexive-on* [*iff*]:
antimono (λ(*P* :: ′*a* ⇒ *bool*). *reflexive-on P* (*R* :: ′*a* ⇒ -))
⟨*proof*⟩

lemma *reflexive-on-if-le-pred-if-reflexive-on*:
fixes *P P*′ :: ′*a* ⇒ *bool* **and** *R* :: ′*a* ⇒ -
assumes *reflexive-on P R*
and $P' \leq P$
shows *reflexive-on P*′ *R*
⟨*proof*⟩

lemma *reflexive-on-sup-eq* [*simp*]:
(*reflexive-on* :: (′*a* ⇒ *bool*) ⇒ (′*a* ⇒ -) ⇒ -) ((*P* :: ′*a* ⇒ *bool*) ⊔ *Q*)
= *reflexive-on P* ⊓ *reflexive-on Q*
⟨*proof*⟩

lemma *reflexive-on-iff-eq-restrict-left-le*:
reflexive-on (*P* :: ′*a* ⇒ *bool*) (*R* :: ′*a* ⇒ -) ⟷ ($(=)\restriction_P \leq R$)
⟨*proof*⟩

definition *reflexive* (*R* :: ′*a* ⇒ -) ≡ *reflexive-on* (⊤ :: ′*a* ⇒ *bool*) *R*

lemma *reflexive-eq-reflexive-on*:
reflexive (*R* :: ′*a* ⇒ -) = *reflexive-on* (⊤ :: ′*a* ⇒ *bool*) *R*
⟨*proof*⟩

lemma *reflexiveI* [*intro*]:
assumes $\bigwedge x.\ R\ x\ x$
shows *reflexive R*
⟨*proof*⟩

lemma *reflexiveD*:
assumes *reflexive R*

shows $R\ x\ x$
⟨*proof*⟩

lemma *reflexive-on-if-reflexive*:
fixes $P :: 'a \Rightarrow bool$ **and** $R :: 'a \Rightarrow$ -
assumes *reflexive* R
shows *reflexive-on* $P\ R$
⟨*proof*⟩

lemma *reflexive-rel-inv-iff-reflexive* [*iff*]:
reflexive $R^{-1} \longleftrightarrow$ *reflexive* R
⟨*proof*⟩

lemma *reflexive-iff-eq-le*: *reflexive* $R \longleftrightarrow ((=) \leq R)$
⟨*proof*⟩

Instantiations

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

lemma *reflexive-top*: *reflexive* $\top$
⟨*proof*⟩

end

Symmetric

theory *Binary-Relations-Symmetric*
imports
Functions-Monotone
begin

consts *symmetric-on* $:: 'a \Rightarrow ('b \Rightarrow 'b \Rightarrow bool) \Rightarrow bool$

overloading
symmetric-on-pred $\equiv$ *symmetric-on* $:: ('a \Rightarrow bool) \Rightarrow ('a \Rightarrow 'a \Rightarrow bool) \Rightarrow bool$
begin
definition *symmetric-on-pred* $P\ R \equiv \forall\, x\ y.\ P\ x \wedge P\ y \wedge R\ x\ y \longrightarrow R\ y\ x$
end

lemma *symmetric-onI* [*intro*]:
assumes $\bigwedge x\ y.\ P\ x \Longrightarrow P\ y \Longrightarrow R\ x\ y \Longrightarrow R\ y\ x$
shows *symmetric-on* $P\ R$
⟨*proof*⟩

lemma *symmetric-onD*:
assumes *symmetric-on* $P\ R$
and $P\ x\ P\ y$
and $R\ x\ y$
shows $R\ y\ x$

⟨*proof*⟩

lemma *symmetric-on-rel-inv-iff-symmetric-on* [*iff*]:
symmetric-on $P\ R^{-1} \longleftrightarrow$ *symmetric-on* $(P :: {'a} \Rightarrow bool)\ (R :: {'a} \Rightarrow \text{-})$
⟨*proof*⟩

lemma *antimono-symmetric-on* [*iff*]:
antimono $(\lambda(P :: {'a} \Rightarrow bool).$ *symmetric-on* $P\ (R :: {'a} \Rightarrow \text{-}))$
⟨*proof*⟩

lemma *symmetric-on-if-le-pred-if-symmetric-on*:
fixes $P\ P' :: {'a} \Rightarrow bool$ **and** $R :: {'a} \Rightarrow \text{-}$
assumes *symmetric-on* $P\ R$
and $P' \le P$
shows *symmetric-on* $P'\ R$
⟨*proof*⟩

definition *symmetric* $(R :: {'a} \Rightarrow \text{-}) \equiv$ *symmetric-on* $(\top :: {'a} \Rightarrow bool)\ R$

lemma *symmetric-eq-symmetric-on*:
symmetric $(R :: {'a} \Rightarrow \text{-}) =$ *symmetric-on* $(\top :: {'a} \Rightarrow bool)\ R$
⟨*proof*⟩

lemma *symmetricI* [*intro*]:
assumes $\bigwedge x\ y.\ R\ x\ y \Longrightarrow R\ y\ x$
shows *symmetric* R
⟨*proof*⟩

lemma *symmetricD*:
assumes *symmetric* R
and $R\ x\ y$
shows $R\ y\ x$
⟨*proof*⟩

lemma *symmetric-on-if-symmetric*:
fixes $P :: {'a} \Rightarrow bool$ **and** $R :: {'a} \Rightarrow \text{-}$
assumes *symmetric* R
shows *symmetric-on* $P\ R$
⟨*proof*⟩

lemma *symmetric-rel-inv-iff-symmetric* [*iff*]: *symmetric* $R^{-1} \longleftrightarrow$ *symmetric* R
⟨*proof*⟩

lemma *rel-inv-eq-self-if-symmetric* [*simp*]:
assumes *symmetric* R
shows $R^{-1} = R$
⟨*proof*⟩

lemma *rel-iff-rel-if-symmetric*:

assumes *symmetric R*
shows $R\ x\ y \longleftrightarrow R\ y\ x$
⟨*proof*⟩

lemma *symmetric-if-rel-inv-eq-self*:
assumes $R^{-1} = R$
shows *symmetric R*
⟨*proof*⟩

lemma *symmetric-iff-rel-inv-eq-self*: *symmetric* $R \longleftrightarrow R^{-1} = R$
⟨*proof*⟩

lemma *symmetric-if-symmetric-on-in-field*:
assumes *symmetric-on* (*in-field R*) *R*
shows *symmetric R*
⟨*proof*⟩

corollary *symmetric-on-in-field-iff-symmetric* [*simp*]:
symmetric-on (*in-field R*) $R \longleftrightarrow$ *symmetric R*
⟨*proof*⟩

Instantiations

lemma *symmetric-eq* [*iff*]: *symmetric* (=)
⟨*proof*⟩

lemma *symmetric-top*: *symmetric* $\top$
⟨*proof*⟩

end

Transitive

theory *Binary-Relations-Transitive*
imports
Binary-Relation-Functions
Functions-Monotone
begin

consts *transitive-on* :: $'a \Rightarrow ('b \Rightarrow 'b \Rightarrow bool) \Rightarrow bool$

overloading
transitive-on-pred $\equiv$ *transitive-on* :: $('a \Rightarrow bool) \Rightarrow ('a \Rightarrow 'a \Rightarrow bool) \Rightarrow bool$
begin
definition *transitive-on-pred* $P\ R \equiv \forall x\ y\ z.\ P\ x \wedge P\ y \wedge P\ z \wedge R\ x\ y \wedge R\ y\ z \longrightarrow R\ x\ z$
end

lemma *transitive-onI* [*intro*]:
assumes $\bigwedge x\ y\ z.\ P\ x \Longrightarrow P\ y \Longrightarrow P\ z \Longrightarrow R\ x\ y \Longrightarrow R\ y\ z \Longrightarrow R\ x\ z$
shows *transitive-on P R*

⟨*proof*⟩

lemma *transitive-onD*:
assumes *transitive-on P R*
and $P\ x\ P\ y\ P\ z$
and $R\ x\ y\ R\ y\ z$
shows $R\ x\ z$
⟨*proof*⟩

lemma *transitive-on-if-rel-comp-self-imp*:
assumes $\bigwedge x\ y.\ P\ x \Longrightarrow P\ y \Longrightarrow (R \circ\circ R)\ x\ y \Longrightarrow R\ x\ y$
shows *transitive-on P R*
⟨*proof*⟩

lemma *transitive-on-rel-inv-iff-transitive-on* [*iff*]:
transitive-on $P\ R^{-1} \longleftrightarrow$ *transitive-on* $(P :: 'a \Rightarrow bool)\ (R :: 'a \Rightarrow \text{-})$
⟨*proof*⟩

lemma *antimono-transitive-on* [*iff*]:
antimono $(\lambda(P :: 'a \Rightarrow bool).$ *transitive-on* $P\ (R :: 'a \Rightarrow \text{-}))$
⟨*proof*⟩

lemma *transitive-on-if-le-pred-if-transitive-on*:
fixes $P\ P' :: 'a \Rightarrow bool$ **and** $R :: 'a \Rightarrow \text{-}$
assumes *transitive-on P R*
and $P' \le P$
shows *transitive-on* $P'\ R$
⟨*proof*⟩

definition *transitive* $(R :: 'a \Rightarrow \text{-}) \equiv$ *transitive-on* $(\top :: 'a \Rightarrow bool)\ R$

lemma *transitive-eq-transitive-on*:
transitive $(R :: 'a \Rightarrow \text{-}) =$ *transitive-on* $(\top :: 'a \Rightarrow bool)\ R$
⟨*proof*⟩

lemma *transitiveI* [*intro*]:
assumes $\bigwedge x\ y\ z.\ R\ x\ y \Longrightarrow R\ y\ z \Longrightarrow R\ x\ z$
shows *transitive R*
⟨*proof*⟩

lemma *transitiveD* [*dest*]:
assumes *transitive R*
and $R\ x\ y\ R\ y\ z$
shows $R\ x\ z$
⟨*proof*⟩

lemma *transitive-on-if-transitive*:
fixes $P :: 'a \Rightarrow bool$ **and** $R :: 'a \Rightarrow \text{-}$
assumes *transitive R*

shows *transitive-on P R*
⟨*proof*⟩

lemma *transitive-if-rel-comp-le-self*:
assumes $R \circ\circ R \le R$
shows *transitive R*
⟨*proof*⟩

lemma *rel-comp-le-self-if-transitive*:
assumes *transitive R*
shows $R \circ\circ R \le R$
⟨*proof*⟩

corollary *transitive-iff-rel-comp-le-self*: *transitive R* $\longleftrightarrow R \circ\circ R \le R$
⟨*proof*⟩

lemma *transitive-if-transitive-on-in-field*:
assumes *transitive-on* (*in-field R*) *R*
shows *transitive R*
⟨*proof*⟩

corollary *transitive-on-in-field-iff-transitive* [*simp*]:
transitive-on (*in-field R*) *R* $\longleftrightarrow$ *transitive R*
⟨*proof*⟩

lemma *transitive-rel-inv-iff-transitive* [*iff*]:
transitive $R^{-1} \longleftrightarrow$ *transitive R*
⟨*proof*⟩

Instantiations **lemma** *transitive-eq*: *transitive* (=)
⟨*proof*⟩

lemma *transitive-top*: *transitive* ⊤
⟨*proof*⟩

end

theory *Binary-Relations-Order*
imports
Binary-Relations-Order-Base
Binary-Relations-Reflexive
Binary-Relations-Symmetric
Binary-Relations-Transitive
begin

Summary Basic results about the order on binary relations.

lemma *in-dom-if-rel-if-rel-comp-le*:
assumes $(R \circ\circ S) \le (S \circ\circ R)$

and $R\ x\ y\ S\ y\ z$
shows *in-dom* $S\ x$
⟨*proof*⟩

lemma *in-codom-if-rel-if-rel-comp-le*:
assumes $(R \circ\circ S) \leq (S \circ\circ R)$
and $R\ x\ y\ S\ y\ z$
shows *in-codom* $R\ z$
⟨*proof*⟩

lemma *rel-comp-le-rel-inv-if-rel-comp-le-if-symmetric*:
assumes *symms*: *symmetric* $R1$ *symmetric* $R2$
and *le*: $(R1 \circ\circ R2) \leq R3$
shows $(R2 \circ\circ R1) \leq R3^{-1}$
⟨*proof*⟩

lemma *rel-inv-le-rel-comp-if-le-rel-comp-if-symmetric*:
assumes *symms*: *symmetric* $R1$ *symmetric* $R2$
and *le*: $R3 \leq (R1 \circ\circ R2)$
shows $R3^{-1} \leq (R2 \circ\circ R1)$
⟨*proof*⟩

corollary *rel-comp-le-rel-comp-if-rel-comp-le-rel-comp-if-symmetric*:
assumes *symmetric* $R1$ *symmetric* $R2$ *symmetric* $R3$ *symmetric* $R4$
and $(R1 \circ\circ R2) \leq (R3 \circ\circ R4)$
shows $(R2 \circ\circ R1) \leq (R4 \circ\circ R3)$
⟨*proof*⟩

corollary *rel-comp-le-rel-comp-iff-if-symmetric*:
assumes *symmetric* $R1$ *symmetric* $R2$ *symmetric* $R3$ *symmetric* $R4$
shows $(R1 \circ\circ R2) \leq (R3 \circ\circ R4) \longleftrightarrow (R2 \circ\circ R1) \leq (R4 \circ\circ R3)$
⟨*proof*⟩

corollary *eq-if-le-rel-if-symmetric*:
assumes *symmetric* R *symmetric* S
and $(R \circ\circ S) \leq (S \circ\circ R)$
shows $(R \circ\circ S) = (S \circ\circ R)$
⟨*proof*⟩

lemma *rel-comp-le-rel-comp-if-le-rel-if-reflexive-on-in-codom-if-transitive*:
assumes *trans*: *transitive* S
and *refl-on*: *reflexive-on* (*in-codom* S) R
and *le-rel*: $R \leq S$
shows $R \circ\circ S \leq S \circ\circ R$
⟨*proof*⟩

end

Antisymmetric

theory *Binary-Relations-Antisymmetric*
  imports
    Binary-Relation-Functions
    HOL-Syntax-Bundles-Lattices
begin

consts *antisymmetric-on* :: $'a \Rightarrow ('b \Rightarrow 'b \Rightarrow bool) \Rightarrow bool$

overloading
  antisymmetric-on-pred $\equiv$ *antisymmetric-on* :: $('a \Rightarrow bool) \Rightarrow ('a \Rightarrow 'a \Rightarrow bool) \Rightarrow bool$
begin
  definition *antisymmetric-on-pred* $P\ R \equiv \forall x\ y.\ P\ x \wedge P\ y \wedge R\ x\ y \wedge R\ y\ x \longrightarrow x = y$
end

lemma *antisymmetric-onI* [*intro*]:
  assumes $\bigwedge x\ y.\ P\ x \Longrightarrow P\ y \Longrightarrow R\ x\ y \Longrightarrow R\ y\ x \Longrightarrow x = y$
  shows *antisymmetric-on* $P\ R$
  ⟨*proof*⟩

lemma *antisymmetric-onD*:
  assumes *antisymmetric-on* $P\ R$
  and $P\ x\ P\ y$
  and $R\ x\ y\ R\ y\ x$
  shows $x = y$
  ⟨*proof*⟩

definition *antisymmetric* $(R :: 'a \Rightarrow \text{-}) \equiv$ *antisymmetric-on* $(\top :: 'a \Rightarrow bool)\ R$

lemma *antisymmetric-eq-antisymmetric-on*:
  antisymmetric $(R :: 'a \Rightarrow \text{-}) =$ *antisymmetric-on* $(\top :: 'a \Rightarrow bool)\ R$
  ⟨*proof*⟩

lemma *antisymmetricI* [*intro*]:
  assumes $\bigwedge x\ y.\ R\ x\ y \Longrightarrow R\ y\ x \Longrightarrow x = y$
  shows *antisymmetric* R
  ⟨*proof*⟩

lemma *antisymmetricD*:
  assumes *antisymmetric* R
  and $R\ x\ y\ R\ y\ x$
  shows $x = y$
  ⟨*proof*⟩

lemma *antisymmetric-on-if-antisymmetric*:
  fixes $P :: 'a \Rightarrow bool$ **and** $R :: 'a \Rightarrow \text{-}$
  assumes *antisymmetric* R

shows *antisymmetric-on P R*
⟨*proof*⟩

lemma *antisymmetric-if-antisymmetric-on-in-field*:
assumes *antisymmetric-on* (*in-field R*) *R*
shows *antisymmetric R*
⟨*proof*⟩

corollary *antisymmetric-on-in-field-iff-antisymmetric* [*simp*]:
antisymmetric-on (*in-field R*) *R* ⟷ *antisymmetric R*
⟨*proof*⟩

end

Injective

theory *Binary-Relations-Injective*
imports
Binary-Relation-Functions
HOL-Syntax-Bundles-Lattices
ML-Unification.ML-Unification-HOL-Setup
begin

consts *rel-injective-on* :: *'a* ⇒ (*'b* ⇒ *'c* ⇒ *bool*) ⇒ *bool*

overloading
rel-injective-on-pred ≡ *rel-injective-on* :: (*'a* ⇒ *bool*) ⇒ (*'a* ⇒ *'b* ⇒ *bool*) ⇒ *bool*
begin
definition *rel-injective-on-pred P R* ≡ ∀ *x x' y*. *P x* ∧ *P x'* ∧ *R x y* ∧ *R x' y* ⟶ *x* = *x'*
end

lemma *rel-injective-onI* [*intro*]:
assumes ⋀*x x' y*. *P x* ⟹ *P x'* ⟹ *R x y* ⟹ *R x' y* ⟹ *x* = *x'*
shows *rel-injective-on P R*
⟨*proof*⟩

lemma *rel-injective-onD*:
assumes *rel-injective-on P R*
and *P x P x'*
and *R x y R x' y*
shows *x* = *x'*
⟨*proof*⟩

consts *rel-injective-at* :: *'a* ⇒ (*'b* ⇒ *'c* ⇒ *bool*) ⇒ *bool*

overloading
rel-injective-at-pred ≡ *rel-injective-at* :: (*'a* ⇒ *bool*) ⇒ (*'b* ⇒ *'a* ⇒ *bool*) ⇒ *bool*

begin
definition *rel-injective-at-pred* $P\ R \equiv \forall x\ x'\ y.\ P\ y \wedge R\ x\ y \wedge R\ x'\ y \longrightarrow x = x'$
end

lemma *rel-injective-atI* [*intro*]:
assumes $\bigwedge x\ x'\ y.\ P\ y \Longrightarrow R\ x\ y \Longrightarrow R\ x'\ y \Longrightarrow x = x'$
shows *rel-injective-at* $P\ R$
⟨*proof*⟩

lemma *rel-injective-atD*:
assumes *rel-injective-at* $P\ R$
and $P\ y$
and $R\ x\ y\ R\ x'\ y$
shows $x = x'$
⟨*proof*⟩

definition *rel-injective* $(R :: 'a \Rightarrow \text{-}) \equiv$ *rel-injective-on* $(\top :: 'a \Rightarrow bool)\ R$

lemma *rel-injective-eq-rel-injective-on*:
rel-injective $(R :: 'a \Rightarrow \text{-}) =$ *rel-injective-on* $(\top :: 'a \Rightarrow bool)\ R$
⟨*proof*⟩

lemma *rel-injectiveI* [*intro*]:
assumes $\bigwedge x\ x'\ y.\ R\ x\ y \Longrightarrow R\ x'\ y \Longrightarrow x = x'$
shows *rel-injective* R
⟨*proof*⟩

lemma *rel-injectiveD*:
assumes *rel-injective* R
and $R\ x\ y\ R\ x'\ y$
shows $x = x'$
⟨*proof*⟩

lemma *rel-injective-eq-rel-injective-at*:
rel-injective $(R :: 'a \Rightarrow 'b \Rightarrow bool) =$ *rel-injective-at* $(\top :: 'b \Rightarrow bool)\ R$
⟨*proof*⟩

lemma *rel-injective-on-if-rel-injective*:
fixes $P :: 'a \Rightarrow bool$ **and** $R :: 'a \Rightarrow \text{-}$
assumes *rel-injective* R
shows *rel-injective-on* $P\ R$
⟨*proof*⟩

lemma *rel-injective-at-if-rel-injective*:
fixes $P :: 'a \Rightarrow bool$ **and** $R :: 'b \Rightarrow 'a \Rightarrow bool$
assumes *rel-injective* R
shows *rel-injective-at* $P\ R$
⟨*proof*⟩

lemma *rel-injective-if-rel-injective-on-in-dom*:
  assumes *rel-injective-on* (*in-dom R*) *R*
  shows *rel-injective R*
  ⟨*proof*⟩

lemma *rel-injective-if-rel-injective-at-in-codom*:
  assumes *rel-injective-at* (*in-codom R*) *R*
  shows *rel-injective R*
  ⟨*proof*⟩

corollary *rel-injective-on-in-dom-iff-rel-injective* [*simp*]:
  rel-injective-on (*in-dom R*) *R* ⟷ *rel-injective R*
  ⟨*proof*⟩

corollary *rel-injective-at-in-codom-iff-rel-injective* [*iff*]:
  rel-injective-at (*in-codom R*) *R* ⟷ *rel-injective R*
  ⟨*proof*⟩

end

Irreflexive

theory *Binary-Relations-Irreflexive*
  imports
    Binary-Relation-Functions
    HOL-Syntax-Bundles-Lattices
begin

consts *irreflexive-on* :: *'a* ⇒ (*'b* ⇒ *'b* ⇒ *bool*) ⇒ *bool*

overloading
  irreflexive-on-pred ≡ *irreflexive-on* :: (*'a* ⇒ *bool*) ⇒ (*'a* ⇒ *'a* ⇒ *bool*) ⇒ *bool*
begin
  definition *irreflexive-on-pred P R* ≡ ∀ *x*. *P x* ⟶ ¬(*R x x*)
end

lemma *irreflexive-onI* [*intro*]:
  assumes ⋀*x*. *P x* ⟹ ¬(*R x x*)
  shows *irreflexive-on P R*
  ⟨*proof*⟩

lemma *irreflexive-onD* [*dest*]:
  assumes *irreflexive-on P R*
  and *P x*
  shows ¬(*R x x*)
  ⟨*proof*⟩

definition *irreflexive* $(R :: 'a \Rightarrow \text{-}) \equiv$ *irreflexive-on* $(\top :: 'a \Rightarrow bool)$ R

lemma *irreflexive-eq-irreflexive-on*:
irreflexive $(R :: 'a \Rightarrow \text{-}) =$ *irreflexive-on* $(\top :: 'a \Rightarrow bool)$ R
⟨*proof*⟩

lemma *irreflexiveI* [*intro*]:
assumes $\bigwedge x.\ \neg(R\ x\ x)$
shows *irreflexive* R
⟨*proof*⟩

lemma *irreflexiveD*:
assumes *irreflexive* R
shows $\neg(R\ x\ x)$
⟨*proof*⟩

lemma *irreflexive-on-if-irreflexive*:
fixes $P :: 'a \Rightarrow bool$ **and** $R :: 'a \Rightarrow \text{-}$
assumes *irreflexive* R
shows *irreflexive-on* P R
⟨*proof*⟩

end

Left Total

theory *Binary-Relations-Left-Total*
imports
Binary-Relation-Functions
HOL-Syntax-Bundles-Lattices
begin

consts *left-total-on* $:: 'a \Rightarrow ('b \Rightarrow 'c \Rightarrow bool) \Rightarrow bool$

overloading
left-total-on-pred $\equiv$ *left-total-on* $:: ('a \Rightarrow bool) \Rightarrow ('a \Rightarrow 'b \Rightarrow bool) \Rightarrow bool$
begin
definition *left-total-on-pred* P $R \equiv \forall x.\ P\ x \longrightarrow$ *in-dom* R x
end

lemma *left-total-onI* [*intro*]:
assumes $\bigwedge x.\ P\ x \Longrightarrow$ *in-dom* R x
shows *left-total-on* P R
⟨*proof*⟩

lemma *left-total-onE* [*elim*]:
assumes *left-total-on* P R
and $P\ x$

obtains *y* **where** *R x y*
⟨*proof*⟩

lemma *in-dom-if-left-total-on*:
assumes *left-total-on P R*
and *P x*
shows *in-dom R x*
⟨*proof*⟩

definition *left-total* $(R :: 'a \Rightarrow -) \equiv$ *left-total-on* $(\top :: 'a \Rightarrow bool)$ R

lemma *left-total-eq-left-total-on*:
left-total $(R :: 'a \Rightarrow -) =$ *left-total-on* $(\top :: 'a \Rightarrow bool)$ R
⟨*proof*⟩

lemma *left-totalI* [*intro*]:
assumes $\bigwedge x.$ *in-dom R x*
shows *left-total R*
⟨*proof*⟩

lemma *left-totalE*:
assumes *left-total R*
obtains *y* **where** *R x y*
⟨*proof*⟩

lemma *in-dom-if-left-total*:
assumes *left-total R*
shows *in-dom R x*
⟨*proof*⟩

lemma *left-total-on-if-left-total*:
fixes $P :: 'a \Rightarrow bool$ **and** $R :: 'a \Rightarrow -$
assumes *left-total R*
shows *left-total-on P R*
⟨*proof*⟩

end

Right Unique

theory *Binary-Relations-Right-Unique*
imports
Binary-Relations-Injective
HOL-Syntax-Bundles-Lattices
begin

consts *right-unique-on* $:: 'a \Rightarrow ('b \Rightarrow 'c \Rightarrow bool) \Rightarrow bool$

overloading
right-unique-on-pred ≡ *right-unique-on* :: $('a \Rightarrow bool) \Rightarrow ('a \Rightarrow 'b \Rightarrow bool) \Rightarrow bool$
begin
definition *right-unique-on-pred* $P\ R \equiv \forall x\ y\ y'.\ P\ x \wedge R\ x\ y \wedge R\ x\ y' \longrightarrow y = y'$
end

lemma *right-unique-onI* [*intro*]:
assumes $\bigwedge x\ y\ y'.\ P\ x \Longrightarrow R\ x\ y \Longrightarrow R\ x\ y' \Longrightarrow y = y'$
shows *right-unique-on* $P\ R$
⟨*proof*⟩

lemma *right-unique-onD*:
assumes *right-unique-on* $P\ R$
and $P\ x$
and $R\ x\ y\ R\ x\ y'$
shows $y = y'$
⟨*proof*⟩

consts *right-unique-at* :: $'a \Rightarrow ('b \Rightarrow 'c \Rightarrow bool) \Rightarrow bool$

overloading
right-unique-at-pred ≡ *right-unique-at* :: $('a \Rightarrow bool) \Rightarrow ('b \Rightarrow 'a \Rightarrow bool) \Rightarrow bool$
begin
definition *right-unique-at-pred* $P\ R \equiv \forall x\ y\ y'.\ P\ y \wedge P\ y' \wedge R\ x\ y \wedge R\ x\ y' \longrightarrow y = y'$
end

lemma *right-unique-atI* [*intro*]:
assumes $\bigwedge x\ y\ y'.\ P\ y \Longrightarrow P\ y' \Longrightarrow R\ x\ y \Longrightarrow R\ x\ y' \Longrightarrow y = y'$
shows *right-unique-at* $P\ R$
⟨*proof*⟩

lemma *right-unique-atD*:
assumes *right-unique-at* $P\ R$
and $P\ y$
and $P\ y'$
and $R\ x\ y\ R\ x\ y'$
shows $y = y'$
⟨*proof*⟩

lemma *right-unique-at-rel-inv-iff-rel-injective-on* [*iff*]:
right-unique-at $(P :: 'a \Rightarrow bool)\ (R^{-1} :: 'b \Rightarrow 'a \Rightarrow bool) \longleftrightarrow$ *rel-injective-on* $P\ R$
⟨*proof*⟩

lemma *rel-injective-on-rel-inv-iff-right-unique-at* [*iff*]:
rel-injective-on $(P :: 'a \Rightarrow bool)\ (R^{-1} :: 'a \Rightarrow 'b \Rightarrow bool) \longleftrightarrow$ *right-unique-at* $P\ R$

⟨*proof*⟩

lemma *right-unique-on-rel-inv-iff-rel-injective-at* [*iff*]:
right-unique-on (P :: $'a \Rightarrow bool$) (R^{-1} :: $'a \Rightarrow 'b \Rightarrow bool$) $\longleftrightarrow$ *rel-injective-at* P R
⟨*proof*⟩

lemma *rel-injective-at-rel-inv-iff-right-unique-on* [*iff*]:
rel-injective-at (P :: $'b \Rightarrow bool$) (R^{-1} :: $'a \Rightarrow 'b \Rightarrow bool$) $\longleftrightarrow$ *right-unique-on* P R
⟨*proof*⟩

definition *right-unique* (R :: $'a \Rightarrow$ -) $\equiv$ *right-unique-on* ($\top$ :: $'a \Rightarrow bool$) R

lemma *right-unique-eq-right-unique-on*:
right-unique (R :: $'a \Rightarrow$ -) = *right-unique-on* ($\top$ :: $'a \Rightarrow bool$) R
⟨*proof*⟩

lemma *right-uniqueI* [*intro*]:
assumes $\bigwedge x\ y\ y'.\ R\ x\ y \Longrightarrow R\ x\ y' \Longrightarrow y = y'$
shows *right-unique* R
⟨*proof*⟩

lemma *right-uniqueD*:
assumes *right-unique* R
and $R\ x\ y$ $R\ x\ y'$
shows $y = y'$
⟨*proof*⟩

lemma *right-unique-eq-right-unique-at*:
right-unique (R :: $'a \Rightarrow 'b \Rightarrow bool$) = *right-unique-at* ($\top$ :: $'b \Rightarrow bool$) R
⟨*proof*⟩

lemma *right-unique-on-if-right-unique*:
fixes P :: $'a \Rightarrow bool$ **and** R :: $'a \Rightarrow$ -
assumes *right-unique* R
shows *right-unique-on* P R
⟨*proof*⟩

lemma *right-unique-at-if-right-unique*:
fixes P :: $'a \Rightarrow bool$ **and** R :: $'b \Rightarrow 'a \Rightarrow bool$
assumes *right-unique* R
shows *right-unique-at* P R
⟨*proof*⟩

lemma *right-unique-if-right-unique-on-in-dom*:
assumes *right-unique-on* (*in-dom* R) R
shows *right-unique* R

⟨*proof*⟩

lemma *right-unique-if-right-unique-at-in-codom*:
assumes *right-unique-at* (*in-codom* R) R
shows *right-unique* R
⟨*proof*⟩

corollary *right-unique-on-in-dom-iff-right-unique* [*iff*]:
right-unique-on (*in-dom* R) $R \longleftrightarrow$ *right-unique* R
⟨*proof*⟩

corollary *right-unique-at-in-codom-iff-right-unique* [*iff*]:
right-unique-at (*in-codom* R) $R \longleftrightarrow$ *right-unique* R
⟨*proof*⟩

lemma *right-unique-rel-inv-iff-rel-injective* [*iff*]:
right-unique $R^{-1} \longleftrightarrow$ *rel-injective* R
⟨*proof*⟩

lemma *rel-injective-rel-inv-iff-right-unique* [*iff*]:
rel-injective $R^{-1} \longleftrightarrow$ *right-unique* R
⟨*proof*⟩

Instantiatiats **lemma** *right-unique-eq*: *right-unique* (=)
⟨*proof*⟩

end

Surjective

theory *Binary-Relations-Surjective*
imports
Binary-Relations-Left-Total
HOL-Syntax-Bundles-Lattices
begin

consts *rel-surjective-at* :: $'a \Rightarrow ('b \Rightarrow 'c \Rightarrow bool) \Rightarrow bool$

overloading
rel-surjective-at-pred ≡ *rel-surjective-at* :: $('a \Rightarrow bool) \Rightarrow ('b \Rightarrow 'a \Rightarrow bool) \Rightarrow bool$
begin
definition *rel-surjective-at-pred* P $R \equiv \forall y.\ P\ y \longrightarrow$ *in-codom* R y
end

lemma *rel-surjective-atI* [*intro*]:
assumes $\bigwedge y.\ P\ y \Longrightarrow$ *in-codom* R y
shows *rel-surjective-at* P R

⟨*proof*⟩

lemma *rel-surjective-atE* [*elim*]:
assumes *rel-surjective-at P R*
and *P y*
obtains *x* **where** *R x y*
⟨*proof*⟩

lemma *in-codom-if-rel-surjective-at-on*:
assumes *rel-surjective-at P R*
and *P y*
shows *in-codom R y*
⟨*proof*⟩

lemma *rel-surjective-at-rel-inv-iff-left-total-on* [*iff*]:
rel-surjective-at (P :: $'a \Rightarrow$ *bool*) (R^{-1} :: $'b \Rightarrow 'a \Rightarrow$ *bool*) $\longleftrightarrow$ *left-total-on P R*
⟨*proof*⟩

lemma *left-total-on-rel-inv-iff-rel-surjective-at* [*iff*]:
left-total-on (P :: $'a \Rightarrow$ *bool*) (R^{-1} :: $'a \Rightarrow 'b \Rightarrow$ *bool*) $\longleftrightarrow$ *rel-surjective-at P R*
⟨*proof*⟩

definition *rel-surjective* (R :: - $\Rightarrow 'a \Rightarrow$ -) $\equiv$ *rel-surjective-at* ($\top$ :: $'a \Rightarrow$ *bool*) R

lemma *rel-surjective-eq-rel-surjective-at*:
rel-surjective (R :: - $\Rightarrow 'a \Rightarrow$ -) = *rel-surjective-at* ($\top$ :: $'a \Rightarrow$ *bool*) R
⟨*proof*⟩

lemma *rel-surjectiveI*:
assumes $\bigwedge y$. *in-codom R y*
shows *rel-surjective R*
⟨*proof*⟩

lemma *rel-surjectiveE*:
assumes *rel-surjective R*
obtains *x* **where** *R x y*
⟨*proof*⟩

lemma *in-codom-if-rel-surjective-at*:
assumes *rel-surjective R*
shows *in-codom R y*
⟨*proof*⟩

lemma *rel-surjective-rel-inv-iff-left-total* [*iff*]: *rel-surjective* R^{-1} $\longleftrightarrow$ *left-total R*
⟨*proof*⟩

lemma *left-total-rel-inv-iff-rel-surjective* [*iff*]: *left-total* R^{-1} $\longleftrightarrow$ *rel-surjective R*
⟨*proof*⟩

lemma *rel-surjective-at-if-surjective*:
fixes $P :: 'a \Rightarrow bool$ **and** $R :: - \Rightarrow 'a \Rightarrow -$
assumes *rel-surjective R*
shows *rel-surjective-at P R*
⟨*proof*⟩

end

1.3.1 Basic Properties

theory *Binary-Relation-Properties*
imports
Binary-Relations-Antisymmetric
Binary-Relations-Injective
Binary-Relations-Irreflexive
Binary-Relations-Left-Total
Binary-Relations-Reflexive
Binary-Relations-Right-Unique
Binary-Relations-Surjective
Binary-Relations-Symmetric
Binary-Relations-Transitive
begin

end

1.3.2 Preorders

theory *Preorders*
imports
Binary-Relations-Reflexive
Binary-Relations-Transitive
begin

definition *preorder-on P R* ≡ *reflexive-on P R* ∧ *transitive-on P R*

lemma *preorder-onI* [*intro*]:
assumes *reflexive-on P R*
and *transitive-on P R*
shows *preorder-on P R*
⟨*proof*⟩

lemma *preorder-onE* [*elim*]:
assumes *preorder-on P R*
obtains *reflexive-on P R transitive-on P R*
⟨*proof*⟩

lemma *reflexive-on-if-preorder-on*:
assumes *preorder-on P R*

shows *reflexive-on* P R
⟨*proof*⟩

lemma *transitive-on-if-preorder-on*:
assumes *preorder-on* P R
shows *transitive-on* P R
⟨*proof*⟩

lemma *transitive-if-preorder-on-in-field*:
assumes *preorder-on* (*in-field* R) R
shows *transitive* R
⟨*proof*⟩

corollary *preorder-on-in-fieldE* [*elim*]:
assumes *preorder-on* (*in-field* R) R
obtains *reflexive-on* (*in-field* R) R *transitive* R
⟨*proof*⟩

lemma *preorder-on-rel-inv-if-preorder-on* [*iff*]:
preorder-on P R^{-1} $\longleftrightarrow$ *preorder-on* (P :: $'a \Rightarrow$ *bool*) (R :: $'a \Rightarrow$ -)
⟨*proof*⟩

lemma *rel-if-all-rel-if-rel-if-reflexive-on*:
assumes *reflexive-on* P R
and $\bigwedge z.\ P\ z \Longrightarrow R\ x\ z \Longrightarrow R\ y\ z$
and $P\ x$
shows $R\ y\ x$
⟨*proof*⟩

lemma *rel-if-all-rel-if-rel-if-reflexive-on'*:
assumes *reflexive-on* P R
and $\bigwedge z.\ P\ z \Longrightarrow R\ z\ x \Longrightarrow R\ z\ y$
and $P\ x$
shows $R\ x\ y$
⟨*proof*⟩

definition *preorder* (R :: $'a \Rightarrow$ -) $\equiv$ *preorder-on* ($\top$ :: $'a \Rightarrow$ *bool*) R

lemma *preorder-eq-preorder-on*:
preorder (R :: $'a \Rightarrow$ -) = *preorder-on* ($\top$ :: $'a \Rightarrow$ *bool*) R
⟨*proof*⟩

lemma *preorderI* [*intro*]:
assumes *reflexive* R
and *transitive* R
shows *preorder* R
⟨*proof*⟩

lemma *preorderE* [*elim*]:

assumes *preorder R*
obtains *reflexive R transitive R*
⟨*proof*⟩

lemma *preorder-on-if-preorder*:
fixes $P :: 'a \Rightarrow bool$ **and** $R :: 'a \Rightarrow$ -
assumes *preorder R*
shows *preorder-on P R*
⟨*proof*⟩

Instantiations **lemma** *preorder-eq*: *preorder* (=)
⟨*proof*⟩

end

1.3.3 Partial Equivalence Relations

theory *Partial-Equivalence-Relations*
imports
Binary-Relations-Symmetric
Preorders
begin

definition *partial-equivalence-rel-on P R* ≡ *transitive-on P R* ∧ *symmetric-on P R*

lemma *partial-equivalence-rel-onI* [*intro*]:
assumes *transitive-on P R*
and *symmetric-on P R*
shows *partial-equivalence-rel-on P R*
⟨*proof*⟩

lemma *partial-equivalence-rel-onE* [*elim*]:
assumes *partial-equivalence-rel-on P R*
obtains *transitive-on P R symmetric-on P R*
⟨*proof*⟩

lemma *partial-equivalence-rel-on-rel-self-if-rel-dom*:
assumes *partial-equivalence-rel-on* $(P :: 'a \Rightarrow bool)$ $(R :: 'a \Rightarrow 'a \Rightarrow bool)$
and *P x P y*
and *R x y*
shows *R x x*
⟨*proof*⟩

lemma *partial-equivalence-rel-on-rel-self-if-rel-codom*:
assumes *partial-equivalence-rel-on* $(P :: 'a \Rightarrow bool)$ $(R :: 'a \Rightarrow 'a \Rightarrow bool)$
and *P x P y*
and *R x y*

shows $R\ y\ y$
⟨*proof*⟩

lemma *partial-equivalence-rel-on-rel-inv-iff-partial-equivalence-rel-on* [*iff*]:
partial-equivalence-rel-on $P\ R^{-1} \longleftrightarrow$ *partial-equivalence-rel-on* $(P :: 'a \Rightarrow bool)$ $(R :: 'a \Rightarrow -)$
⟨*proof*⟩

definition *partial-equivalence-rel* $(R :: 'a \Rightarrow -) \equiv$ *partial-equivalence-rel-on* $(\top :: 'a \Rightarrow bool)\ R$

lemma *partial-equivalence-rel-eq-partial-equivalence-rel-on*:
partial-equivalence-rel $(R :: 'a \Rightarrow -) =$ *partial-equivalence-rel-on* $(\top :: 'a \Rightarrow bool)$ R
⟨*proof*⟩

lemma *partial-equivalence-relI* [*intro*]:
assumes *transitive* R
and *symmetric* R
shows *partial-equivalence-rel* R
⟨*proof*⟩

lemma *reflexive-on-in-field-if-partial-equivalence-rel*:
assumes *partial-equivalence-rel* R
shows *reflexive-on* (*in-field* R) R
⟨*proof*⟩

lemma *partial-equivalence-relE* [*elim*]:
assumes *partial-equivalence-rel* R
obtains *preorder-on* (*in-field* R) R *symmetric* R
⟨*proof*⟩

lemma *partial-equivalence-rel-on-if-partial-equivalence-rel*:
fixes $P :: 'a \Rightarrow bool$ **and** $R :: 'a \Rightarrow -$
assumes *partial-equivalence-rel* R
shows *partial-equivalence-rel-on* $P\ R$
⟨*proof*⟩

lemma *partial-equivalence-rel-rel-inv-iff-partial-equivalence-rel* [*iff*]:
partial-equivalence-rel $R^{-1} \longleftrightarrow$ *partial-equivalence-rel* R
⟨*proof*⟩

corollary *in-codom-eq-in-dom-if-partial-equivalence-rel*:
assumes *partial-equivalence-rel* R
shows *in-codom* $R =$ *in-dom* R
⟨*proof*⟩

lemma *partial-equivalence-rel-rel-comp-self-eq-self*:
assumes *partial-equivalence-rel* R

shows $(R \circ\circ R) = R$
⟨*proof*⟩

lemma *partial-equivalence-rel-if-partial-equivalence-rel-on-in-field*:
assumes *partial-equivalence-rel-on* (*in-field R*) *R*
shows *partial-equivalence-rel R*
⟨*proof*⟩

corollary *partial-equivalence-rel-on-in-field-iff-partial-equivalence-rel* [*iff*]:
partial-equivalence-rel-on (*in-field R*) *R* ⟷ *partial-equivalence-rel R*
⟨*proof*⟩

Instantiations **lemma** *partial-equivalence-rel-eq*: *partial-equivalence-rel* (=)
⟨*proof*⟩

lemma *partial-equivalence-rel-top*: *partial-equivalence-rel* ⊤
⟨*proof*⟩

end

1.3.4 Equivalences

theory *Equivalence-Relations*
imports
Partial-Equivalence-Relations
begin

definition *equivalence-rel-on P R* ≡
partial-equivalence-rel-on P R ∧ *reflexive-on P R*

lemma *equivalence-rel-onI* [*intro*]:
assumes *partial-equivalence-rel-on P R*
and *reflexive-on P R*
shows *equivalence-rel-on P R*
⟨*proof*⟩

lemma *equivalence-rel-onE* [*elim*]:
assumes *equivalence-rel-on P R*
obtains *partial-equivalence-rel-on P R reflexive-on P R*
⟨*proof*⟩

lemma *equivalence-rel-on-in-field-if-partial-equivalence-rel*:
assumes *partial-equivalence-rel R*
shows *equivalence-rel-on* (*in-field R*) *R*
⟨*proof*⟩

corollary *partial-equivalence-rel-iff-equivalence-rel-on-in-field*:
partial-equivalence-rel R ⟷ *equivalence-rel-on* (*in-field R*) *R*

⟨*proof*⟩

definition *equivalence-rel* (R :: $'a \Rightarrow$ -) ≡ *equivalence-rel-on* (⊤ :: $'a \Rightarrow$ *bool*) R

lemma *equivalence-rel-eq-equivalence-rel-on*:
equivalence-rel (R :: $'a \Rightarrow$ -) = *equivalence-rel-on* (⊤ :: $'a \Rightarrow$ *bool*) R
⟨*proof*⟩

lemma *equivalence-relI* [*intro*]:
assumes *partial-equivalence-rel* R
and *reflexive* R
shows *equivalence-rel* R
⟨*proof*⟩

lemma *equivalence-relE* [*elim*]:
assumes *equivalence-rel* R
obtains *partial-equivalence-rel* R *reflexive* R
⟨*proof*⟩

lemma *equivalence-rel-on-if-equivalence*:
fixes P :: $'a \Rightarrow$ *bool* **and** R :: $'a \Rightarrow$ -
assumes *equivalence-rel* R
shows *equivalence-rel-on* P R
⟨*proof*⟩

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

lemma *equivalence-top*: *equivalence-rel* ⊤
⟨*proof*⟩

end

1.3.5 Partial Orders

theory *Partial-Orders*
imports
Binary-Relations-Antisymmetric
Preorders
begin

definition *partial-order-on* P R ≡ *preorder-on* P R ∧ *antisymmetric-on* P R

lemma *partial-order-onI* [*intro*]:
assumes *preorder-on* P R
and *antisymmetric-on* P R
shows *partial-order-on* P R
⟨*proof*⟩

lemma *partial-order-onE* [*elim*]:
  assumes *partial-order-on P R*
  obtains *preorder-on P R antisymmetric-on P R*
  ⟨*proof*⟩

lemma *transitive-if-partial-order-on-in-field*:
  assumes *partial-order-on* (*in-field R*) *R*
  shows *transitive R*
  ⟨*proof*⟩

lemma *antisymmetric-if-partial-order-on-in-field*:
  assumes *partial-order-on* (*in-field R*) *R*
  shows *antisymmetric R*
  ⟨*proof*⟩

definition *partial-order* (R :: $'a \Rightarrow$ -) $\equiv$ *partial-order-on* ($\top$ :: $'a \Rightarrow$ *bool*) R

lemma *partial-order-eq-partial-order-on*:
  partial-order (R :: $'a \Rightarrow$ -) = *partial-order-on* ($\top$ :: $'a \Rightarrow$ *bool*) R
  ⟨*proof*⟩

lemma *partial-orderI* [*intro*]:
  assumes *preorder R*
  and *antisymmetric R*
  shows *partial-order R*
  ⟨*proof*⟩

lemma *partial-orderE* [*elim*]:
  assumes *partial-order R*
  obtains *preorder R antisymmetric R*
  ⟨*proof*⟩

lemma *partial-order-on-if-partial-order*:
  fixes P :: $'a \Rightarrow$ *bool* **and** R :: $'a \Rightarrow$ -
  assumes *partial-order R*
  shows *partial-order-on P R*
  ⟨*proof*⟩

end

1.3.6 Restricted Equality

theory *Restricted-Equality*
  imports
    Binary-Relations-Order-Base
    Binary-Relation-Functions
    Equivalence-Relations

Partial-Orders
begin

Summary Introduces the concept of restricted equalities. An equality (=) can be restricted to only apply to a subset of its elements. The restriction can be formulated, for example, by a predicate or a set.

consts *eq-restrict* :: $'a \Rightarrow 'b \Rightarrow 'b \Rightarrow bool$

bundle *eq-restrict-syntax*
begin
syntax
-eq-restrict :: $'a \Rightarrow ('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a \Rightarrow bool$ ((-) =(_) (-) [*51*,*51*,*51*] *50*)
notation *eq-restrict* ('(=(_)'))
end
bundle *no-eq-restrict-syntax*
begin
no-syntax
-eq-restrict :: $'a \Rightarrow ('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a \Rightarrow bool$ ((-) =(_) (-) [*51*,*51*,*51*] *50*)
no-notation *eq-restrict* ('(=(_)'))
end
unbundle *eq-restrict-syntax*

translations
$x =_P y \rightleftharpoons$ *CONST eq-restrict* $P\ x\ y$

overloading
eq-restrict-pred $\equiv$ *eq-restrict* :: $('a \Rightarrow bool) \Rightarrow 'a \Rightarrow 'a \Rightarrow bool$
begin
definition *eq-restrict-pred* $(P :: 'a \Rightarrow bool) \equiv ((=) :: 'a \Rightarrow \text{-})\restriction_P$
end

lemma *eq-restrict-eq-eq-restrict-left*: $((=_{P\ ::\ 'a \Rightarrow bool}) :: 'a \Rightarrow \text{-}) = (=)\restriction_P$
⟨*proof*⟩

lemma *eq-restrictI* [*intro*]:
assumes $x = y$
and $P\ x$
shows $x =_P y$
⟨*proof*⟩

lemma *eq-restrictE* [*elim*]:
assumes $x =_P y$
obtains $P\ x\ y = x$
⟨*proof*⟩

lemma *eq-restrict-iff*: $x =_P y \longleftrightarrow y = x \wedge P\ x$ ⟨*proof*⟩

lemma *eq-restrict-le-eq*: $((=_{P\ ::\ 'a \Rightarrow bool}) :: 'a \Rightarrow \text{-}) \leq (=)$

⟨*proof*⟩

lemma *eq-restrict-top-eq-eq* [*simp*]: $(=_{\top :: 'a \Rightarrow bool}) = ((=) :: 'a \Rightarrow -)$
⟨*proof*⟩

lemma *in-dom-eq-restrict-eq* [*simp*]: *in-dom* $(=_P) = P$ ⟨*proof*⟩
lemma *in-codom-eq-restrict-eq* [*simp*]: *in-codom* $(=_P) = P$ ⟨*proof*⟩
lemma *in-field-eq-restrict-eq* [*simp*]: *in-field* $(=_P) = P$ ⟨*proof*⟩

Order Properties **context**
fixes $P :: 'a \Rightarrow bool$
begin

context
begin
lemma *reflexive-on-eq-restrict*: *reflexive-on* P $((=_P) :: 'a \Rightarrow -)$ ⟨*proof*⟩
lemma *transitive-eq-restrict*: *transitive* $((=_P) :: 'a \Rightarrow -)$ ⟨*proof*⟩
lemma *symmetric-eq-restrict*: *symmetric* $((=_P) :: 'a \Rightarrow -)$ ⟨*proof*⟩
lemma *antisymmetric-eq-restrict*: *antisymmetric* $((=_P) :: 'a \Rightarrow -)$ ⟨*proof*⟩
end

context
begin
lemma *preorder-on-eq-restrict*: *preorder-on* P $((=_P) :: 'a \Rightarrow -)$
⟨*proof*⟩
lemma *partial-equivalence-rel-eq-restrict*: *partial-equivalence-rel* $((=_P) :: 'a \Rightarrow -)$
⟨*proof*⟩
end

lemma *partial-order-on-eq-restrict*: *partial-order-on* P $((=_P) :: 'a \Rightarrow -)$
⟨*proof*⟩
lemma *equivalence-rel-on-eq-restrict*: *equivalence-rel-on* P $((=_P) :: 'a \Rightarrow -)$
⟨*proof*⟩
end

end

theory *LBinary-Relations*
imports
Binary-Relation-Functions
Binary-Relations-Lattice
Binary-Relations-Order
Binary-Relation-Properties
Restricted-Equality
begin

Summary Basic concepts on binary relations.

end

Injective

theory *Functions-Injective*
  imports
    Functions-Base
    HOL-Syntax-Bundles-Lattices
begin

consts *injective-on* :: $'a \Rightarrow ('b \Rightarrow 'c) \Rightarrow bool$

overloading
  injective-on-pred $\equiv$ *injective-on* :: $('a \Rightarrow bool) \Rightarrow ('a \Rightarrow 'b) \Rightarrow bool$
begin
  definition *injective-on-pred* $P\ f \equiv \forall\, x\ x'.\ P\ x \longrightarrow P\ x' \longrightarrow f\ x = f\ x' \longrightarrow x =$
x'
end

lemma *injective-onI* [*intro*]:
  assumes $\bigwedge x\ x'.\ P\ x \Longrightarrow P\ x' \Longrightarrow f\ x = f\ x' \Longrightarrow x = x'$
  shows *injective-on* $P\ f$
  ⟨*proof*⟩

lemma *injective-onD*:
  assumes *injective-on* $P\ f$
  and $P\ x\ P\ x'$
  and $f\ x = f\ x'$
  shows $x = x'$
  ⟨*proof*⟩

definition *injective* $(f :: 'a \Rightarrow \text{-}) \equiv$ *injective-on* $(\top :: 'a \Rightarrow bool)\ f$

lemma *injective-eq-injective-on*:
  injective $(f :: 'a \Rightarrow \text{-}) =$ *injective-on* $(\top :: 'a \Rightarrow bool)\ f$
  ⟨*proof*⟩

lemma *injectiveI* [*intro*]:
  assumes $\bigwedge x\ x'.\ f\ x = f\ x' \Longrightarrow x = x'$
  shows *injective* f
  ⟨*proof*⟩

lemma *injectiveD*:
  assumes *injective* f
  and $f\ x = f\ x'$
  shows $x = x'$
  ⟨*proof*⟩

lemma *injective-on-if-injective*:
  fixes $P :: 'a \Rightarrow bool$ **and** $f :: 'a \Rightarrow \text{-}$
  assumes *injective* f
  shows *injective-on* $P\ f$

⟨*proof*⟩

Instantiations **lemma** *injective-id*: *injective id* ⟨*proof*⟩

end

Inverse

theory *Functions-Inverse*
imports
Functions-Injective
begin

consts *inverse-on* :: $'a \Rightarrow ('b \Rightarrow 'c) \Rightarrow ('c \Rightarrow 'b) \Rightarrow bool$

overloading
inverse-on-pred ≡ *inverse-on* :: $('a \Rightarrow bool) \Rightarrow ('a \Rightarrow 'b) \Rightarrow ('b \Rightarrow 'a) \Rightarrow bool$
begin
definition *inverse-on-pred* $P\ f\ g \equiv \forall x.\ P\ x \longrightarrow g\ (f\ x) = x$
end

lemma *inverse-onI* [*intro*]:
assumes $\bigwedge x.\ P\ x \Longrightarrow g\ (f\ x) = x$
shows *inverse-on* $P\ f\ g$
⟨*proof*⟩

lemma *inverse-onD*:
assumes *inverse-on* $P\ f\ g$
and $P\ x$
shows $g\ (f\ x) = x$
⟨*proof*⟩

lemma *injective-on-if-inverse-on*:
assumes *inv*: *inverse-on* $(P :: 'a \Rightarrow bool)\ (f :: 'a \Rightarrow -)\ g$
shows *injective-on* $P\ f$
⟨*proof*⟩

definition *inverse* $(f :: 'a \Rightarrow -) \equiv$ *inverse-on* $(\top :: 'a \Rightarrow bool)\ f$

lemma *inverse-eq-inverse-on*:
inverse $(f :: 'a \Rightarrow -) =$ *inverse-on* $(\top :: 'a \Rightarrow bool)\ f$
⟨*proof*⟩

lemma *inverseI* [*intro*]:
assumes $\bigwedge x.\ g\ (f\ x) = x$
shows *inverse* $f\ g$
⟨*proof*⟩

lemma *inverseD*:
assumes *inverse f g*
shows $g\ (f\ x) = x$
⟨*proof*⟩

lemma *inverse-on-if-inverse*:
fixes $P :: 'a \Rightarrow bool$ **and** $f :: 'a \Rightarrow 'b$
assumes *inverse f g*
shows *inverse-on P f g*
⟨*proof*⟩

end

Bijection

theory *Functions-Bijection*
imports
Functions-Inverse
Functions-Monotone
begin

consts *bijection-on* $:: 'a \Rightarrow 'b \Rightarrow ('c \Rightarrow 'd) \Rightarrow ('d \Rightarrow 'c) \Rightarrow bool$

overloading
bijection-on-pred ≡ *bijection-on* $:: ('a \Rightarrow bool) \Rightarrow ('b \Rightarrow bool) \Rightarrow ('a \Rightarrow 'b) \Rightarrow ('b \Rightarrow 'a) \Rightarrow bool$
begin
definition *bijection-on-pred* $P\ P'\ f\ g \equiv$
$([P] \Rrightarrow_m P')\ f \land$
$([P'] \Rrightarrow_m P)\ g \land$
inverse-on $P\ f\ g \land$
inverse-on $P'\ g\ f$
end

lemma *bijection-onI* [*intro*]:
assumes $([P] \Rrightarrow_m P')\ f$
and $([P'] \Rrightarrow_m P)\ g$
and *inverse-on* $P\ f\ g$
and *inverse-on* $P'\ g\ f$
shows *bijection-on* $P\ P'\ f\ g$
⟨*proof*⟩

lemma *bijection-onE*:
assumes *bijection-on* $P\ P'\ f\ g$
obtains $([P] \Rrightarrow_m P')\ f$ $([P'] \Rrightarrow_m P)\ g$
inverse-on $P\ f\ g$ *inverse-on* $P'\ g\ f$
⟨*proof*⟩

context
fixes P :: $'a \Rightarrow$ *bool*
and P' :: $'b \Rightarrow$ *bool*
and f :: $'a \Rightarrow\ 'b$
begin

lemma *mono-wrt-pred-if-bijection-on-left*:
assumes *bijection-on* P P' f g
shows $([P] \Rrightarrow_m P')$ f
⟨*proof*⟩

lemma *mono-wrt-pred-if-bijection-on-right*:
assumes *bijection-on* P P' f g
shows $([P'] \Rrightarrow_m P)$ g
⟨*proof*⟩

lemma *bijection-on-pred-right*:
assumes *bijection-on* P P' f g
and P x
shows P' $(f\ x)$
⟨*proof*⟩

lemma *bijection-on-pred-left*:
assumes *bijection-on* P P' f g
and P' y
shows P $(g\ y)$
⟨*proof*⟩

lemma *inverse-on-if-bijection-on-left-right*:
assumes *bijection-on* P P' f g
shows *inverse-on* P f g
⟨*proof*⟩

lemma *inverse-on-if-bijection-on-right-left*:
assumes *bijection-on* P P' f g
shows *inverse-on* P' g f
⟨*proof*⟩

lemma *bijection-on-left-right-eq-self*:
assumes *bijection-on* P P' f g
and P x
shows $g\ (f\ x) = x$
⟨*proof*⟩

lemma *bijection-on-right-left-eq-self'*:
assumes *bijection-on* P P' f g
and P' y
shows $f\ (g\ y) = y$
⟨*proof*⟩

lemma *bijection-on-right-left-if-bijection-on-left-right*:
assumes *bijection-on* P P' f g
shows *bijection-on* P' P g f
⟨*proof*⟩

lemma *injective-on-if-bijection-on-left*:
assumes *bijection-on* P P' f g
shows *injective-on* P f
⟨*proof*⟩

lemma *injective-on-if-bijection-on-right*:
assumes *bijection-on* P P' f g
shows *injective-on* P' g
⟨*proof*⟩

end

definition *bijection* $(f :: {'a} \Rightarrow {'b}) \equiv$ *bijection-on* $(\top :: {'a} \Rightarrow bool)$ $(\top :: {'b} \Rightarrow bool)$ f

lemma *bijection-eq-bijection-on*:
bijection $(f :: {'a} \Rightarrow {'b}) =$ *bijection-on* $(\top :: {'a} \Rightarrow bool)$ $(\top :: {'b} \Rightarrow bool)$ f
⟨*proof*⟩

lemma *bijectionI* [*intro*]:
assumes *inverse* f g
and *inverse* g f
shows *bijection* f g
⟨*proof*⟩

lemma *bijectionE* [*elim*]:
assumes *bijection* f g
obtains *inverse* f g *inverse* g f
⟨*proof*⟩

lemma *inverse-if-bijection-left-right*:
assumes *bijection* f g
shows *inverse* f g
⟨*proof*⟩

lemma *inverse-if-bijection-right-left*:
assumes *bijection* f g
shows *inverse* g f
⟨*proof*⟩

lemma *bijection-right-left-if-bijection-left-right*:
assumes *bijection* f g

shows *bijection g f*
⟨*proof*⟩

Instantiations **lemma** *bijection-on-self-id*:
fixes P :: $'a \Rightarrow bool$
shows *bijection-on* P P (*id* :: $'a \Rightarrow$ -) *id*
⟨*proof*⟩

end

Surjective

theory *Functions-Surjective*
imports
HOL-Syntax-Bundles-Lattices
begin

consts *surjective-at* :: $'a \Rightarrow ('b \Rightarrow 'c) \Rightarrow bool$

overloading
surjective-at-pred ≡ *surjective-at* :: $('a \Rightarrow bool) \Rightarrow ('b \Rightarrow 'a) \Rightarrow bool$
begin
definition *surjective-at-pred* $P\ f \equiv \forall y.\ P\ y \longrightarrow (\exists x.\ y = f\ x)$
end

lemma *surjective-atI* [*intro*]:
assumes $\bigwedge y.\ P\ y \Longrightarrow \exists x.\ y = f\ x$
shows *surjective-at* $P\ f$
⟨*proof*⟩

lemma *surjective-atE* [*elim*]:
assumes *surjective-at* $P\ f$
and $P\ y$
obtains x **where** $y = f\ x$
⟨*proof*⟩

definition *surjective* $(f :: - \Rightarrow 'a) \equiv$ *surjective-at* $(\top :: 'a \Rightarrow bool)\ f$

lemma *surjective-eq-surjective-at*:
surjective $(f :: - \Rightarrow 'a) =$ *surjective-at* $(\top :: 'a \Rightarrow bool)\ f$
⟨*proof*⟩

lemma *surjectiveI* [*intro*]:
assumes $\bigwedge y.\ \exists x.\ y = f\ x$
shows *surjective* f
⟨*proof*⟩

lemma *surjectiveE*:

assumes *surjective f*
obtains *x* **where** $y = f\ x$
⟨*proof*⟩

lemma *surjective-at-if-surjective*:
fixes $P :: 'a \Rightarrow bool$ **and** $f :: - \Rightarrow 'a$
assumes *surjective f*
shows *surjective-at P f*
⟨*proof*⟩

end

1.3.7 Basic Properties

theory *Function-Properties*
imports
Functions-Bijection
Functions-Injective
Functions-Inverse
Functions-Monotone
Functions-Surjective
begin

Summary Basic properties on functions.

end

theory *LFunctions*
imports
Functions-Base
Function-Properties
Function-Relators
begin

Summary Basic concepts on functions.

end

1.3.8 Functions On Orders

Basics

theory *Order-Functions-Base*
imports
Functions-Monotone
Restricted-Equality
begin

Bi-Relation **definition** *bi-related R x y* ≡ *R x y* ∧ *R y x*

bundle *bi-related-syntax* **begin**
syntax
-bi-related :: *'a* ⇒ (*'a* ⇒ *'a* ⇒ *bool*) ⇒ *'a* ⇒ *bool* ((-) $\equiv_{(\text{-})}$ (-) [*51*,*51*,*51*] *50*)
notation *bi-related* ('(≡(-)'))
end
bundle *no-bi-related-syntax* **begin**
no-syntax
-bi-related :: *'a* ⇒ (*'a* ⇒ *'a* ⇒ *bool*) ⇒ *'a* ⇒ *bool* ((-) $\equiv_{(\text{-})}$ (-) [*51*,*51*,*51*] *50*)
no-notation *bi-related* ('(≡(-)'))
end
unbundle *bi-related-syntax*
translations
$x \equiv_R y$ ⇌ *CONST bi-related R x y*

lemma *bi-relatedI* [*intro*]:
assumes *R x y*
and *R y x*
shows $x \equiv_R y$
⟨*proof*⟩

lemma *bi-relatedE* [*elim*]:
assumes $x \equiv_R y$
obtains *R x y R y x*
⟨*proof*⟩

lemma *symmetric-bi-related* [*iff*]: *symmetric* ($\equiv_R$)
⟨*proof*⟩

lemma *reflexive-bi-related-if-reflexive* [*intro*]:
assumes *reflexive R*
shows *reflexive* ($\equiv_R$)
⟨*proof*⟩

lemma *transitive-bi-related-if-transitive* [*intro*]:
assumes *transitive R*
shows *transitive* ($\equiv_R$)
⟨*proof*⟩

lemma *mono-bi-related* [*iff*]: *mono bi-related*
⟨*proof*⟩

lemma *bi-related-if-le-rel-if-bi-related*:
assumes $x \equiv_R y$
and $R \le S$
shows $x \equiv_S y$
⟨*proof*⟩

lemma *eq-if-bi-related-if-antisymmetric-on*:
assumes *antisymmetric-on* P R
and $x \equiv_R y$
and $P\ x\ P\ y$
shows $x = y$
⟨*proof*⟩

lemma *eq-if-bi-related-if-in-field-le-if-antisymmetric-on*:
assumes *antisymmetric-on* P R
and *in-field* $R \leq P$
and $x \equiv_R y$
shows $x = y$
⟨*proof*⟩

lemma *bi-related-le-eq-if-antisymmetric-on-in-field*:
assumes *antisymmetric-on* (*in-field* R) R
shows $(\equiv_R) \leq (=)$
⟨*proof*⟩

lemma *bi-related-if-all-rel-iff-if-reflexive-on*:
assumes *reflexive-on* P R
and $\bigwedge z.\ P\ z \Longrightarrow R\ x\ z \longleftrightarrow R\ y\ z$
and $P\ x\ P\ y$
shows $x \equiv_R y$
⟨*proof*⟩

lemma *bi-related-if-all-rel-iff-if-reflexive-on′*:
assumes *reflexive-on* P R
and $\bigwedge z.\ P\ z \Longrightarrow R\ z\ x \longleftrightarrow R\ z\ y$
and $P\ x\ P\ y$
shows $x \equiv_R y$
⟨*proof*⟩

corollary *eq-if-all-rel-iff-if-antisymmetric-on-if-reflexive-on*:
assumes *reflexive-on* P R **and** *antisymmetric-on* P R
and $\bigwedge z.\ P\ z \Longrightarrow R\ x\ z \longleftrightarrow R\ y\ z$
and $P\ x\ P\ y$
shows $x = y$
⟨*proof*⟩

corollary *eq-if-all-rel-iff-if-antisymmetric-on-if-reflexive-on′*:
assumes *reflexive-on* P R **and** *antisymmetric-on* P R
and $\bigwedge z.\ P\ z \Longrightarrow R\ z\ x \longleftrightarrow R\ z\ y$
and $P\ x\ P\ y$
shows $x = y$
⟨*proof*⟩

Inflationary **consts** *inflationary-on* :: $'a \Rightarrow ('b \Rightarrow 'b \Rightarrow bool) \Rightarrow ('b \Rightarrow 'b) \Rightarrow bool$

overloading
inflationary-on-pred ≡ *inflationary-on* ::
$('a \Rightarrow bool) \Rightarrow ('a \Rightarrow 'a \Rightarrow bool) \Rightarrow ('a \Rightarrow 'a) \Rightarrow bool$
begin

Often also called "extensive".

definition *inflationary-on-pred* P $(R :: 'a \Rightarrow 'a \Rightarrow -)$ $f \equiv \forall x.\ P\ x \longrightarrow R\ x\ (f\ x)$
end

lemma *inflationary-onI* [*intro*]:
assumes $\bigwedge x.\ P\ x \Longrightarrow R\ x\ (f\ x)$
shows *inflationary-on* $P\ R\ f$
⟨*proof*⟩

lemma *inflationary-onD* [*dest*]:
assumes *inflationary-on* $P\ R\ f$
and $P\ x$
shows $R\ x\ (f\ x)$
⟨*proof*⟩

lemma *inflationary-on-eq-dep-mono-wrt-pred*: *inflationary-on* = *dep-mono-wrt-pred*
⟨*proof*⟩

lemma *antimono-inflationary-on-pred* [*iff*]:
antimono $(\lambda(P :: 'a \Rightarrow bool).$ *inflationary-on* $P\ (R :: 'a \Rightarrow -))$
⟨*proof*⟩

lemma *inflationary-on-if-le-pred-if-inflationary-on*:
fixes $P\ P' :: 'a \Rightarrow bool$ **and** $R :: 'a \Rightarrow -$
assumes *inflationary-on* $P\ R\ f$
and $P' \leq P$
shows *inflationary-on* $P'\ R\ f$
⟨*proof*⟩

lemma *mono-inflationary-on-rel* [*iff*]:
mono $(\lambda(R :: 'a \Rightarrow -).$ *inflationary-on* $(P :: 'a \Rightarrow bool)\ R)$
⟨*proof*⟩

lemma *inflationary-on-if-le-rel-if-inflationary-on*:
assumes *inflationary-on* $P\ R\ f$
and $\bigwedge x.\ P\ x \Longrightarrow R\ x\ (f\ x) \Longrightarrow R'\ x\ (f\ x)$
shows *inflationary-on* $P\ R'\ f$
⟨*proof*⟩

lemma *le-in-dom-if-inflationary-on*:
assumes *inflationary-on* $P\ R\ f$

shows $P \leq$ *in-dom* R
⟨*proof*⟩

lemma *inflationary-on-sup-eq* [*simp*]:
(*inflationary-on* :: ($'a \Rightarrow bool$) $\Rightarrow$ ($'a \Rightarrow$ -) $\Rightarrow$ -) ((P :: $'a \Rightarrow bool$) $\sqcup$ Q)
= *inflationary-on* $P \sqcap$ *inflationary-on* Q
⟨*proof*⟩

definition *inflationary* (R :: $'a \Rightarrow$ -) $f \equiv$ *inflationary-on* ($\top$ :: $'a \Rightarrow bool$) R f

lemma *inflationary-eq-inflationary-on*:
inflationary (R :: $'a \Rightarrow$ -) f = *inflationary-on* ($\top$ :: $'a \Rightarrow bool$) R f
⟨*proof*⟩

lemma *inflationaryI* [*intro*]:
assumes $\bigwedge x.\ R\ x\ (f\ x)$
shows *inflationary* R f
⟨*proof*⟩

lemma *inflationaryD*:
assumes *inflationary* R f
shows $R\ x\ (f\ x)$
⟨*proof*⟩

lemma *inflationary-on-if-inflationary*:
fixes P :: $'a \Rightarrow bool$ **and** R :: $'a \Rightarrow$ -
assumes *inflationary* R f
shows *inflationary-on* P R f
⟨*proof*⟩

lemma *inflationary-eq-dep-mono-wrt-pred*: *inflationary* = *dep-mono-wrt-pred* $\top$
⟨*proof*⟩

Deflationary **definition** *deflationary-on* P $R \equiv$ *inflationary-on* P R^{-1}

lemma *deflationary-on-eq-inflationary-on-rel-inv*:
deflationary-on P R = *inflationary-on* P R^{-1}
⟨*proof*⟩

declare *deflationary-on-eq-inflationary-on-rel-inv*[*symmetric*, *simp*]

corollary *deflationary-on-rel-inv-eq-inflationary-on* [*simp*]:
deflationary-on P R^{-1} = *inflationary-on* P R
⟨*proof*⟩

lemma *deflationary-onI* [*intro*]:
assumes $\bigwedge x.\ P\ x \Longrightarrow R\ (f\ x)\ x$
shows *deflationary-on* P R f

⟨*proof*⟩

lemma *deflationary-onD* [*dest*]:
assumes *deflationary-on P R f*
and *P x*
shows *R* (*f x*) *x*
⟨*proof*⟩

lemma *deflationary-on-eq-dep-mono-wrt-pred-rel-inv*:
deflationary-on P R = ([*x* :: *P*] $\Rrightarrow_m$ R^{-1} *x*)
⟨*proof*⟩

lemma *antimono-deflationary-on-pred* [*iff*]:
antimono (λ(*P* :: ′*a* ⇒ *bool*). *deflationary-on P* (*R* :: ′*a* ⇒ -))
⟨*proof*⟩

lemma *deflationary-on-if-le-pred-if-deflationary-on*:
fixes *P P*′ :: ′*a* ⇒ *bool* **and** *R* :: ′*a* ⇒ -
assumes *deflationary-on P R f*
and *P*′ ≤ *P*
shows *deflationary-on P*′ *R f*
⟨*proof*⟩

lemma *mono-deflationary-on-rel* [*iff*]:
mono (λ(*R* :: ′*a* ⇒ -). *deflationary-on* (*P* :: ′*a* ⇒ *bool*) *R*)
⟨*proof*⟩

lemma *deflationary-on-if-le-rel-if-deflationary-on*:
assumes *deflationary-on P R f*
and ⋀*x*. *P x* ⟹ *R* (*f x*) *x* ⟹ *R*′ (*f x*) *x*
shows *deflationary-on P R*′ *f*
⟨*proof*⟩

lemma *le-in-dom-if-deflationary-on*:
assumes *deflationary-on P R f*
shows *P* ≤ *in-codom R*
⟨*proof*⟩

lemma *deflationary-on-sup-eq* [*simp*]:
(*deflationary-on* :: (′*a* ⇒ *bool*) ⇒ (′*a* ⇒ -) ⇒ -) ((*P* :: ′*a* ⇒ *bool*) ⊔ *Q*)
= *deflationary-on P* ⊓ *deflationary-on Q*
⟨*proof*⟩

definition *deflationary R* (*f* :: ′*a* ⇒ -) ≡ *deflationary-on* (⊤ :: ′*a* ⇒ *bool*) *R f*

lemma *deflationary-eq-deflationary-on*:
deflationary R (*f* :: ′*a* ⇒ -) = *deflationary-on* (⊤ :: ′*a* ⇒ *bool*) *R f*
⟨*proof*⟩

lemma *deflationaryI* [*intro*]:
  assumes $\bigwedge x.\ R\ (f\ x)\ x$
  shows *deflationary* $R\ f$
  ⟨*proof*⟩

lemma *deflationaryD*:
  assumes *deflationary* $R\ f$
  shows $R\ (f\ x)\ x$
  ⟨*proof*⟩

lemma *deflationary-on-if-deflationary*:
  fixes $P :: 'a \Rightarrow$ *bool* **and** $f :: 'a \Rightarrow$ -
  assumes *deflationary* $R\ f$
  shows *deflationary-on* $P\ R\ f$
  ⟨*proof*⟩

lemma *deflationary-eq-dep-mono-wrt-pred-rel-inv*:
  deflationary R = *dep-mono-wrt-pred* $\top\ R^{-1}$
  ⟨*proof*⟩

Relational Equivalence **definition** *rel-equivalence-on* ≡ *inflationary-on* ⊓ *deflationary-on*

lemma *rel-equivalence-on-eq*:
  rel-equivalence-on = *inflationary-on* ⊓ *deflationary-on*
  ⟨*proof*⟩

lemma *rel-equivalence-onI* [*intro*]:
  assumes *inflationary-on* $P\ R\ f$
  and *deflationary-on* $P\ R\ f$
  shows *rel-equivalence-on* $P\ R\ f$
  ⟨*proof*⟩

lemma *rel-equivalence-onE* [*elim*]:
  assumes *rel-equivalence-on* $P\ R\ f$
  obtains *inflationary-on* $P\ R\ f$ *deflationary-on* $P\ R\ f$
  ⟨*proof*⟩

lemma *rel-equivalence-on-eq-dep-mono-wrt-pred-inf*:
  rel-equivalence-on $P\ R$ = *dep-mono-wrt-pred* $P\ (R \sqcap R^{-1})$
  ⟨*proof*⟩

lemma *bi-related-if-rel-equivalence-on*:
  assumes *rel-equivalence-on* $P\ R\ f$
  and $P\ x$
  shows $x \equiv_R f\ x$
  ⟨*proof*⟩

lemma *rel-equivalence-on-if-all-bi-related*:

assumes $\bigwedge x.\ P\ x \Longrightarrow x \equiv_R f\ x$
shows *rel-equivalence-on* $P\ R\ f$
⟨*proof*⟩

corollary *rel-equivalence-on-iff-all-bi-related*:
rel-equivalence-on $P\ R\ f \longleftrightarrow (\forall x.\ P\ x \longrightarrow x \equiv_R f\ x)$
⟨*proof*⟩

lemma *rel-equivalence-onD* [*dest*]:
assumes *rel-equivalence-on* $P\ R\ f$
and $P\ x$
shows $R\ x\ (f\ x)\ R\ (f\ x)\ x$
⟨*proof*⟩

lemma *rel-equivalence-on-rel-inv-eq-rel-equivalence-on* [*simp*]:
rel-equivalence-on $P\ R^{-1}$ = *rel-equivalence-on* $P\ R$
⟨*proof*⟩

lemma *antimono-rel-equivalence-on-pred* [*iff*]:
antimono ($\lambda(P :: {'a} \Rightarrow bool)$. *rel-equivalence-on* $P\ (R :: {'a} \Rightarrow \text{-})$)
⟨*proof*⟩

lemma *rel-equivalence-on-if-le-pred-if-rel-equivalence-on*:
fixes $P\ P' :: {'a} \Rightarrow bool$ **and** $R :: {'a} \Rightarrow \text{-}$
assumes *rel-equivalence-on* $P\ R\ f$
and $P' \leq P$
shows *rel-equivalence-on* $P'\ R\ f$
⟨*proof*⟩

lemma *rel-equivalence-on-sup-eq* [*simp*]:
(*rel-equivalence-on* :: $({'a} \Rightarrow bool) \Rightarrow ({'a} \Rightarrow \text{-}) \Rightarrow \text{-})\ ((P :: {'a} \Rightarrow bool) \sqcup Q)$
= *rel-equivalence-on* $P \sqcap$ *rel-equivalence-on* Q
⟨*proof*⟩

lemma *in-codom-eq-in-dom-if-rel-equivalence-on-in-field*:
assumes *rel-equivalence-on* (*in-field* R) $R\ f$
shows *in-codom* R = *in-dom* R
⟨*proof*⟩

lemma *reflexive-on-if-transitive-on-if-mon-wrt-pred-if-rel-equivalence-on*:
assumes *rel-equivalence-on* $P\ R\ f$
and $([P] \Rrightarrow_m P)\ f$
and *transitive-on* $P\ R$
shows *reflexive-on* $P\ R$
⟨*proof*⟩

lemma *inflationary-on-eq-rel-equivalence-on-if-symmetric*:
assumes *symmetric* R
shows *inflationary-on* $P\ R$ = *rel-equivalence-on* $P\ R$

⟨*proof*⟩

lemma *deflationary-on-eq-rel-equivalence-on-if-symmetric*:
assumes *symmetric R*
shows *deflationary-on P R* = *rel-equivalence-on P R*
⟨*proof*⟩

definition *rel-equivalence* (*R* :: *'a* ⇒ -) *f* ≡ *rel-equivalence-on* (⊤ :: *'a* ⇒ *bool*) *R f*

lemma *rel-equivalence-eq-rel-equivalence-on*:
rel-equivalence (*R* :: *'a* ⇒ -) *f* = *rel-equivalence-on* (⊤ :: *'a* ⇒ *bool*) *R f*
⟨*proof*⟩

lemma *rel-equivalenceI* [*intro*]:
assumes *inflationary R f*
and *deflationary R f*
shows *rel-equivalence R f*
⟨*proof*⟩

lemma *rel-equivalenceE* [*elim*]:
assumes *rel-equivalence R f*
obtains *inflationary R f deflationary R f*
⟨*proof*⟩

lemma *inflationary-if-rel-equivalence*:
assumes *rel-equivalence R f*
shows *inflationary R f*
⟨*proof*⟩

lemma *deflationary-if-rel-equivalence*:
assumes *rel-equivalence R f*
shows *deflationary R f*
⟨*proof*⟩

lemma *rel-equivalence-on-if-rel-equivalence*:
fixes *P* :: *'a* ⇒ *bool* **and** *R* :: *'a* ⇒ -
assumes *rel-equivalence R f*
shows *rel-equivalence-on P R f*
⟨*proof*⟩

lemma *bi-related-if-rel-equivalence*:
assumes *rel-equivalence R f*
shows $x \equiv_R f\, x$
⟨*proof*⟩

lemma *rel-equivalence-if-all-bi-related*:
assumes $\bigwedge x.\ x \equiv_R f\, x$

shows *rel-equivalence R f*
⟨*proof*⟩

lemma *rel-equivalenceD*:
assumes *rel-equivalence R f*
shows *R x (f x) R (f x) x*
⟨*proof*⟩

lemma *reflexive-on-in-field-if-transitive-if-rel-equivalence-on*:
assumes *rel-equivalence-on (in-field R) R f*
and *transitive R*
shows *reflexive-on (in-field R) R*
⟨*proof*⟩

corollary *preorder-on-in-field-if-transitive-if-rel-equivalence-on*:
assumes *rel-equivalence-on (in-field R) R f*
and *transitive R*
shows *preorder-on (in-field R) R*
⟨*proof*⟩

end

1.3.9 Order Functors

Basic Setup and Results

theory *Order-Functors-Base*
imports
Functions-Inverse
Order-Functions-Base
begin

In the following, we do not add any assumptions to our locales but rather add them as needed to the theorem statements. This allows consumers to state preciser results; particularly, the development of Transport depends on this setup.

locale *orders* =
fixes $L :: 'a \Rightarrow 'b \Rightarrow bool$
and $R :: 'c \Rightarrow 'd \Rightarrow bool$
begin

notation *L* (**infix** $\le_L$ *50*)
notation *R* (**infix** $\le_R$ *50*)

We call $(\le_L)$ the *left relation* and $(\le_R)$ the *right relation.*

abbreviation (*input*) *ge-left* $\equiv (\le_L)^{-1}$
notation *ge-left* (**infix** $\ge_L$ *50*)

abbreviation (*input*) *ge-right* ≡ $(\leq_R)^{-1}$
notation *ge-right* (**infix** $\geq_R$ *50*)

end

Homogeneous orders

locale *hom-orders* = *orders L R*
for *L* :: *'a* ⇒ *'a* ⇒ *bool*
and *R* :: *'b* ⇒ *'b* ⇒ *bool*

locale *order-functor* = *hom-orders L R*
for *L* :: *'a* ⇒ *'a* ⇒ *bool*
and *R* :: *'b* ⇒ *'b* ⇒ *bool*
and *l* :: *'a* ⇒ *'b*
begin

lemma *left-right-rel-left-self-if-reflexive-on-left-if-mono-left*:
assumes $((\leq_L) \Rrightarrow_m (\leq_R))$ *l*
and *reflexive-on P* $(\leq_L)$
and *P x*
shows *l x* $\leq_R$ *l x*
⟨*proof*⟩

lemma *left-right-rel-left-self-if-reflexive-on-in-dom-right-if-mono-left*:
assumes $((\leq_L) \Rrightarrow_m (\leq_R))$ *l*
and *reflexive-on* (*in-dom* $(\leq_R)$) $(\leq_R)$
and *in-dom* $(\leq_L)$ *x*
shows *l x* $\leq_R$ *l x*
⟨*proof*⟩

lemma *left-right-rel-left-self-if-reflexive-on-in-codom-right-if-mono-left*:
assumes $((\leq_L) \Rrightarrow_m (\leq_R))$ *l*
and *reflexive-on* (*in-codom* $(\leq_R)$) $(\leq_R)$
and *in-codom* $(\leq_L)$ *x*
shows *l x* $\leq_R$ *l x*
⟨*proof*⟩

lemma *left-right-rel-left-self-if-reflexive-on-in-field-right-if-mono-left*:
assumes $((\leq_L) \Rrightarrow_m (\leq_R))$ *l*
and *reflexive-on* (*in-field* $(\leq_R)$) $(\leq_R)$
and *in-field* $(\leq_L)$ *x*
shows *l x* $\leq_R$ *l x*
⟨*proof*⟩

lemma *mono-wrt-rel-left-if-reflexive-on-if-le-eq-if-mono-wrt-in-field*:
assumes ([*in-field* $(\leq_L)$] $\Rrightarrow_m$ *P*) *l*
and $(\leq_L) \leq (=)$
and *reflexive-on P* $(\leq_R)$
shows $((\leq_L) \Rrightarrow_m (\leq_R))$ *l*

⟨*proof*⟩

end

locale *order-functors* = *order-functor* $L\ R\ l$ + *flip-of* : *order-functor* $R\ L\ r$
for $L\ R\ l\ r$
begin

We call the composition $r \circ l$ the *unit* and the term $l \circ r$ the *counit* of the order functors pair. This is terminology is borrowed from category theory - the functors are an *adjoint*.

definition *unit* $\equiv r \circ l$

notation *unit* (η)

lemma *unit-eq-comp*: $\eta = r \circ l$ ⟨*proof*⟩

lemma *unit-eq* [*simp*]: $\eta\ x = r\ (l\ x)$ ⟨*proof*⟩

context
begin

Note that by flipping the roles of the left and rights functors, we obtain a flipped interpretation of *order-functors*. In many cases, this allows us to obtain symmetric definitions and theorems for free. As such, in many cases, we do we do not explicitly state those free results but users can obtain them as needed by creating said flipped interpretation.

interpretation *flip* : *order-functors* $R\ L\ r\ l$ ⟨*proof*⟩

definition *counit* $\equiv$ *flip.unit*

notation *counit* (ε)

lemma *counit-eq-comp*: $\varepsilon = l \circ r$ ⟨*proof*⟩

lemma *counit-eq* [*simp*]: $\varepsilon\ x = l\ (r\ x)$ ⟨*proof*⟩

end

context
begin

interpretation *flip* : *order-functors* $R\ L\ r\ l$ ⟨*proof*⟩

lemma *flip-counit-eq-unit*: *flip.counit* $= \eta$
⟨*proof*⟩

lemma *flip-unit-eq-counit*: *flip.unit* $= \varepsilon$

⟨*proof*⟩

lemma *inflationary-on-unit-if-left-rel-right-if-left-right-relI*:
assumes $((\le_L) \Rrightarrow_m (\le_R))\ l$
and *reflexive-on* $P\ (\le_L)$
and $\bigwedge x\ y.\ P\ x \Longrightarrow l\ x \le_R y \Longrightarrow x \le_L r\ y$
shows *inflationary-on* $P\ (\le_L)\ \eta$
⟨*proof*⟩

lemma *deflationary-on-unit-if-right-left-rel-if-right-rel-leftI*:
assumes $((\le_L) \Rrightarrow_m (\le_R))\ l$
and *reflexive-on* $P\ (\le_L)$
and $\bigwedge x\ y.\ P\ x \Longrightarrow y \le_R l\ x \Longrightarrow r\ y \le_L x$
shows *deflationary-on* $P\ (\le_L)\ \eta$
⟨*proof*⟩

context
fixes $P :: 'a \Rightarrow bool$
begin

lemma *rel-equivalence-on-unit-iff-inflationary-on-if-inverse-on*:
assumes *inverse-on* $P\ l\ r$
shows *rel-equivalence-on* $P\ (\le_L)\ \eta \longleftrightarrow$ *inflationary-on* $P\ (\le_L)\ \eta$
⟨*proof*⟩

lemma *reflexive-on-left-if-inflationary-on-unit-if-inverse-on*:
assumes *inverse-on* $P\ l\ r$
and *inflationary-on* $P\ (\le_L)\ \eta$
shows *reflexive-on* $P\ (\le_L)$
⟨*proof*⟩

lemma *rel-equivalence-on-unit-if-reflexive-on-if-inverse-on*:
assumes *inverse-on* $P\ l\ r$
and *reflexive-on* $P\ (\le_L)$
shows *rel-equivalence-on* $P\ (\le_L)\ \eta$
⟨*proof*⟩

end

corollary *rel-equivalence-on-unit-iff-reflexive-on-if-inverse-on*:
fixes $P :: 'a \Rightarrow bool$
assumes *inverse-on* $P\ l\ r$
shows *rel-equivalence-on* $P\ (\le_L)\ \eta \longleftrightarrow$ *reflexive-on* $P\ (\le_L)$
⟨*proof*⟩

end

Here is an example of a free theorem.

notepad

begin
⟨*proof*⟩
end

end

end

1.4 Galois

1.4.1 Basic Abbreviations

theory *Galois-Base*
imports
Order-Functors-Base
begin

locale *galois* = *order-functors*
begin

The locale *galois* serves to define concepts that ultimately lead to the definition of Galois connections and Galois equivalences. Galois connections and equivalences are special cases of adjoints and adjoint equivalences, respectively, known from category theory. As such, in what follows, we sometimes borrow vocabulary from category theory to highlight this connection.

A *Galois connection* between two relations $(\le_L)$ and $(\le_R)$ consists of two monotone functions (i.e. order functors) l and r such that $(x \le_L r\ y) = (l\ x \le_R y)$. We call this the *Galois property.* l is called the *left adjoint* and r the *right adjoint.* We call $(\le_L)$ the *left relation* and $(\le_R)$ the *right relation.* By composing the adjoints, we obtain the unit η and counit ε of the Galois connection.

end

end

1.4.2 Basics For Relator For Galois Connections

theory *Galois-Relator-Base*
imports
Galois-Base
begin

locale *galois-rel* = *orders L R*
for L :: $'a \Rightarrow 'b \Rightarrow bool$
and R :: $'c \Rightarrow 'd \Rightarrow bool$
and r :: $'d \Rightarrow 'b$
begin

Morally speaking, the Galois relator characterises when two terms x and y are "similar".

definition *Galois* $x\ y \equiv$ *in-codom* $(\leq_R)\ y \wedge x \leq_L r\ y$

abbreviation *left-Galois* $\equiv$ *Galois*
notation *left-Galois* (**infix** ${}_L\lessapprox$ *50*)

abbreviation (*input*) *ge-Galois-left* $\equiv ({}_L\lessapprox)^{-1}$
notation *ge-Galois-left* (**infix** $\gtrapprox_L$ *50*)

Here we only introduced the (left) Galois relator $({}_L\lessapprox)$. All other variants can be introduced by considering suitable flipped and inversed interpretations (see `Half_Galois_Property.thy`).

lemma *left-GaloisI* [*intro*]:
  assumes *in-codom* $(\leq_R)\ y$
  and $x \leq_L r\ y$
  shows $x\ {}_L\lessapprox\ y$
  ⟨*proof*⟩

lemma *left-GaloisE* [*elim*]:
  assumes $x\ {}_L\lessapprox\ y$
  obtains *in-codom* $(\leq_R)\ y\ x \leq_L r\ y$
  ⟨*proof*⟩

corollary *in-dom-left-if-left-Galois*:
  assumes $x\ {}_L\lessapprox\ y$
  shows *in-dom* $(\leq_L)\ x$
  ⟨*proof*⟩

corollary *left-Galois-iff-in-codom-and-left-rel-right*:
  $x\ {}_L\lessapprox\ y \longleftrightarrow$ *in-codom* $(\leq_R)\ y \wedge x \leq_L r\ y$
  ⟨*proof*⟩

lemma *left-Galois-restrict-left-eq-left-Galois-left-restrict-left*:
  $({}_L\lessapprox)\restriction_{P\ ::\ 'a \Rightarrow bool} =$ *galois-rel.Galois* $(\leq_L)\restriction_P\ (\leq_R)\ r$
  ⟨*proof*⟩

lemma *left-Galois-restrict-right-eq-left-Galois-right-restrict-right*:
  $({}_L\lessapprox)\upharpoonleft_{P\ ::\ 'd \Rightarrow bool} =$ *galois-rel.Galois* $(\leq_L)\ (\leq_R)\upharpoonleft_P\ r$
  ⟨*proof*⟩

end

end

Equivalences

theory *Order-Equivalences*

imports
Order-Functors-Base
Partial-Equivalence-Relations
Preorders
begin

context *order-functors*
begin

definition *order-equivalence* $\equiv$
$((\le_L) \Rrightarrow_m (\le_R))\ l \wedge$
$((\le_R) \Rrightarrow_m (\le_L))\ r \wedge$
rel-equivalence-on $($*in-field* $(\le_L))\ (\le_L)\ \eta \wedge$
rel-equivalence-on $($*in-field* $(\le_R))\ (\le_R)\ \varepsilon$

notation *order-functors.order-equivalence* (**infix** $\equiv_o$ *50*)

lemma *order-equivalenceI* [*intro*]:
assumes $((\le_L) \Rrightarrow_m (\le_R))\ l$
and $((\le_R) \Rrightarrow_m (\le_L))\ r$
and *rel-equivalence-on* $($*in-field* $(\le_L))\ (\le_L)\ \eta$
and *rel-equivalence-on* $($*in-field* $(\le_R))\ (\le_R)\ \varepsilon$
shows $((\le_L) \equiv_o (\le_R))\ l\ r$
⟨*proof*⟩

lemma *order-equivalenceE* [*elim*]:
assumes $((\le_L) \equiv_o (\le_R))\ l\ r$
obtains $((\le_L) \Rrightarrow_m (\le_R))\ l\ ((\le_R) \Rrightarrow_m (\le_L))\ r$
rel-equivalence-on $($*in-field* $(\le_L))\ (\le_L)\ \eta$
rel-equivalence-on $($*in-field* $(\le_R))\ (\le_R)\ \varepsilon$
⟨*proof*⟩

interpretation *of* : *order-functors* $S\ T\ f\ g$ **for** $S\ T\ f\ g$ ⟨*proof*⟩

lemma *rel-inv-order-equivalence-eq-order-equivalence* [*simp*]:
$((\le_R) \equiv_o (\le_L))^{-1} = ((\le_L) \equiv_o (\le_R))$
⟨*proof*⟩

corollary *order-equivalence-right-left-iff-order-equivalence-left-right*:
$((\le_R) \equiv_o (\le_L))\ r\ l \longleftrightarrow ((\le_L) \equiv_o (\le_R))\ l\ r$
⟨*proof*⟩

Due to the symmetry given by $((\le_R) \equiv_o (\le_L))\ r\ l =$ *order-equivalence*, for any theorem on $(\le_L)$, we obtain a corresponding theorem on $(\le_R)$ by flipping the roles of the two functors. As such, in what follows, we do not explicitly state these free theorems but users can obtain them as needed by creating a flipped interpretation of *order-functors*.

lemma *order-equivalence-rel-inv-eq-order-equivalence* [*simp*]:
$((\ge_L) \equiv_o (\ge_R)) = ((\le_L) \equiv_o (\le_R))$

⟨*proof*⟩

lemma *in-codom-left-eq-in-dom-left-if-order-equivalence*:
assumes $((\le_L) \equiv_o (\le_R))\ l\ r$
shows *in-codom* $(\le_L)$ = *in-dom* $(\le_L)$
⟨*proof*⟩

corollary *preorder-on-in-field-left-if-transitive-if-order-equivalence*:
assumes $((\le_L) \equiv_o (\le_R))\ l\ r$
and *transitive* $(\le_L)$
shows *preorder-on* (*in-field* $(\le_L)$) $(\le_L)$
⟨*proof*⟩

lemma *order-equivalence-partial-equivalence-rel-not-reflexive-not-transitive*:
assumes $\exists (y :: 'b)\ y'.\ y \neq y'$
shows $\exists (L :: 'a \Rightarrow 'a \Rightarrow bool)\ (R :: 'b \Rightarrow 'b \Rightarrow bool)\ l\ r.$
$(L \equiv_o R)\ l\ r \wedge$ *partial-equivalence-rel* $L \wedge$
$\neg$(*reflexive-on* (*in-field* R) R) $\wedge$ $\neg$(*transitive-on* (*in-field* R) R)
⟨*proof*⟩

end

end

1.4.3 Half Galois Property

theory *Half-Galois-Property*
imports
Galois-Relator-Base
Order-Equivalences
begin

As the definition of the Galois property also works on heterogeneous relations, we define the concepts in a locale that generalises *galois*.

locale *galois-prop* = *orders* L R
for $L :: 'a \Rightarrow 'b \Rightarrow bool$
and $R :: 'c \Rightarrow 'd \Rightarrow bool$
and $l :: 'a \Rightarrow 'c$
and $r :: 'd \Rightarrow 'b$
begin

sublocale *galois-rel* L R r ⟨*proof*⟩

interpretation *gr-flip-inv* : *galois-rel* $(\ge_R)$ $(\ge_L)$ l ⟨*proof*⟩

abbreviation *right-ge-Galois* ≡ *gr-flip-inv.Galois*
notation *right-ge-Galois* (**infix** ${}_R\gtrapprox$ *50*)

abbreviation (*input*) *Galois-right* $\equiv$ *gr-flip-inv.ge-Galois-left*
notation *Galois-right* (**infix** $\lessapprox_R$ *50*)

lemma *Galois-rightI* [*intro*]:
  assumes *in-dom* $(\leq_L)$ x
  and $l\ x \leq_R y$
  shows $x \lessapprox_R y$
  ⟨*proof*⟩

lemma *Galois-rightE* [*elim*]:
  assumes $x \lessapprox_R y$
  obtains *in-dom* $(\leq_L)$ x $l\ x \leq_R y$
  ⟨*proof*⟩

corollary *Galois-right-iff-in-dom-and-left-right-rel*:
  $x \lessapprox_R y \longleftrightarrow$ *in-dom* $(\leq_L)\ x \wedge l\ x \leq_R y$
  ⟨*proof*⟩

Unlike common literature, we split the definition of the Galois property into two halves. This has its merits in modularity of proofs and preciser statement of required assumptions.

definition *half-galois-prop-left* $\equiv \forall x\ y.\ x\ {}_L\lessapprox\ y \longrightarrow l\ x \leq_R y$

notation *galois-prop.half-galois-prop-left* (**infix** ${}_h\trianglelefteq$ *50*)

lemma *half-galois-prop-leftI* [*intro*]:
  assumes $\bigwedge x\ y.\ x\ {}_L\lessapprox\ y \Longrightarrow l\ x \leq_R y$
  shows $((\leq_L)\ {}_h\trianglelefteq\ (\leq_R))\ l\ r$
  ⟨*proof*⟩

lemma *half-galois-prop-leftD* [*dest*]:
  assumes $((\leq_L)\ {}_h\trianglelefteq\ (\leq_R))\ l\ r$
  and $x\ {}_L\lessapprox\ y$
  shows $l\ x \leq_R y$
  ⟨*proof*⟩

Observe that the second half can be obtained by creating an appropriately flipped and inverted interpretation of *galois-prop*. Indeed, many concepts in our formalisation are "closed" under inversion, i.e. taking their inversion yields a statement for a related concept. Many theorems can thus be derived for free by inverting (and flipping) the concepts at hand. In such cases, we only state those theorems that require some non-trivial setup. All other theorems can simply be obtained by creating a suitable locale interpretation.

interpretation *flip-inv* : *galois-prop* $(\geq_R)$ $(\geq_L)$ $r\ l$ ⟨*proof*⟩

definition *half-galois-prop-right* $\equiv$ *flip-inv.half-galois-prop-left*

notation *galois-prop.half-galois-prop-right* (**infix** $\trianglelefteq_h$ *50*)

lemma *half-galois-prop-rightI* [*intro*]:
  assumes $\bigwedge x\ y.\ x \lessapprox_R y \Longrightarrow x \le_L r\ y$
  shows $((\le_L) \trianglelefteq_h (\le_R))\ l\ r$
  ⟨*proof*⟩

lemma *half-galois-prop-rightD* [*dest*]:
  assumes $((\le_L) \trianglelefteq_h (\le_R))\ l\ r$
  and $x \lessapprox_R y$
  shows $x \le_L r\ y$
  ⟨*proof*⟩

interpretation g : *galois-prop* $S\ T\ f\ g$ **for** $S\ T\ f\ g$ ⟨*proof*⟩

lemma *rel-inv-half-galois-prop-right-eq-half-galois-prop-left-rel-inv* [*simp*]:
  $((\le_R) \trianglelefteq_h (\le_L))^{-1} = ((\ge_L)\ {}_h\trianglelefteq\ (\ge_R))$
  ⟨*proof*⟩

corollary *half-galois-prop-left-rel-inv-iff-half-galois-prop-right* [*iff*]:
  $((\ge_L)\ {}_h\trianglelefteq\ (\ge_R))\ f\ g \longleftrightarrow ((\le_R) \trianglelefteq_h (\le_L))\ g\ f$
  ⟨*proof*⟩

lemma *rel-inv-half-galois-prop-left-eq-half-galois-prop-right-rel-inv* [*simp*]:
  $((\le_R)\ {}_h\trianglelefteq\ (\le_L))^{-1} = ((\ge_L) \trianglelefteq_h (\ge_R))$
  ⟨*proof*⟩

corollary *half-galois-prop-right-rel-inv-iff-half-galois-prop-left* [*iff*]:
  $((\ge_L) \trianglelefteq_h (\ge_R))\ f\ g \longleftrightarrow ((\le_R)\ {}_h\trianglelefteq\ (\le_L))\ g\ f$
  ⟨*proof*⟩

end

context *galois*
begin

sublocale *galois-prop* $L\ R\ l\ r$ ⟨*proof*⟩

interpretation *flip* : *galois* $R\ L\ r\ l$ ⟨*proof*⟩

abbreviation *right-Galois* ≡ *flip.Galois*
notation *right-Galois* (**infix** ${}_R\lessapprox$ *50*)

abbreviation (*input*) *ge-Galois-right* ≡ *flip.ge-Galois-left*
notation *ge-Galois-right* (**infix** $\gtrapprox_R$ *50*)

abbreviation *left-ge-Galois* ≡ *flip.right-ge-Galois*
notation *left-ge-Galois* (**infix** ${}_L\gtrapprox$ *50*)

abbreviation (*input*) *Galois-left* $\equiv$ *flip.Galois-right*
notation *Galois-left* (**infix** $\lessapprox_L$ *50*)

context
begin

interpretation *flip-inv* : *galois* $(\geq_R)$ $(\geq_L)$ *r l* ⟨*proof*⟩

lemma *rel-unit-if-left-rel-if-mono-wrt-relI*:
  assumes $((\leq_L) \Rrightarrow_m (\leq_R))\ l$
  and $x \lessapprox_R l\ x' \Longrightarrow x \leq_L \eta\ x'$
  and $x \leq_L x'$
  shows $x \leq_L \eta\ x'$
  ⟨*proof*⟩

corollary *rel-unit-if-left-rel-if-half-galois-prop-right-if-mono-wrt-rel*:
  assumes $((\leq_L) \Rrightarrow_m (\leq_R))\ l$
  and $((\leq_L) \trianglelefteq_h (\leq_R))\ l\ r$
  and $x \leq_L x'$
  shows $x \leq_L \eta\ x'$
  ⟨*proof*⟩

corollary *rel-unit-if-reflexive-on-if-half-galois-prop-right-if-mono-wrt-rel*:
  assumes $((\leq_L) \Rrightarrow_m (\leq_R))\ l$
  and $((\leq_L) \trianglelefteq_h (\leq_R))\ l\ r$
  and *reflexive-on* P $(\leq_L)$
  and $P\ x$
  shows $x \leq_L \eta\ x$
  ⟨*proof*⟩

corollary *inflationary-on-unit-if-reflexive-on-if-half-galois-prop-rightI*:
  fixes $P :: 'a \Rightarrow bool$
  assumes $((\leq_L) \Rrightarrow_m (\leq_R))\ l$
  and $((\leq_L) \trianglelefteq_h (\leq_R))\ l\ r$
  and *reflexive-on* P $(\leq_L)$
  shows *inflationary-on* P $(\leq_L)$ η
  ⟨*proof*⟩

interpretation *flip* : *galois-prop* $R\ L\ r\ l$ ⟨*proof*⟩

lemma *right-rel-if-Galois-left-right-if-deflationary-onI*:
  assumes $((\leq_R) \Rrightarrow_m (\leq_L))\ r$
  and $((\leq_R) \trianglelefteq_h (\leq_L))\ r\ l$
  and *deflationary-on* P $(\leq_R)$ ε
  and *transitive* $(\leq_R)$
  and $y \lessapprox_L r\ y'$
  and $P\ y'$
  shows $y \leq_R y'$
  ⟨*proof*⟩

lemma *half-galois-prop-left-left-right-if-transitive-if-deflationary-on-if-mono-wrt-rel*:
  assumes $((\leq_L) \Rrightarrow_m (\leq_R))\ l$
  and *deflationary-on* (*in-codom* $(\leq_R)$) $(\leq_R)\ \varepsilon$
  and *transitive* $(\leq_R)$
  shows $((\leq_L)\ {}_h\trianglelefteq\ (\leq_R))\ l\ r$
  ⟨*proof*⟩

end

interpretation *flip-inv* : *galois* $(\geq_R)$ $(\geq_L)$ $r\ l$
  rewrites *flip-inv.unit* $\equiv \varepsilon$ **and** *flip-inv.counit* $\equiv \eta$
  and $\bigwedge R\ S.\ (R^{-1} \Rrightarrow_m S^{-1}) \equiv (R \Rrightarrow_m S)$
  and $\bigwedge R\ S\ f\ g.\ (R^{-1} \trianglelefteq_h S^{-1})\ f\ g \equiv (S\ {}_h\trianglelefteq\ R)\ g\ f$
  and $((\geq_R)\ {}_h\trianglelefteq\ (\geq_L))\ r\ l \equiv ((\leq_L) \trianglelefteq_h (\leq_R))\ l\ r$
  and $\bigwedge R.\ R^{-1-1} \equiv R$
  and $\bigwedge P\ R.$ *inflationary-on* $P\ R^{-1} \equiv$ *deflationary-on* $P\ R$
  and $\bigwedge P\ R.$ *deflationary-on* $P\ R^{-1} \equiv$ *inflationary-on* $P\ R$
  and $\bigwedge(P :: {}'b \Rightarrow bool).$ *reflexive-on* $P\ (\geq_R) \equiv$ *reflexive-on* $P\ (\leq_R)$
  and $\bigwedge R.$ *transitive* $R^{-1} \equiv$ *transitive* R
  and $\bigwedge R.$ *in-codom* $R^{-1} \equiv$ *in-dom* R
  ⟨*proof*⟩

corollary *counit-rel-if-right-rel-if-mono-wrt-relI*:
  assumes $((\leq_R) \Rrightarrow_m (\leq_L))\ r$
  and $r\ y\ {}_L\lessapprox\ y' \Longrightarrow \varepsilon\ y \leq_R y'$
  and $y \leq_R y'$
  shows $\varepsilon\ y \leq_R y'$
  ⟨*proof*⟩

corollary *counit-rel-if-right-rel-if-half-galois-prop-left-if-mono-wrt-rel*:
  assumes $((\leq_R) \Rrightarrow_m (\leq_L))\ r$
  and $((\leq_L)\ {}_h\trianglelefteq\ (\leq_R))\ l\ r$
  and $y \leq_R y'$
  shows $\varepsilon\ y \leq_R y'$
  ⟨*proof*⟩

corollary *counit-rel-if-reflexive-on-if-half-galois-prop-left-if-mono-wrt-rel*:
  assumes $((\leq_R) \Rrightarrow_m (\leq_L))\ r$
  and $((\leq_L)\ {}_h\trianglelefteq\ (\leq_R))\ l\ r$
  and *reflexive-on* $P\ (\leq_R)$
  and $P\ y$
  shows $\varepsilon\ y \leq_R y$
  ⟨*proof*⟩

corollary *deflationary-on-counit-if-reflexive-on-if-half-galois-prop-leftI*:
  fixes $P :: {}'b \Rightarrow bool$
  assumes $((\leq_R) \Rrightarrow_m (\leq_L))\ r$
  and $((\leq_L)\ {}_h\trianglelefteq\ (\leq_R))\ l\ r$

and *reflexive-on* P $(\leq_R)$
shows *deflationary-on* P $(\leq_R)$ ε
⟨*proof*⟩

corollary *left-rel-if-left-right-Galois-if-inflationary-onI*:
assumes $((\leq_L) \Rrightarrow_m (\leq_R))$ l
and $((\leq_R)\ {}_h\trianglelefteq\ (\leq_L))$ r l
and *inflationary-on* P $(\leq_L)$ η
and *transitive* $(\leq_L)$
and $l\ x\ {}_R\lessapprox\ x'$
and $P\ x$
shows $x \leq_L x'$
⟨*proof*⟩

corollary *half-galois-prop-right-left-right-if-transitive-if-inflationary-on-if-mono-wrt-rel*:
assumes $((\leq_R) \Rrightarrow_m (\leq_L))$ r
and *inflationary-on* (*in-dom* $(\leq_L)$) $(\leq_L)$ η
and *transitive* $(\leq_L)$
shows $((\leq_L) \trianglelefteq_h (\leq_R))$ l r
⟨*proof*⟩

end

context *order-functors*
begin

interpretation *g* : *galois* L R l r ⟨*proof*⟩
interpretation *flip-g* : *galois* R L r l
rewrites *flip-g.unit* $\equiv \varepsilon$ **and** *flip-g.counit* $\equiv \eta$
⟨*proof*⟩

lemma *left-rel-if-left-right-rel-left-if-order-equivalenceI*:
assumes $((\leq_L) \equiv_o (\leq_R))$ l r
and *transitive* $(\leq_L)$
and $l\ x \leq_R l\ x'$
and *in-dom* $(\leq_L)$ x
and *in-codom* $(\leq_L)$ x'
shows $x \leq_L x'$
⟨*proof*⟩

end

end

1.4.4 Galois Property

theory *Galois-Property*
imports

Half-Galois-Property
begin

context *galois-prop*
begin

definition *galois-prop* $\equiv$ $((\leq_L)\ {}_h\trianglelefteq\ (\leq_R)) \sqcap ((\leq_L)\ \trianglelefteq_h\ (\leq_R))$

notation *galois-prop.galois-prop* (**infix** $\trianglelefteq$ *50*)

lemma *galois-propI* [*intro*]:
assumes $((\leq_L)\ {}_h\trianglelefteq\ (\leq_R))\ l\ r$
and $((\leq_L)\ \trianglelefteq_h\ (\leq_R))\ l\ r$
shows $((\leq_L)\ \trianglelefteq\ (\leq_R))\ l\ r$
⟨*proof*⟩

lemma *galois-propI'*:
assumes $\bigwedge x\ y.$ *in-dom* $(\leq_L)\ x \Longrightarrow$ *in-codom* $(\leq_R)\ y \Longrightarrow x \leq_L r\ y \longleftrightarrow l\ x \leq_R$
y
shows $((\leq_L)\ \trianglelefteq\ (\leq_R))\ l\ r$
⟨*proof*⟩

lemma *galois-propE* [*elim*]:
assumes $((\leq_L)\ \trianglelefteq\ (\leq_R))\ l\ r$
obtains $((\leq_L)\ {}_h\trianglelefteq\ (\leq_R))\ l\ r\ ((\leq_L)\ \trianglelefteq_h\ (\leq_R))\ l\ r$
⟨*proof*⟩

interpretation g : *galois-prop* $S\ T\ f\ g$ **for** $S\ T\ f\ g$⟨*proof*⟩

lemma *galois-prop-eq-half-galois-prop-left-rel-inf-half-galois-prop-right*:
$((\leq_L)\ \trianglelefteq\ (\leq_R)) = ((\leq_L)\ {}_h\trianglelefteq\ (\leq_R)) \sqcap ((\leq_L)\ \trianglelefteq_h\ (\leq_R))$
⟨*proof*⟩

lemma *galois-prop-left-rel-right-iff-left-right-rel*:
assumes $((\leq_L)\ \trianglelefteq\ (\leq_R))\ l\ r$
and *in-dom* $(\leq_L)\ x$ *in-codom* $(\leq_R)\ y$
shows $x \leq_L r\ y \longleftrightarrow l\ x \leq_R y$
⟨*proof*⟩

lemma *rel-inv-galois-prop-eq-galois-prop-rel-inv* [*simp*]:
$((\leq_R)\ \trianglelefteq\ (\leq_L))^{-1} = ((\geq_L)\ \trianglelefteq\ (\geq_R))$
⟨*proof*⟩

corollary *galois-prop-rel-inv-iff-galois-prop* [*iff*]:
$((\geq_L)\ \trianglelefteq\ (\geq_R))\ f\ g \longleftrightarrow ((\leq_R)\ \trianglelefteq\ (\leq_L))\ g\ f$
⟨*proof*⟩

end

context *galois*
begin

lemma *galois-prop-left-right-if-transitive-if-deflationary-on-if-inflationary-on-if-mono-wrt-rel*:
assumes $((\leq_L) \Rrightarrow_m (\leq_R))\ l$ **and** $((\leq_R) \Rrightarrow_m (\leq_L))\ r$
and *inflationary-on* $(in\text{-}dom\ (\leq_L))\ (\leq_L)\ \eta$
and *deflationary-on* $(in\text{-}codom\ (\leq_R))\ (\leq_R)\ \varepsilon$
and *transitive* $(\leq_L)$ *transitive* $(\leq_R)$
shows $((\leq_L) \trianglelefteq (\leq_R))\ l\ r$
⟨*proof*⟩

end

end

1.4.5 Galois Connections

theory *Galois-Connections*
imports
Galois-Property
begin

context *galois*
begin

definition *galois-connection* ≡
$((\leq_L) \Rrightarrow_m (\leq_R))\ l \wedge ((\leq_R) \Rrightarrow_m (\leq_L))\ r \wedge ((\leq_L) \trianglelefteq (\leq_R))\ l\ r$

notation *galois.galois-connection* (**infix** ⊣ *50*)

lemma *galois-connectionI* [*intro*]:
assumes $((\leq_L) \Rrightarrow_m (\leq_R))\ l$ **and** $((\leq_R) \Rrightarrow_m (\leq_L))\ r$
and $((\leq_L) \trianglelefteq (\leq_R))\ l\ r$
shows $((\leq_L) \dashv (\leq_R))\ l\ r$
⟨*proof*⟩

lemma *galois-connectionE* [*elim*]:
assumes $((\leq_L) \dashv (\leq_R))\ l\ r$
obtains $((\leq_L) \Rrightarrow_m (\leq_R))\ l\ ((\leq_R) \Rrightarrow_m (\leq_L))\ r\ ((\leq_L) \trianglelefteq (\leq_R))\ l\ r$
⟨*proof*⟩

context
begin

interpretation *g* : *galois S T f g* **for** *S T f g*⟨*proof*⟩

lemma *rel-inv-galois-connection-eq-galois-connection-rel-inv* [*simp*]:
$((\leq_R) \dashv (\leq_L))^{-1} = ((\geq_L) \dashv (\geq_R))$

⟨*proof*⟩

corollary *galois-connection-rel-inv-iff-galois-connection* [*iff*]:
$((\geq_L) \dashv (\geq_R))\ l\ r \longleftrightarrow ((\leq_R) \dashv (\leq_L))\ r\ l$
⟨*proof*⟩

lemma *rel-unit-if-left-rel-if-galois-connection*:
assumes $((\leq_L) \dashv (\leq_R))\ l\ r$
and $x \leq_L x'$
shows $x \leq_L \eta\ x'$
⟨*proof*⟩

end

lemma *counit-rel-if-right-rel-if-galois-connection*:
assumes $((\leq_L) \dashv (\leq_R))\ l\ r$
and $y \leq_R y'$
shows $\varepsilon\ y \leq_R y'$
⟨*proof*⟩

lemma *rel-unit-if-reflexive-on-if-galois-connection*:
assumes $((\leq_L) \dashv (\leq_R))\ l\ r$
and *reflexive-on* $P\ (\leq_L)$
and $P\ x$
shows $x \leq_L \eta\ x$
⟨*proof*⟩

lemma *counit-rel-if-reflexive-on-if-galois-connection*:
assumes $((\leq_L) \dashv (\leq_R))\ l\ r$
and *reflexive-on* $P\ (\leq_R)$
and $P\ y$
shows $\varepsilon\ y \leq_R y$
⟨*proof*⟩

lemma *inflationary-on-unit-if-reflexive-on-if-galois-connection*:
fixes $P :: 'a \Rightarrow bool$
assumes $((\leq_L) \dashv (\leq_R))\ l\ r$
and *reflexive-on* $P\ (\leq_L)$
shows *inflationary-on* $P\ (\leq_L)\ \eta$
⟨*proof*⟩

lemma *deflationary-on-counit-if-reflexive-on-if-galois-connection*:
fixes $P :: 'b \Rightarrow bool$
assumes $((\leq_L) \dashv (\leq_R))\ l\ r$
and *reflexive-on* $P\ (\leq_R)$
shows *deflationary-on* $P\ (\leq_R)\ \varepsilon$
⟨*proof*⟩

end

end

1.4.6 Galois Equivalences

theory *Galois-Equivalences*
  imports
    Galois-Connections
    Order-Equivalences
    Partial-Equivalence-Relations
begin

context *galois*
begin

In the literature, an adjoint equivalence is an adjunction for which the unit and counit are natural isomorphisms. Translated to the category of orders, this means that a *Galois equivalence* between two relations $(\leq_L)$ and $(\leq_R)$ is a Galois connection for which the unit η is *deflationary* and the counit ε is *inflationary*.

For reasons of symmetry, we give a different definition which next to *galois-connection* requires *galois-prop l r*. In other words, a Galois equivalence is a Galois connection for which the left and right adjoints are also right and left adjoints, respectively. As shown below, in the case of preorders, the definitions coincide.

definition *galois-equivalence* $\equiv ((\leq_L) \dashv (\leq_R))\ l\ r \wedge ((\leq_R) \trianglelefteq (\leq_L))\ r\ l$

notation *galois.galois-equivalence* (**infix** $\equiv_G$ *50*)

lemma *galois-equivalenceI* [*intro*]:
  assumes $((\leq_L) \dashv (\leq_R))\ l\ r$
  and $((\leq_R) \trianglelefteq (\leq_L))\ r\ l$
  shows $((\leq_L) \equiv_G (\leq_R))\ l\ r$
  ⟨*proof*⟩

lemma *galois-equivalenceE* [*elim*]:
  assumes $((\leq_L) \equiv_G (\leq_R))\ l\ r$
  obtains $((\leq_L) \dashv (\leq_R))\ l\ r\ ((\leq_R) \dashv (\leq_L))\ r\ l$
  ⟨*proof*⟩

context
begin

interpretation g : *galois* $S\ T\ f\ g$ **for** $S\ T\ f\ g$⟨*proof*⟩

lemma *galois-equivalence-eq-galois-connection-rel-inf-galois-prop*:
  $((\leq_L) \equiv_G (\leq_R)) = ((\leq_L) \dashv (\leq_R)) \sqcap ((\geq_L) \trianglelefteq (\geq_R))$

$\langle proof \rangle$

lemma *rel-inv-galois-equivalence-eq-galois-equivalence* [*simp*]:
$((\leq_R) \equiv_G (\leq_L))^{-1} = ((\leq_L) \equiv_G (\leq_R))$
$\langle proof \rangle$

corollary *galois-equivalence-right-left-iff-galois-equivalence-left-right*:
$((\leq_R) \equiv_G (\leq_L))\ r\ l \longleftrightarrow ((\leq_L) \equiv_G (\leq_R))\ l\ r$
$\langle proof \rangle$

lemma *galois-equivalence-rel-inv-eq-galois-equivalence* [*simp*]:
$((\geq_L) \equiv_G (\geq_R)) = ((\leq_L) \equiv_G (\leq_R))$
$\langle proof \rangle$

lemma *inflationary-on-unit-if-reflexive-on-if-galois-equivalence*:
fixes $P :: 'a \Rightarrow bool$
assumes $((\leq_L) \equiv_G (\leq_R))\ l\ r$
and *reflexive-on* $P\ (\leq_L)$
shows *inflationary-on* $P\ (\leq_L)\ \eta$
$\langle proof \rangle$

end

lemma *deflationary-on-unit-if-reflexive-on-if-galois-equivalence*:
fixes $P :: 'a \Rightarrow bool$
assumes $((\leq_L) \equiv_G (\leq_R))\ l\ r$
and *reflexive-on* $P\ (\leq_L)$
shows *deflationary-on* $P\ (\leq_L)\ \eta$
$\langle proof \rangle$

Every *galois-equivalence* on reflexive orders is a Galois equivalence in the sense of the common literature.

lemma *rel-equivalence-on-unit-if-reflexive-on-if-galois-equivalence*:
fixes $P :: 'a \Rightarrow bool$
assumes $((\leq_L) \equiv_G (\leq_R))\ l\ r$
and *reflexive-on* $P\ (\leq_L)$
shows *rel-equivalence-on* $P\ (\leq_L)\ \eta$
$\langle proof \rangle$

lemma *galois-equivalence-partial-equivalence-rel-not-reflexive-not-transitive*:
assumes $\exists (y :: 'b)\ y'.\ y \neq y'$
shows $\exists (L :: 'a \Rightarrow 'a \Rightarrow bool)\ (R :: 'b \Rightarrow 'b \Rightarrow bool)\ l\ r.$
$(L \equiv_G R)\ l\ r \wedge$ *partial-equivalence-rel* $L \wedge$
$\neg($*reflexive-on* $($*in-field* $R)\ R) \wedge \neg($*transitive-on* $($*in-field* $R)\ R)$
$\langle proof \rangle$

1.4.7 Equivalence of Order Equivalences and Galois Equivalences

In general categories, every adjoint equivalence is an equivalence but not vice versa. In the category of preorders, however, they are morally the same: the adjoint zigzag equations are satisfied up to unique isomorphism rather than equality. In the category of partial orders, the concepts coincide.

lemma *half-galois-prop-left-left-right-if-transitive-if-order-equivalence*:
assumes $((\leq_L) \equiv_o (\leq_R))\ l\ r$
and *transitive* $(\leq_L)$ *transitive* $(\leq_R)$
shows $((\leq_L)\ {}_h\trianglelefteq\ (\leq_R))\ l\ r$
⟨*proof*⟩

lemma *half-galois-prop-right-left-right-if-transitive-if-order-equivalence*:
assumes $((\leq_L) \equiv_o (\leq_R))\ l\ r$
and *transitive* $(\leq_L)$ *transitive* $(\leq_R)$
shows $((\leq_L) \trianglelefteq_h (\leq_R))\ l\ r$
⟨*proof*⟩

lemma *galois-prop-left-right-if-transitive-if-order-equivalence*:
assumes $((\leq_L) \equiv_o (\leq_R))\ l\ r$
and *transitive* $(\leq_L)$ *transitive* $(\leq_R)$
shows $((\leq_L) \trianglelefteq (\leq_R))\ l\ r$
⟨*proof*⟩

corollary *galois-connection-left-right-if-transitive-if-order-equivalence*:
assumes $((\leq_L) \equiv_o (\leq_R))\ l\ r$
and *transitive* $(\leq_L)$ *transitive* $(\leq_R)$
shows $((\leq_L) \dashv (\leq_R))\ l\ r$
⟨*proof*⟩

interpretation *flip* : *galois* $R\ L\ r\ l$
rewrites *flip.unit* $\equiv \varepsilon$
⟨*proof*⟩

corollary *galois-equivalence-left-right-if-transitive-if-order-equivalence*:
assumes $((\leq_L) \equiv_o (\leq_R))\ l\ r$
and *transitive* $(\leq_L)$ *transitive* $(\leq_R)$
shows $((\leq_L) \equiv_G (\leq_R))\ l\ r$
⟨*proof*⟩

lemma *order-equivalence-if-reflexive-on-in-field-if-galois-equivalence*:
assumes $((\leq_L) \equiv_G (\leq_R))\ l\ r$
and *reflexive-on* (*in-field* $(\leq_L)$) $(\leq_L)$ *reflexive-on* (*in-field* $(\leq_R)$) $(\leq_R)$
shows $((\leq_L) \equiv_o (\leq_R))\ l\ r$
⟨*proof*⟩

corollary *galois-equivalence-eq-order-equivalence-if-preorder-on-in-field*:
assumes *preorder-on* (*in-field* $(\leq_L)$) $(\leq_L)$ *preorder-on* (*in-field* $(\leq_R)$) $(\leq_R)$

shows $((\leq_L) \equiv_G (\leq_R)) = ((\leq_L) \equiv_o (\leq_R))$
⟨*proof*⟩

end

end

1.4.8 Relator For Galois Connections

theory *Galois-Relator*
imports
Galois-Relator-Base
Galois-Property
begin

context *galois-prop*
begin

interpretation *flip-inv* : *galois-rel* $(\geq_R)$ $(\geq_L)$ l ⟨*proof*⟩

lemma *left-Galois-if-Galois-right-if-half-galois-prop-right*:
assumes $((\leq_L) \trianglelefteq_h (\leq_R))$ l r
and $x \lessapprox_R y$
shows $x\ {}_L\lessapprox\ y$
⟨*proof*⟩

lemma *Galois-right-if-left-Galois-if-half-galois-prop-left*:
assumes $((\leq_L)\ {}_h\trianglelefteq\ (\leq_R))$ l r
and $x\ {}_L\lessapprox\ y$
shows $x \lessapprox_R y$
⟨*proof*⟩

corollary *Galois-right-iff-left-Galois-if-galois-prop* [*iff*]:
assumes $((\leq_L) \trianglelefteq (\leq_R))$ l r
shows $x \lessapprox_R y \longleftrightarrow x\ {}_L\lessapprox\ y$
⟨*proof*⟩

lemma *rel-inv-Galois-eq-flip-Galois-rel-inv-if-galois-prop* [*simp*]:
assumes $((\leq_L) \trianglelefteq (\leq_R))$ l r
shows $(\gtrapprox_L) = ({}_R\gtrapprox)$
⟨*proof*⟩

corollary *flip-Galois-rel-inv-iff-Galois-if-galois-prop* [*iff*]:
assumes $((\leq_L) \trianglelefteq (\leq_R))$ l r
shows $y\ {}_R\gtrapprox\ x \longleftrightarrow x\ {}_L\lessapprox\ y$
⟨*proof*⟩

corollary *inv-flip-Galois-rel-inv-eq-Galois-if-galois-prop* [*simp*]:

assumes $((\le_L) \trianglelefteq (\le_R))\ l\ r$
shows $(\lessapprox_R) = ({}_L\lessapprox)$ — Note that *flip-inv.left-Galois*$^{-1}$ = *flip-inv.left-Galois*$^{-1}$
⟨*proof*⟩

end

context *galois*
begin

interpretation *flip-inv* : *galois* $(\ge_R)$ $(\ge_L)$ $r\ l$ ⟨*proof*⟩

context
begin

interpretation *flip* : *galois* $R\ L\ r\ l$ ⟨*proof*⟩

lemma *left-Galois-left-if-left-relI*:
assumes $((\le_L) \Rrightarrow_m (\le_R))\ l$
and $((\le_L) \trianglelefteq_h (\le_R))\ l\ r$
and $x \le_L x'$
shows $x\ {}_L\lessapprox\ l\ x'$
⟨*proof*⟩

corollary *left-Galois-left-if-reflexive-on-if-half-galois-prop-rightI*:
assumes $((\le_L) \Rrightarrow_m (\le_R))\ l$
and $((\le_L) \trianglelefteq_h (\le_R))\ l\ r$
and *reflexive-on* $P\ (\le_L)$
and $P\ x$
shows $x\ {}_L\lessapprox\ l\ x$
⟨*proof*⟩

lemma *left-Galois-left-if-in-codom-if-inflationary-onI*:
assumes $((\le_L) \Rrightarrow_m (\le_R))\ l$
and *inflationary-on* $P\ (\le_L)\ \eta$
and *in-codom* $(\le_L)\ x$
and $P\ x$
shows $x\ {}_L\lessapprox\ l\ x$
⟨*proof*⟩

lemma *left-Galois-left-if-in-codom-if-inflationary-on-in-codomI*:
assumes $((\le_L) \Rrightarrow_m (\le_R))\ l$
and *inflationary-on* (*in-codom* $(\le_L)$) $(\le_L)\ \eta$
and *in-codom* $(\le_L)\ x$
shows $x\ {}_L\lessapprox\ l\ x$
⟨*proof*⟩

lemma *left-Galois-left-if-left-rel-if-inflationary-on-in-fieldI*:
assumes $((\le_L) \Rrightarrow_m (\le_R))\ l$
and *inflationary-on* (*in-field* $(\le_L)$) $(\le_L)\ \eta$

and $x \le_L x$
shows $x \,{}_{L}\lessapprox\, l\ x$
⟨*proof*⟩

lemma *right-left-Galois-if-right-relI*:
assumes $((\le_R) \Rrightarrow_m (\le_L))\ r$
and $y \le_R y'$
shows $r\ y \,{}_{L}\lessapprox\, y'$
⟨*proof*⟩

corollary *right-left-Galois-if-reflexive-onI*:
assumes $((\le_R) \Rrightarrow_m (\le_L))\ r$
and *reflexive-on* $P\ (\le_R)$
and $P\ y$
shows $r\ y \,{}_{L}\lessapprox\, y$
⟨*proof*⟩

lemma *left-Galois-if-right-rel-if-left-GaloisI*:
assumes $((\le_R) \Rrightarrow_m (\le_L))\ r$
and *transitive* $(\le_L)$
and $x \,{}_{L}\lessapprox\, y$
and $y \le_R z$
shows $x \,{}_{L}\lessapprox\, z$
⟨*proof*⟩

lemma *left-Galois-if-left-Galois-if-left-relI*:
assumes *transitive* $(\le_L)$
and $x \le_L y$
and $y \,{}_{L}\lessapprox\, z$
shows $x \,{}_{L}\lessapprox\, z$
⟨*proof*⟩

lemma *left-rel-if-right-Galois-if-left-GaloisI*:
assumes $((\le_R) \,{}_{h}\trianglelefteq\, (\le_L))\ r\ l$
and *transitive* $(\le_L)$
and $x \,{}_{L}\lessapprox\, y$
and $y \,{}_{R}\lessapprox\, z$
shows $x \le_L z$
⟨*proof*⟩

lemma *Dep-Fun-Rel-left-Galois-right-Galois-if-mono-wrt-rel* [*intro*]:
assumes $((\le_L) \Rrightarrow_m (\le_R))\ l$
shows $(({}_{L}\lessapprox) \Rrightarrow ({}_{R}\lessapprox))\ l\ r$
⟨*proof*⟩

lemma *left-ge-Galois-eq-left-Galois-if-in-codom-eq-in-dom-if-symmetric*:
assumes *symmetric* $(\le_L)$
and *in-codom* $(\le_R)$ = *in-dom* $(\le_R)$
shows $({}_{L}\gtrapprox) = ({}_{L}\lessapprox)$ — Note that *flip.right-ge-Galois* = *flip.right-ge-Galois*

⟨*proof*⟩

end

interpretation *flip* : *galois R L r l* ⟨*proof*⟩

lemma *ge-Galois-right-eq-left-Galois-if-symmetric-if-in-codom-eq-in-dom-if-galois-prop*:
assumes $((\le_L) \trianglelefteq (\le_R))\ l\ r$
and *in-codom* $(\le_L)$ = *in-dom* $(\le_L)$
and *symmetric* $(\le_R)$
shows $(\gtrapprox_R) = (_L\lessapprox)$ — Note that *flip.left-Galois*$^{-1}$ = *flip.left-Galois*$^{-1}$
⟨*proof*⟩

interpretation *gp* : *galois-prop* $(_L\lessapprox)$ $(_R\lessapprox)$ *l l* ⟨*proof*⟩

lemma *half-galois-prop-left-left-Galois-right-Galois-if-half-galois-prop-leftI* [*intro*]:
assumes $((\le_L)\ {}_h\trianglelefteq\ (\le_R))\ l\ r$
shows $((_L\lessapprox)\ {}_h\trianglelefteq\ (_R\lessapprox))\ l\ l$
⟨*proof*⟩

lemma *half-galois-prop-right-left-Galois-right-Galois-if-half-galois-prop-rightI* [*intro*]:
assumes $((\le_L) \trianglelefteq_h (\le_R))\ l\ r$
shows $((_L\lessapprox) \trianglelefteq_h (_R\lessapprox))\ l\ l$
⟨*proof*⟩

corollary *galois-prop-left-Galois-right-Galois-if-galois-prop* [*intro*]:
assumes $((\le_L) \trianglelefteq (\le_R))\ l\ r$
shows $((_L\lessapprox) \trianglelefteq (_R\lessapprox))\ l\ l$
⟨*proof*⟩

end

end

theory *Galois*
imports
Galois-Equivalences
Galois-Relator
begin

Summary We define the concept of (partial) Galois connections, Galois equivalences, and the Galois relator. For details refer to [2].

end

Closure Operators

theory *Closure-Operators*
imports

Order-Functions-Base
begin

definition *idempotent-on* P R f ≡ *rel-equivalence-on* P (*rel-map* f R) f

lemma *idempotent-onI* [*intro*]:
assumes $\bigwedge x.\ P\ x \Longrightarrow f\ x \equiv_R f\ (f\ x)$
shows *idempotent-on* P R f
⟨*proof*⟩

lemma *idempotent-onE* [*elim*]:
assumes *idempotent-on* P R f
and $P\ x$
obtains $R\ (f\ (f\ x))\ (f\ x)$ $R\ (f\ x)\ (f\ (f\ x))$
⟨*proof*⟩

lemma *rel-equivalence-on-rel-map-iff-idempotent-on* [*iff*]:
rel-equivalence-on P (*rel-map* f R) f ⟷ *idempotent-on* P R f
⟨*proof*⟩

lemma *bi-related-if-idempotent-onD*:
assumes *idempotent-on* P R f
and $P\ x$
shows $f\ x \equiv_R f\ (f\ x)$
⟨*proof*⟩

definition *idempotent* (R :: $'a \Rightarrow$ -) f ≡ *idempotent-on* ($\top$ :: $'a \Rightarrow$ *bool*) R f

lemma *idempotent-eq-idempotent-on*:
idempotent (R :: $'a \Rightarrow$ -) f = *idempotent-on* ($\top$ :: $'a \Rightarrow$ *bool*) R f
⟨*proof*⟩

lemma *idempotentI* [*intro*]:
assumes $\bigwedge x.\ R\ (f\ (f\ x))\ (f\ x)$
and $\bigwedge x.\ R\ (f\ x)\ (f\ (f\ x))$
shows *idempotent* R f
⟨*proof*⟩

lemma *idempotentE* [*elim*]:
assumes *idempotent* R f
obtains $R\ (f\ (f\ x))\ (f\ x)$ $R\ (f\ x)\ (f\ (f\ x))$
⟨*proof*⟩

lemma *idempotent-on-if-idempotent*:
fixes P :: $'a \Rightarrow$ *bool* **and** R :: $'a \Rightarrow$ -
assumes *idempotent* R f
shows *idempotent-on* P R f
⟨*proof*⟩

definition *closure-operator R f* ≡
$(R \Rrightarrow_m R)$ *f* ∧ *inflationary-on* (*in-field R*) *R f* ∧ *idempotent-on* (*in-field R*) *R f*

lemma *closure-operatorI* [*intro*]:
assumes $(R \Rrightarrow_m R)$ *f*
and *inflationary-on* (*in-field R*) *R f*
and *idempotent-on* (*in-field R*) *R f*
shows *closure-operator R f*
⟨*proof*⟩

lemma *closure-operatorE* [*elim*]:
assumes *closure-operator R f*
obtains $(R \Rrightarrow_m R)$ *f* *inflationary-on* (*in-field R*) *R f*
idempotent-on (*in-field R*) *R f*
⟨*proof*⟩

lemma *mono-wrt-rel-if-closure-operator*:
assumes *closure-operator R f*
shows $(R \Rrightarrow_m R)$ *f*
⟨*proof*⟩

lemma *inflationary-on-in-field-if-closure-operator*:
assumes *closure-operator R f*
shows *inflationary-on* (*in-field R*) *R f*
⟨*proof*⟩

lemma *idempotent-on-in-field-if-closure-operator*:
assumes *closure-operator R f*
shows *idempotent-on* (*in-field R*) *R f*
⟨*proof*⟩

end

theory *Order-Functions*
imports
Order-Functions-Base
Closure-Operators
begin

Summary Basic functions on orders.

end

theory *Order-Functors*
imports
Order-Functors-Base
Order-Equivalences
begin

Summary Functors between orders aka. order-homomorphisms aka. monotone functions.

end

1.5 Orders

theory *Orders*
imports
Equivalence-Relations
Order-Functions
Order-Functors
Partial-Equivalence-Relations
Partial-Orders
Preorders
begin

Summary Basic order-theoretic concepts.

end

1.6 HOL-Basics

theory *HOL-Basics*
imports
LBinary-Relations
LFunctions
Galois
Orders
Predicates
begin

Summary Library on top of HOL axioms, as required for Transport [2]. Requires *only* the HOL axioms, nothing else. Includes:

1. Basic concepts on binary relations, relativised properties, and restricted equalities e.g. *left-total-on* and *eq-restrict*.

2. Basic concepts on functions, relativised properties, and generalised relators, e.g. *injective-on* and *dep-mono-wrt-pred*.

3. Basic concepts on orders and relativised order-theoretic properties, e.g. *partial-equivalence-rel-on*.

4. Galois connections, Galois equivalences, order equivalences, and other related concepts on order functors, e.g. *galois.galois-equivalence*.

5. Basic concepts on predicates.

6. Syntax bundles for HOL `HOL_Syntax_Bundles`.
7. Alignments for concepts that have counterparts in the HOL library - see `HOL_Alignments`.

end

theory *HOL-Mem-Of*
 imports
 HOL.Set
begin

definition *mem-of A x* $\equiv$ $x \in A$
lemma *mem-of-eq* [*simp*]: *mem-of* $\equiv \lambda A\ x.\ x \in A$ ⟨*proof*⟩
lemma *mem-of-iff* [*iff*]: *mem-of A x* $\longleftrightarrow x \in A$ ⟨*proof*⟩

end

1.7 Relation Syntax

theory *HOL-Syntax-Bundles-Relations*
 imports *HOL.Relation*
begin

bundle *HOL-relation-syntax*
begin
notation *relcomp* (**infixr** *O 75*)
notation *relcompp* (**infixr** *OO 75*)
notation *converse* ((-$^{-1}$) [*1000*] *999*)
notation *conversep* ((-$^{-1-1}$) [*1000*] *1000*)
notation (*ASCII*)
 converse ((-^−*1*) [*1000*] *999*) **and**
 conversep ((-^−−*1*) [*1000*] *1000*)
end
bundle *no-HOL-relation-syntax*
begin
no-notation *relcomp* (**infixr** *O 75*)
no-notation *relcompp* (**infixr** *OO 75*)
no-notation *converse* ((-$^{-1}$) [*1000*] *999*)
no-notation *conversep* ((-$^{-1-1}$) [*1000*] *1000*)
no-notation (*ASCII*)
 converse ((-^−*1*) [*1000*] *999*) **and**
 conversep ((-^−−*1*) [*1000*] *1000*)
end

end

1.7.1 Alignment With Definitions from HOL.Main

theory *HOL-Alignment-Binary-Relations*
  imports
    Main
    HOL-Mem-Of
    HOL-Syntax-Bundles-Relations
    LBinary-Relations
begin

unbundle *no-HOL-relation-syntax*

named-theorems *HOL-bin-rel-alignment*

Properties

Antisymmetric **overloading**
  antisymmetric-on-set ≡ *antisymmetric-on* :: *'a set* ⇒ (*'a* ⇒ *'a* ⇒ *bool*) ⇒ *bool*
begin
  definition *antisymmetric-on-set* (*S* :: *'a set*) :: (*'a* ⇒ -) ⇒ - ≡
    antisymmetric-on (*mem-of S*)
end

lemma *antisymmetric-on-set-eq-antisymmetric-on-pred* [*simp*]:
  (*antisymmetric-on* (*S* :: *'a set*) :: (*'a* ⇒ -) ⇒ *bool*) =
    antisymmetric-on (*mem-of S*)
  ⟨*proof*⟩

lemma *antisymmetric-on-set-iff-antisymmetric-on-pred* [*iff*]:
  antisymmetric-on (*S* :: *'a set*) (*R* :: *'a* ⇒ -) ⟷ *antisymmetric-on* (*mem-of S*)
R
  ⟨*proof*⟩

lemma *antisymp-eq-antisymmetric* [*HOL-bin-rel-alignment*]:
  antisymp = *antisymmetric*
  ⟨*proof*⟩

Injective **overloading**
  rel-injective-on-set ≡ *rel-injective-on* :: *'a set* ⇒ (*'a* ⇒ *'b* ⇒ *bool*) ⇒ *bool*
  rel-injective-at-set ≡ *rel-injective-at* :: *'a set* ⇒ (*'b* ⇒ *'a* ⇒ *bool*) ⇒ *bool*
begin
  definition *rel-injective-on-set* (*S* :: *'a set*) :: (*'a* ⇒ -) ⇒ - ≡
    rel-injective-on (*mem-of S*)
  definition *rel-injective-at-set* (*S* :: *'a set*) :: (*'b* ⇒ *'a* ⇒ -) ⇒ - ≡
    rel-injective-at (*mem-of S*)
end

lemma *rel-injective-on-set-eq-rel-injective-on-pred* [*simp*]:
  (*rel-injective-on* (*S* :: *'a set*) :: (*'a* ⇒ -) ⇒ *bool*) =

rel-injective-on (*mem-of* *S*)
⟨*proof*⟩

lemma *rel-injective-on-set-iff-rel-injective-on-pred* [*iff*]:
rel-injective-on (*S* :: *'a set*) (*R* :: *'a* ⇒ -) ⟷ *rel-injective-on* (*mem-of* *S*) *R*
⟨*proof*⟩

lemma *rel-injective-at-set-eq-rel-injective-at-pred* [*simp*]:
(*rel-injective-at* (*S* :: *'a set*) :: (*'b* ⇒ *'a* ⇒ *bool*) ⇒ *bool*) =
rel-injective-at (*mem-of* *S*)
⟨*proof*⟩

lemma *rel-injective-at-set-iff-rel-injective-at-pred* [*iff*]:
rel-injective-at (*S* :: *'a set*) (*R* :: *'b* ⇒ *'a* ⇒ *bool*) ⟷ *rel-injective-at* (*mem-of* *S*) *R*
⟨*proof*⟩

lemma *left-unique-eq-rel-injective* [*HOL-bin-rel-alignment*]:
left-unique = *rel-injective*
⟨*proof*⟩

Irreflexive **overloading**
irreflexive-on-set ≡ *irreflexive-on* :: *'a set* ⇒ (*'a* ⇒ *'a* ⇒ *bool*) ⇒ *bool*
begin
definition *irreflexive-on-set* (*S* :: *'a set*) :: (*'a* ⇒ -) ⇒ - ≡
irreflexive-on (*mem-of* *S*)
end

lemma *irreflexive-on-set-eq-irreflexive-on-pred* [*simp*]:
(*irreflexive-on* (*S* :: *'a set*) :: (*'a* ⇒ -) ⇒ *bool*) =
irreflexive-on (*mem-of* *S*)
⟨*proof*⟩

lemma *irreflexive-on-set-iff-irreflexive-on-pred* [*iff*]:
irreflexive-on (*S* :: *'a set*) (*R* :: *'a* ⇒ -) ⟷
irreflexive-on (*mem-of* *S*) *R*
⟨*proof*⟩

lemma *irreflp-on-eq-irreflexive-on* [*HOL-bin-rel-alignment*]:
irreflp-on = *irreflexive-on*
⟨*proof*⟩

lemma *irreflp-eq-irreflexive* [*HOL-bin-rel-alignment*]: *irreflp* = *irreflexive*
⟨*proof*⟩

Left-Total **overloading**
left-total-on-set ≡ *left-total-on* :: *'a set* ⇒ (*'a* ⇒ *'b* ⇒ *bool*) ⇒ *bool*
begin
definition *left-total-on-set* (*S* :: *'a set*) :: (*'a* ⇒ -) ⇒ - ≡

left-total-on (*mem-of S*)
end

lemma *left-total-on-set-eq-left-total-on-pred* [*simp*]:
(*left-total-on* (*S* :: *'a set*) :: (*'a* ⇒ -) ⇒ *bool*) =
left-total-on (*mem-of S*)
⟨*proof*⟩

lemma *left-total-on-set-iff-left-total-on-pred* [*iff*]:
left-total-on (*S* :: *'a set*) (*R* :: *'a* ⇒ -) ⟷ *left-total-on* (*mem-of S*) *R*
⟨*proof*⟩

lemma *Transfer-left-total-eq-left-total* [*HOL-bin-rel-alignment*]:
Transfer.left-total = *Binary-Relations-Left-Total.left-total*
⟨*proof*⟩

Reflexive **overloading**
reflexive-on-set ≡ *reflexive-on* :: *'a set* ⇒ (*'a* ⇒ *'a* ⇒ *bool*) ⇒ *bool*
begin
definition *reflexive-on-set* (*S* :: *'a set*) :: (*'a* ⇒ -) ⇒ - ≡
reflexive-on (*mem-of S*)
end

lemma *reflexive-on-set-eq-reflexive-on-pred* [*simp*]:
(*reflexive-on* (*S* :: *'a set*) :: (*'a* ⇒ *'a* ⇒ *bool*) ⇒ *bool*) =
reflexive-on (*mem-of S*)
⟨*proof*⟩

lemma *reflexive-on-set-iff-reflexive-on-pred* [*iff*]:
reflexive-on (*S* :: *'a set*) (*R* :: *'a* ⇒ *'a* ⇒ *bool*) ⟷
reflexive-on (*mem-of S*) *R*
⟨*proof*⟩

lemma *reflp-on-eq-reflexive-on* [*HOL-bin-rel-alignment*]:
reflp-on = *reflexive-on*
⟨*proof*⟩

lemma *reflp-eq-reflexive* [*HOL-bin-rel-alignment*]: *reflp* = *reflexive*
⟨*proof*⟩

Right-Unique **overloading**
right-unique-on-set ≡ *right-unique-on* :: *'a set* ⇒ (*'a* ⇒ *'b* ⇒ *bool*) ⇒ *bool*
right-unique-at-set ≡ *right-unique-at* :: *'a set* ⇒ (*'b* ⇒ *'a* ⇒ *bool*) ⇒ *bool*
begin
definition *right-unique-on-set* (*S* :: *'a set*) :: (*'a* ⇒ -) ⇒ - ≡
right-unique-on (*mem-of S*)
definition *right-unique-at-set* (*S* :: *'a set*) :: (*'b* ⇒ *'a* ⇒ -) ⇒ - ≡
right-unique-at (*mem-of S*)
end

lemma *right-unique-on-set-eq-right-unique-on-pred* [*simp*]:
(*right-unique-on* (*S* :: *'a set*) :: (*'a* ⇒ *-*) ⇒ *bool*) =
right-unique-on (*mem-of S*)
⟨*proof*⟩

lemma *right-unique-on-set-iff-right-unique-on-pred* [*iff*]:
right-unique-on (*S* :: *'a set*) (*R* :: *'a* ⇒ *-*) ⟷ *right-unique-on* (*mem-of S*) *R*
⟨*proof*⟩

lemma *right-unique-at-set-eq-right-unique-at-pred* [*simp*]:
(*right-unique-at* (*S* :: *'a set*) :: (*'b* ⇒ *'a* ⇒ *bool*) ⇒ *bool*) =
right-unique-at (*mem-of S*)
⟨*proof*⟩

lemma *right-unique-at-set-iff-right-unique-at-pred* [*iff*]:
right-unique-at (*S* :: *'a set*) (*R* :: *'b* ⇒ *'a* ⇒ *bool*) ⟷ *right-unique-at* (*mem-of S*) *R*
⟨*proof*⟩

lemma *Transfer-right-unique-eq-right-unique* [*HOL-bin-rel-alignment*]:
Transfer.right-unique = *Binary-Relations-Right-Unique.right-unique*
⟨*proof*⟩

Surjective **overloading**
rel-surjective-at-set ≡ *rel-surjective-at* :: *'a set* ⇒ (*'b* ⇒ *'a* ⇒ *bool*) ⇒ *bool*
begin
definition *rel-surjective-at-set* (*S* :: *'a set*) :: (*'b* ⇒ *'a* ⇒ *-*) ⇒ *-* ≡
rel-surjective-at (*mem-of S*)
end

lemma *rel-surjective-at-set-eq-rel-surjective-at-pred* [*simp*]:
(*rel-surjective-at* (*S* :: *'a set*) :: (*'b* ⇒ *'a* ⇒ *-*) ⇒ *bool*) =
rel-surjective-at (*mem-of S*)
⟨*proof*⟩

lemma *rel-surjective-at-set-iff-rel-surjective-at-pred* [*iff*]:
rel-surjective-at (*S* :: *'a set*) (*R* :: *'b* ⇒ *'a* ⇒ *-*) ⟷ *rel-surjective-at* (*mem-of S*) *R*
⟨*proof*⟩

lemma *Transfer-right-total-eq-rel-surjective* [*HOL-bin-rel-alignment*]:
Transfer.right-total = *rel-surjective*
⟨*proof*⟩

Symmetric **overloading**
symmetric-on-set ≡ *symmetric-on* :: *'a set* ⇒ (*'a* ⇒ *'a* ⇒ *bool*) ⇒ *bool*
begin
definition *symmetric-on-set* (*S* :: *'a set*) :: (*'a* ⇒ *-*) ⇒ *-* ≡

symmetric-on (*mem-of S*)
end

lemma *symmetric-on-set-eq-symmetric-on-pred* [*simp*]:
(*symmetric-on* (*S* :: *'a set*) :: (*'a* ⇒ *'a* ⇒ *bool*) ⇒ *bool*) =
symmetric-on (*mem-of S*)
⟨*proof*⟩

lemma *symmetric-on-set-iff-symmetric-on-pred* [*iff*]:
symmetric-on (*S* :: *'a set*) (*R* :: *'a* ⇒ *'a* ⇒ *bool*) ⟷
symmetric-on (*mem-of S*) *R*
⟨*proof*⟩

lemma *symp-eq-symmetric* [*HOL-bin-rel-alignment*]: *symp* = *symmetric*
⟨*proof*⟩

Transitive **overloading**
transitive-on-set ≡ *transitive-on* :: *'a set* ⇒ (*'a* ⇒ *'a* ⇒ *bool*) ⇒ *bool*
begin
definition *transitive-on-set* (*S* :: *'a set*) :: (*'a* ⇒ -) ⇒ - ≡
transitive-on (*mem-of S*)
end

lemma *transitive-on-set-eq-transitive-on-pred* [*simp*]:
(*transitive-on* (*S* :: *'a set*) :: (*'a* ⇒ *'a* ⇒ *bool*) ⇒ *bool*) =
transitive-on (*mem-of S*)
⟨*proof*⟩

lemma *transitive-on-set-iff-transitive-on-pred* [*iff*]:
transitive-on (*S* :: *'a set*) (*R* :: *'a* ⇒ *'a* ⇒ *bool*) ⟷
transitive-on (*mem-of S*) *R*
⟨*proof*⟩

lemma *transp-eq-transitive* [*HOL-bin-rel-alignment*]: *transp* = *transitive*
⟨*proof*⟩

Functions **lemma** *relcompp-eq-rel-comp* [*HOL-bin-rel-alignment*]: *relcompp* = *rel-comp*
⟨*proof*⟩

lemma *conversep-eq-rel-inv* [*HOL-bin-rel-alignment*]: *conversep* = *rel-inv*
⟨*proof*⟩

lemma *Domainp-eq-in-dom* [*HOL-bin-rel-alignment*]: *Domainp* = *in-dom*
⟨*proof*⟩

lemma *Rangep-eq-in-codom* [*HOL-bin-rel-alignment*]: *Rangep* = *in-codom*
⟨*proof*⟩

overloading
restrict-left-set ≡ *restrict-left* :: $('a \Rightarrow 'b \Rightarrow bool) \Rightarrow ('a\ set) \Rightarrow 'a \Rightarrow 'b \Rightarrow bool$
begin
definition *restrict-left-set* $(R :: 'a \Rightarrow -)\ (S :: 'a\ set) \equiv R\restriction_{mem\text{-}of\ S}$
end

lemma *restrict-left-set-eq-restrict-left-pred* [*simp*]:
$(R\restriction_{S\ ::\ 'a\ set} :: 'a \Rightarrow -) = R\restriction_{mem\text{-}of\ S}$
⟨*proof*⟩

lemma *restrict-left-set-iff-restrict-left-pred* [*iff*]:
$(R\restriction_{S\ ::\ 'a\ set} :: 'a \Rightarrow -)\ x\ y \longleftrightarrow R\restriction_{mem\text{-}of\ S}\ x\ y$
⟨*proof*⟩

Restricted Equality **lemma** *eq-onp-eq-eq-restrict* [*HOL-bin-rel-alignment*]:
eq-onp = *eq-restrict*
⟨*proof*⟩

overloading
eq-restrict-set ≡ *eq-restrict* :: $'a\ set \Rightarrow 'a \Rightarrow 'a \Rightarrow bool$
begin
definition *eq-restrict-set* $(S :: 'a\ set) \equiv ((=_{mem\text{-}of\ S}) :: 'a \Rightarrow -)$
end

lemma *eq-restrict-set-eq-eq-restrict-pred* [*simp*]:
$((=_{S\ ::\ 'a\ set}) :: 'a \Rightarrow -) = (=_{mem\text{-}of\ S})$
⟨*proof*⟩

lemma *eq-restrict-set-iff-eq-restrict-pred* [*iff*]:
$(x :: 'a) =_{(S\ ::\ 'a\ set)} y \longleftrightarrow x =_{mem\text{-}of\ S} y$
⟨*proof*⟩

end

1.7.2 Function Syntax

theory *HOL-Syntax-Bundles-Functions*
imports *HOL.Fun*
begin

bundle *HOL-function-syntax*
begin
notation *comp* (**infixl** ∘ *55*)
end
bundle *no-HOL-function-syntax*
begin
no-notation *comp* (**infixl** ∘ *55*)
end

end

1.7.3 Alignment With Definitions from HOL.Main

theory *HOL-Alignment-Functions*
imports
HOL-Alignment-Binary-Relations
HOL-Syntax-Bundles-Functions
LFunctions
begin

unbundle *no-HOL-function-syntax*

named-theorems *HOL-fun-alignment*

Functions

Bijection **overloading**
bijection-on-set ≡ *bijection-on* :: *'a set* ⇒ *'b set* ⇒ (*'a* ⇒ *'b*) ⇒ (*'b* ⇒ *'a*) ⇒ *bool*
begin
definition *bijection-on-set* (*S* :: *'a set*) (*S'* :: *'b set*) :: (*'a* ⇒ *'b*) ⇒ (*'b* ⇒ *'a*) ⇒ *bool* ≡
bijection-on (*mem-of S*) (*mem-of S'*)
end

lemma *bijection-on-set-eq-bijection-on-pred* [*simp*]:
(*bijection-on* (*S* :: *'a set*) (*S'* :: *'b set*) :: (*'a* ⇒ *'b*) ⇒ -) =
bijection-on (*mem-of S*) (*mem-of S'*)
⟨*proof*⟩

lemma *bijection-on-set-iff-bijection-on-pred* [*iff*]:
bijection-on (*S* :: *'a set*) (*S'* :: *'b set*) (*f* :: *'a* ⇒ *'b*) *g* ⟷
bijection-on (*mem-of S*) (*mem-of S'*) *f g*
⟨*proof*⟩

lemma *bij-betw-bijection-onE*:
assumes *bij-betw f S S'*
obtains *g* **where** *bijection-on S S' f g*
⟨*proof*⟩

lemma *bij-betw-if-bijection-on*:
assumes *bijection-on S S' f g*
shows *bij-betw f S S'*
⟨*proof*⟩

corollary *bij-betw-iff-ex-bijection-on* [*HOL-fun-alignment*]:
bij-betw f S S' ⟷ (∃ *g*. *bijection-on S S' f g*)

⟨*proof*⟩

Injective **overloading**
injective-on-set ≡ *injective-on* :: *'a set* ⇒ (*'a* ⇒ *'b*) ⇒ *bool*
begin
definition *injective-on-set* (*S* :: *'a set*) :: (*'a* ⇒ *'b*) ⇒ *bool* ≡
injective-on (*mem-of S*)
end

lemma *injective-on-set-eq-injective-on-pred* [*simp*]:
(*injective-on* (*S* :: *'a set*) :: (*'a* ⇒ *'b*) ⇒ -) = *injective-on* (*mem-of S*)
⟨*proof*⟩

lemma *injective-on-set-iff-injective-on-pred* [*iff*]:
injective-on (*S* :: *'a set*) (*f* :: *'a* ⇒ *'b*) ⟷ *injective-on* (*mem-of S*) *f*
⟨*proof*⟩

lemma *inj-on-iff-injective-on* [*HOL-fun-alignment*]: *inj-on f P* ⟷ *injective-on P f*
⟨*proof*⟩

lemma *inj-eq-injective* [*HOL-fun-alignment*]: *inj* = *injective*
⟨*proof*⟩

Inverse **overloading**
inverse-on-set ≡ *inverse-on* :: *'a set* ⇒ (*'a* ⇒ *'b*) ⇒ (*'b* ⇒ *'a*) ⇒ *bool*
begin
definition *inverse-on-set* (*S* :: *'a set*) :: (*'a* ⇒ *'b*) ⇒ - ≡
inverse-on (*mem-of S*)
end

lemma *inverse-on-set-eq-inverse-on-pred* [*simp*]:
(*inverse-on* (*S* :: *'a set*) :: (*'a* ⇒ *'b*) ⇒ -) = *inverse-on* (*mem-of S*)
⟨*proof*⟩

lemma *inverse-on-set-iff-inverse-on-pred* [*iff*]:
inverse-on (*S* :: *'a set*) (*f* :: *'a* ⇒ *'b*) *g* ⟷ *inverse-on* (*mem-of S*) *f g*
⟨*proof*⟩

Monotone **lemma** *monotone-on-eq-mono-wrt-rel-restrict-left-right* [*HOL-fun-alignment*]:
monotone-on S R = *mono-wrt-rel* ($R\restriction_{S}\upharpoonleft_{S}$)
⟨*proof*⟩

lemma *monotone-eq-mono-wrt-rel* [*HOL-fun-alignment*]: *monotone* = *mono-wrt-rel*
⟨*proof*⟩

lemma *pred-fun-eq-mono-wrt-pred* [*HOL-fun-alignment*]: *pred-fun* = *mono-wrt-pred*
⟨*proof*⟩

lemma *Fun-mono-eq-mono* [*HOL-fun-alignment*]: *Fun.mono* = *mono*
⟨*proof*⟩

lemma *Fun-antimono-eq-antimono* [*HOL-fun-alignment*]: *Fun.antimono* = *antimono*
⟨*proof*⟩

Surjective **overloading**
surjective-at-set ≡ *surjective-at* :: ′*a set* ⇒ (′*b* ⇒ ′*a*) ⇒ *bool*
begin
definition *surjective-at-set* (*S* :: ′*a set*) :: (′*b* ⇒ ′*a*) ⇒ *bool* ≡
surjective-at (*mem-of S*)
end

lemma *surjective-at-set-eq-surjective-at-pred* [*simp*]:
(*surjective-at* (*S* :: ′*a set*) :: (′*b* ⇒ ′*a*) ⇒ -) = *surjective-at* (*mem-of S*)
⟨*proof*⟩

lemma *surjective-at-set-iff-surjective-at-pred* [*iff*]:
surjective-at (*S* :: ′*a set*) (*f* :: ′*b* ⇒ ′*a*) ⟷ *surjective-at* (*mem-of S*) *f*
⟨*proof*⟩

lemma *surj-eq-surjective* [*HOL-fun-alignment*]: *surj* = *surjective*
⟨*proof*⟩

Functions **lemma** *Fun-id-eq-id* [*HOL-fun-alignment*]: *Fun.id* = *Functions-Base.id*
⟨*proof*⟩

lemma *Fun-comp-eq-comp* [*HOL-fun-alignment*]: *Fun.comp* = *Functions-Base.comp*
⟨*proof*⟩

lemma *map-fun-eq-fun-map* [*HOL-fun-alignment*]: *map-fun* = *fun-map*
⟨*proof*⟩

Relators **lemma** *rel-fun-eq-Fun-Rel-rel* [*HOL-fun-alignment*]: *rel-fun* = *Fun-Rel-rel*
⟨*proof*⟩

end

1.8 Order Syntax

theory *HOL-Syntax-Bundles-Orders*
imports *HOL.Orderings*
begin

bundle *HOL-order-syntax*

begin
notation
  less-eq ('(≤')) **and**
  less-eq ((-/ ≤ -) [*51*, *51*] *50*) **and**
  less ('(<')) **and**
  less ((-/ < -) [*51*, *51*] *50*)
notation (*input*) *greater-eq* (**infix** ≥ *50*)
notation (*input*) *greater* (**infix** > *50*)
notation (*ASCII*)
  less-eq ('(<=')) **and**
  less-eq ((-/ <= -) [*51*, *51*] *50*)
notation (*input*) *greater-eq* (**infix** >= *50*)
end
bundle *no-HOL-order-syntax*
begin
no-notation
  less-eq ('(≤')) **and**
  less-eq ((-/ ≤ -) [*51*, *51*] *50*) **and**
  less ('(<')) **and**
  less ((-/ < -) [*51*, *51*] *50*)
no-notation (*input*) *greater-eq* (**infix** ≥ *50*)
no-notation (*input*) *greater* (**infix** > *50*)
no-notation (*ASCII*)
  less-eq ('(<=')) **and**
  less-eq ((-/ <= -) [*51*, *51*] *50*)
no-notation (*input*) *greater-eq* (**infix** >= *50*)
end

end

1.8.1 Alignment With Definitions from HOL

theory *HOL-Alignment-Orders*
  imports
    HOL−Library.Preorder
    HOL-Alignment-Binary-Relations
    HOL-Syntax-Bundles-Orders
    Orders
begin

named-theorems *HOL-order-alignment*

Functions

Bi-Related **lemma** (**in** *preorder-equiv*) *equiv-eq-bi-related* [*HOL-order-alignment*]:
  equiv = *bi-related* (≤)
  ⟨*proof*⟩

Inflationary **overloading**
inflationary-on-set ≡ *inflationary-on* :: *'a set* ⇒ (*'a* ⇒ *'a* ⇒ *bool*) ⇒ (*'a* ⇒ *'a*) ⇒ *bool*
begin
definition *inflationary-on-set* (*S* :: *'a set*) :: (*'a* ⇒ -) ⇒ - ≡
inflationary-on (*mem-of S*)
end

lemma *inflationary-on-set-eq-inflationary-on-pred* [*simp*]:
(*inflationary-on* (*S* :: *'a set*) :: (*'a* ⇒ -) ⇒ -) = *inflationary-on* (*mem-of S*)
⟨*proof*⟩

lemma *inflationary-on-set-iff-inflationary-on-pred* [*iff*]:
inflationary-on (*S* :: *'a set*) (*R* :: *'a* ⇒ -) *f* ⟷ *inflationary-on* (*mem-of S*) *R f*
⟨*proof*⟩

Terms like *deflationary-on*, *rel-equivalence-on*, and *idempotent-on* are automatically overloaded. One can get similar correspondence lemmas by unfolding the corresponding definitional theorems, e.g. *deflationary-on ?P ?R* = *inflationary-on ?P* $?R^{-1}$.

Properties

Equivalence Relations **lemma** *equiv-eq-equivalence-rel* [*HOL-order-alignment*]:
equivp = *equivalence-rel*
⟨*proof*⟩

Partial Equivalence Relations **lemma** *part-equiv-eq-partial-equivalence-rel-if-rel* [*HOL-order-alignment*]:
assumes *R x y*
shows *part-equivp R* = *partial-equivalence-rel R*
⟨*proof*⟩

Partial Orders **lemma** (**in** *order*) *partial-order* [*HOL-order-alignment*]: *partial-order* (≤)
⟨*proof*⟩

Preorders **lemma** (**in** *partial-preordering*) *preorder* [*HOL-order-alignment*]: *preorder* (≤)
⟨*proof*⟩

lemma *partial-preordering-eq* [*HOL-order-alignment*]:
partial-preordering = *Preorders.preorder*
⟨*proof*⟩

end

1.9 HOL Alignments

theory *HOL-Alignments*
  imports
    HOL-Alignment-Binary-Relations
    HOL-Alignment-Functions
    HOL-Alignment-Orders
begin

Summary Alignment of concepts with HOL counterparts

end

1.9.1 Alignment With Definitions from HOL-Algebra

theory *HOL-Algebra-Alignment-Orders*
  imports
    HOL−Algebra.Order
    HOL-Alignment-Orders
begin

named-theorems *HOL-Algebra-order-alignment*

context *equivalence*
begin

lemma *reflexive-on-carrier* [*HOL-Algebra-order-alignment*]:
  reflexive-on (*carrier S*) (.=)
  ⟨*proof*⟩

lemma *transitive-on-carrier* [*HOL-Algebra-order-alignment*]:
  transitive-on (*carrier S*) (.=)
  ⟨*proof*⟩

lemma *preorder-on-carrier* [*HOL-Algebra-order-alignment*]:
  preorder-on (*carrier S*) (.=)
  ⟨*proof*⟩

lemma *symmetric-on-carrier* [*HOL-Algebra-order-alignment*]:
  symmetric-on (*carrier S*) (.=)
  ⟨*proof*⟩

lemma *partial-equivalence-rel-on-carrier* [*HOL-Algebra-order-alignment*]:
  partial-equivalence-rel-on (*carrier S*) (.=)
  ⟨*proof*⟩

lemma *equivalence-rel-on-carrier* [*HOL-Algebra-order-alignment*]:
  equivalence-rel-on (*carrier S*) (.=)
  ⟨*proof*⟩

end

lemma *equivalence-iff-equivalence-rel-on-carrier* [*HOL-Algebra-order-alignment*]:
equivalence $S \longleftrightarrow$ *equivalence-rel-on* (*carrier* S) $(.=_S)$
⟨*proof*⟩

context *partial-order*
begin

lemma *reflexive-on-carrier* [*HOL-Algebra-order-alignment*]:
reflexive-on (*carrier* L) $(\sqsubseteq)$
⟨*proof*⟩

lemma *transitive-on-carrier* [*HOL-Algebra-order-alignment*]:
transitive-on (*carrier* L) $(\sqsubseteq)$
⟨*proof*⟩

lemma *preorder-on-carrier* [*HOL-Algebra-order-alignment*]:
preorder-on (*carrier* L) $(\sqsubseteq)$
⟨*proof*⟩

lemma *antisymmetric-on-carrier* [*HOL-Algebra-order-alignment*]:
antisymmetric-on (*carrier* L) $(\sqsubseteq)$
⟨*proof*⟩

lemma *partial-order-on-carrier* [*HOL-Algebra-order-alignment*]:
partial-order-on (*carrier* L) $(\sqsubseteq)$
⟨*proof*⟩

end

end

1.9.2 Alignment With Definitions from HOL-Algebra

theory *HOL-Algebra-Alignment-Galois*
imports
HOL−Algebra.Galois-Connection
HOL-Algebra-Alignment-Orders
Galois
begin

named-theorems *HOL-Algebra-galois-alignment*

context *galois-connection*
begin

context

fixes L R l r
defines $L \equiv (\sqsubseteq_{\mathcal{X}})\restriction_{carrier\ \mathcal{X}}\upharpoonleft_{carrier\ \mathcal{X}}$ **and** $R \equiv (\sqsubseteq_{\mathcal{Y}})\restriction_{carrier\ \mathcal{Y}}\upharpoonleft_{carrier\ \mathcal{Y}}$
and $l \equiv \pi^*$ **and** $r \equiv \pi_*$
notes *defs*[*simp*] = *L-def R-def l-def r-def* **and** *restrict-right-eq*[*simp*]
and *restrict-leftI*[*intro*!] *restrict-leftE*[*elim*!]
begin

interpretation *galois* L R l r ⟨*proof*⟩

lemma *mono-wrt-rel-lower* [*HOL-Algebra-galois-alignment*]: $(L \Rrightarrow_m R)\ l$
⟨*proof*⟩

lemma *mono-wrt-rel-upper* [*HOL-Algebra-galois-alignment*]: $(R \Rrightarrow_m L)\ r$
⟨*proof*⟩

lemma *half-galois-prop-left* [*HOL-Algebra-galois-alignment*]: $(L\ {}_h\!\trianglelefteq\ R)\ l\ r$
⟨*proof*⟩

lemma *half-galois-prop-right* [*HOL-Algebra-galois-alignment*]: $(L \trianglelefteq_h R)\ l\ r$
⟨*proof*⟩

lemma *galois-prop* [*HOL-Algebra-galois-alignment*]: $(L \trianglelefteq R)\ l\ r$
⟨*proof*⟩

lemma *galois-connection* [*HOL-Algebra-galois-alignment*]: $(L \dashv R)\ l\ r$
⟨*proof*⟩

end
end

context *galois-bijection*
begin

context
fixes L R l r
defines $L \equiv (\sqsubseteq_{\mathcal{X}})\restriction_{carrier\ \mathcal{X}}\upharpoonleft_{carrier\ \mathcal{X}}$ **and** $R \equiv (\sqsubseteq_{\mathcal{Y}})\restriction_{carrier\ \mathcal{Y}}\upharpoonleft_{carrier\ \mathcal{Y}}$
and $l \equiv \pi^*$ **and** $r \equiv \pi_*$
notes *defs*[*simp*] = *L-def R-def l-def r-def* **and** *restrict-right-eq*[*simp*]
and *restrict-leftI*[*intro*!] *restrict-leftE*[*elim*!] *in-codom-restrict-leftE*[*elim*!]
begin

interpretation *galois* R L r l ⟨*proof*⟩

lemma *half-galois-prop-left-right-left* [*HOL-Algebra-galois-alignment*]:
$(R\ {}_h\!\trianglelefteq\ L)\ r\ l$
⟨*proof*⟩

lemma *half-galois-prop-right-right-left* [*HOL-Algebra-galois-alignment*]:
$(R \trianglelefteq_h L)\ r\ l$

⟨*proof*⟩

lemma *prop-right-right-left* [*HOL-Algebra-galois-alignment*]: $(R \trianglelefteq L)$ *r l*
⟨*proof*⟩

lemma *galois-equivalence* [*HOL-Algebra-galois-alignment*]: $(L \equiv_G R)$ *l r*
⟨*proof*⟩

end
end

end

1.10 HOL-Algebra Alignments

theory *HOL-Algebra-Alignments*
imports
HOL-Algebra-Alignment-Galois
HOL-Algebra-Alignment-Orders
begin

Summary Alignment of concepts with HOL-Algebra counterparts

end

1.11 HOL Syntax Bundles

1.11.1 Basic Syntax

theory *HOL-Syntax-Bundles-Base*
imports *HOL-Basics-Base*
begin

bundle *HOL-ascii-syntax*
begin
notation (*ASCII*)
Not (~ - [*40*] *40*) **and**
conj (**infixr** & *35*) **and**
disj (**infixr** | *30*) **and**
implies (**infixr** −−> *25*) **and**
not-equal (**infixl** ~= *50*)
syntax *-Let* :: [*letbinds*, *'a*] ⇒ *'a* ((*let* (-)/ *in* (-)) *10*)
end
bundle *no-HOL-ascii-syntax*
begin
no-notation (*ASCII*)
Not (~ - [*40*] *40*) **and**
conj (**infixr** & *35*) **and**

disj (**infixr** | *30*) **and**
implies (**infixr** −−> *25*) **and**
not-equal (**infixl** ~= *50*)
no-syntax *-Let* :: [*letbinds*, *'a*] ⇒ *'a* ((*let* (-)/ *in* (-)) *10*)
end

end

1.11.2 Group Syntax

theory *HOL-Syntax-Bundles-Groups*
imports *HOL.Groups*
begin

bundle *HOL-groups-syntax*
begin
notation *Groups.zero* (*0*)
notation *Groups.one* (*1*)
notation *Groups.plus* (**infixl** + *65*)
notation *Groups.minus* (**infixl** − *65*)
notation *Groups.uminus* (− - [*81*] *80*)
notation *Groups.times* (**infixl** ∗ *70*)
notation *abs* (|-|)
end
bundle *no-HOL-groups-syntax*
begin
no-notation *Groups.zero* (*0*)
no-notation *Groups.one* (*1*)
no-notation *Groups.plus* (**infixl** + *65*)
no-notation *Groups.minus* (**infixl** − *65*)
no-notation *Groups.uminus* (− - [*81*] *80*)
no-notation *Groups.times* (**infixl** ∗ *70*)
no-notation *abs* (|-|)
end

end

theory *HOL-Syntax-Bundles*
imports
HOL-Syntax-Bundles-Base
HOL-Syntax-Bundles-Functions
HOL-Syntax-Bundles-Groups
HOL-Syntax-Bundles-Lattices
HOL-Syntax-Bundles-Orders
HOL-Syntax-Bundles-Relations
begin

Summary Bundles to enable and disable syntax from HOL.

end

Chapter 2

Transport

2.1 Basic Setup

theory *Transport-Base*
imports
Galois-Equivalences
Galois-Relator
begin

Summary Basic setup for commonly used concepts in Transport, including a suitable locale.

locale *transport* = *galois L R l r*
for $L :: 'a \Rightarrow 'a \Rightarrow bool$
and $R :: 'b \Rightarrow 'b \Rightarrow bool$
and $l :: 'a \Rightarrow 'b$
and $r :: 'b \Rightarrow 'a$
begin

2.1.1 Ordered Galois Connections

definition *preorder-galois-connection* ≡
$((\le_L) \dashv (\le_R))\ l\ r$
$\land$ *preorder-on* (*in-field* $(\le_L)$) $(\le_L)$
$\land$ *preorder-on* (*in-field* $(\le_R)$) $(\le_R)$

notation *transport.preorder-galois-connection* (**infix** $\dashv_{pre}$ *50*)

lemma *preorder-galois-connectionI* [*intro*]:
assumes $((\le_L) \dashv (\le_R))\ l\ r$
and *preorder-on* (*in-field* $(\le_L)$) $(\le_L)$
and *preorder-on* (*in-field* $(\le_R)$) $(\le_R)$
shows $((\le_L) \dashv_{pre} (\le_R))\ l\ r$
⟨*proof*⟩

lemma *preorder-galois-connectionE* [*elim*]:

assumes $((\le_L)\ \dashv_{pre}\ (\le_R))$ *l r*
obtains $((\le_L)\ \dashv\ (\le_R))$ *l r preorder-on* (*in-field* $(\le_L)$) $(\le_L)$
preorder-on (*in-field* $(\le_R)$) $(\le_R)$
⟨*proof*⟩

context
begin

interpretation *t* : *transport S T f g* **for** *S T f g* ⟨*proof*⟩

lemma *rel-inv-preorder-galois-connection-eq-preorder-galois-connection-rel-inv* [*simp*]:
$((\le_R)\ \dashv_{pre}\ (\le_L))^{-1} = ((\ge_L)\ \dashv_{pre}\ (\ge_R))$
⟨*proof*⟩

end

corollary *preorder-galois-connection-rel-inv-iff-preorder-galois-connection* [*iff*]:
$((\ge_L)\ \dashv_{pre}\ (\ge_R))\ l\ r \longleftrightarrow ((\le_R)\ \dashv_{pre}\ (\le_L))\ r\ l$
⟨*proof*⟩

definition *partial-equivalence-rel-galois-connection* ≡
$((\le_L)\ \dashv\ (\le_R))$ *l r*
∧ *partial-equivalence-rel* $(\le_L)$
∧ *partial-equivalence-rel* $(\le_R)$

notation *transport.partial-equivalence-rel-galois-connection* (**infix** $\dashv_{PER}$ *50*)

lemma *partial-equivalence-rel-galois-connectionI* [*intro*]:
assumes $((\le_L)\ \dashv\ (\le_R))$ *l r*
and *partial-equivalence-rel-on* (*in-field* $(\le_L)$) $(\le_L)$
and *partial-equivalence-rel-on* (*in-field* $(\le_R)$) $(\le_R)$
shows $((\le_L)\ \dashv_{PER}\ (\le_R))$ *l r*
⟨*proof*⟩

lemma *partial-equivalence-rel-galois-connectionE* [*elim*]:
assumes $((\le_L)\ \dashv_{PER}\ (\le_R))$ *l r*
obtains $((\le_L)\ \dashv_{pre}\ (\le_R))$ *l r symmetric* $(\le_L)$ *symmetric* $(\le_R)$
⟨*proof*⟩

context
begin

interpretation *t* : *transport S T f g* **for** *S T f g* ⟨*proof*⟩

lemma *rel-inv-partial-equivalence-rel-galois-connection-eq-partial-equivalence-rel-galois-connection-rel-inv*
[*simp*]: $((\le_R)\ \dashv_{PER}\ (\le_L))^{-1} = ((\ge_L)\ \dashv_{PER}\ (\ge_R))$
⟨*proof*⟩

end

corollary *partial-equivalence-rel-galois-connection-rel-inv-iff-partial-equivalence-rel-galois-connection*
[*iff*]: $((\geq_L) \dashv_{PER} (\geq_R))\ l\ r \longleftrightarrow ((\leq_R) \dashv_{PER} (\leq_L))\ r\ l$
⟨*proof*⟩

lemma *left-Galois-comp-ge-Galois-left-eq-left-if-partial-equivalence-rel-galois-connection*:
assumes $((\leq_L) \dashv_{PER} (\leq_R))\ l\ r$
shows $((_{L}\lessapprox) \circ\circ (\gtrapprox_L)) = (\leq_L)$
⟨*proof*⟩

2.1.2 Ordered Equivalences

definition *preorder-equivalence* ≡
$((\leq_L) \equiv_G (\leq_R))\ l\ r$
$\wedge$ *preorder-on* (*in-field* $(\leq_L)$) $(\leq_L)$
$\wedge$ *preorder-on* (*in-field* $(\leq_R)$) $(\leq_R)$

notation *transport.preorder-equivalence* (**infix** $\equiv_{pre}$ *50*)

lemma *preorder-equivalence-if-galois-equivalenceI* [*intro*]:
assumes $((\leq_L) \equiv_G (\leq_R))\ l\ r$
and *preorder-on* (*in-field* $(\leq_L)$) $(\leq_L)$
and *preorder-on* (*in-field* $(\leq_R)$) $(\leq_R)$
shows $((\leq_L) \equiv_{pre} (\leq_R))\ l\ r$
⟨*proof*⟩

lemma *preorder-equivalence-if-order-equivalenceI*:
assumes $((\leq_L) \equiv_o (\leq_R))\ l\ r$
and *transitive* $(\leq_L)$
and *transitive* $(\leq_R)$
shows $((\leq_L) \equiv_{pre} (\leq_R))\ l\ r$
⟨*proof*⟩

lemma *preorder-equivalence-galois-equivalenceE* [*elim*]:
assumes $((\leq_L) \equiv_{pre} (\leq_R))\ l\ r$
obtains $((\leq_L) \equiv_G (\leq_R))\ l\ r$ *preorder-on* (*in-field* $(\leq_L)$) $(\leq_L)$
preorder-on (*in-field* $(\leq_R)$) $(\leq_R)$
⟨*proof*⟩

lemma *preorder-equivalence-order-equivalenceE*:
assumes $((\leq_L) \equiv_{pre} (\leq_R))\ l\ r$
obtains $((\leq_L) \equiv_o (\leq_R))\ l\ r$ *preorder-on* (*in-field* $(\leq_L)$) $(\leq_L)$
preorder-on (*in-field* $(\leq_R)$) $(\leq_R)$
⟨*proof*⟩

context
begin

interpretation *t* : *transport S T f g* **for** *S T f g* ⟨*proof*⟩

lemma *rel-inv-preorder-equivalence-eq-preorder-equivalence* [*simp*]:
$((\le_R) \equiv_{pre} (\le_L))^{-1} = ((\le_L) \equiv_{pre} (\le_R))$
⟨*proof*⟩

end

corollary *preorder-equivalence-right-left-iff-preorder-equivalence-left-right*:
$((\le_R) \equiv_{pre} (\le_L))\ r\ l \longleftrightarrow ((\le_L) \equiv_{pre} (\le_R))\ l\ r$
⟨*proof*⟩

lemma *preorder-equivalence-rel-inv-eq-preorder-equivalence* [*simp*]:
$((\ge_L) \equiv_{pre} (\ge_R)) = ((\le_L) \equiv_{pre} (\le_R))$
⟨*proof*⟩

definition *partial-equivalence-rel-equivalence* ≡
$((\le_L) \equiv_G (\le_R))\ l\ r$
∧ *partial-equivalence-rel* $(\le_L)$
∧ *partial-equivalence-rel* $(\le_R)$

notation *transport.partial-equivalence-rel-equivalence* (**infix** $\equiv_{PER}$ *50*)

lemma *partial-equivalence-rel-equivalence-if-galois-equivalenceI* [*intro*]:
assumes $((\le_L) \equiv_G (\le_R))\ l\ r$
and *partial-equivalence-rel* $(\le_L)$
and *partial-equivalence-rel* $(\le_R)$
shows $((\le_L) \equiv_{PER} (\le_R))\ l\ r$
⟨*proof*⟩

lemma *partial-equivalence-rel-equivalence-if-order-equivalenceI*:
assumes $((\le_L) \equiv_o (\le_R))\ l\ r$
and *partial-equivalence-rel* $(\le_L)$
and *partial-equivalence-rel* $(\le_R)$
shows $((\le_L) \equiv_{PER} (\le_R))\ l\ r$
⟨*proof*⟩

lemma *partial-equivalence-rel-equivalenceE* [*elim*]:
assumes $((\le_L) \equiv_{PER} (\le_R))\ l\ r$
obtains $((\le_L) \equiv_{pre} (\le_R))\ l\ r$ *symmetric* $(\le_L)$ *symmetric* $(\le_R)$
⟨*proof*⟩

context
begin

interpretation t : *transport* $S\ T\ f\ g$ **for** $S\ T\ f\ g$ ⟨*proof*⟩

lemma *rel-inv-partial-equivalence-rel-equivalence-eq-partial-equivalence-rel-equivalence* [*simp*]:
$((\le_R) \equiv_{PER} (\le_L))^{-1} = ((\le_L) \equiv_{PER} (\le_R))$

⟨*proof*⟩

end

corollary *partial-equivalence-rel-equivalence-right-left-iff-partial-equivalence-rel-equivalence-left-right*:
$((\le_R) \equiv_{PER} (\le_L))\ r\ l \longleftrightarrow ((\le_L) \equiv_{PER} (\le_R))\ l\ r$
⟨*proof*⟩

lemma *partial-equivalence-rel-equivalence-rel-inv-eq-partial-equivalence-rel-equivalence*
[*simp*]: $((\ge_L) \equiv_{PER} (\ge_R)) = ((\le_L) \equiv_{PER} (\le_R))$
⟨*proof*⟩

end

end

2.2 Transport using Bijections

theory *Transport-Bijections*
imports
Functions-Bijection
Transport-Base
begin

Summary Setup for Transport using bijective transport functions.

locale *transport-bijection* =
fixes $L :: 'a \Rightarrow 'a \Rightarrow bool$
and $R :: 'b \Rightarrow 'b \Rightarrow bool$
and $l :: 'a \Rightarrow 'b$
and $r :: 'b \Rightarrow 'a$
assumes *mono-wrt-rel-left*: $(L \Rrightarrow_m R)\ l$
and *mono-wrt-rel-right*: $(R \Rrightarrow_m L)\ r$
and *inverse-left-right*: *inverse-on* (*in-field* L) $l\ r$
and *inverse-right-left*: *inverse-on* (*in-field* R) $r\ l$
begin

interpretation *transport* $L\ R\ l\ r$ ⟨*proof*⟩
interpretation *g-flip-inv* : *galois* $(\ge_R)\ (\ge_L)\ r\ l$ ⟨*proof*⟩

lemma *bijection-on-in-field*: *bijection-on* (*in-field* $(\le_L)$) (*in-field* $(\le_R)$) $l\ r$
⟨*proof*⟩

lemma *half-galois-prop-left*: $((\le_L)\ {}_h\trianglelefteq\ (\le_R))\ l\ r$
⟨*proof*⟩

lemma *half-galois-prop-right*: $((\le_L) \trianglelefteq_h (\le_R))\ l\ r$
⟨*proof*⟩

lemma *galois-prop*: $((\leq_L) \trianglelefteq (\leq_R))$ *l r*
⟨*proof*⟩

lemma *galois-connection*: $((\leq_L) \dashv (\leq_R))$ *l r*
⟨*proof*⟩

lemma *rel-equivalence-on-unitI*:
assumes *reflexive-on* (*in-field* $(\leq_L)$) $(\leq_L)$
shows *rel-equivalence-on* (*in-field* $(\leq_L)$) $(\leq_L)$ η
⟨*proof*⟩

interpretation *flip* : *transport-bijection R L r l*
rewrites *order-functors.unit r l* $\equiv \varepsilon$
⟨*proof*⟩

lemma *galois-equivalence*: $((\leq_L) \equiv_G (\leq_R))$ *l r*
⟨*proof*⟩

lemmas *rel-equivalence-on-counitI* = *flip.rel-equivalence-on-unitI*

lemma *order-equivalenceI*:
assumes *reflexive-on* (*in-field* $(\leq_L)$) $(\leq_L)$
and *reflexive-on* (*in-field* $(\leq_R)$) $(\leq_R)$
shows $((\leq_L) \equiv_o (\leq_R))$ *l r*
⟨*proof*⟩

lemma *preorder-equivalenceI*:
assumes *preorder-on* (*in-field* $(\leq_L)$) $(\leq_L)$
and *preorder-on* (*in-field* $(\leq_R)$) $(\leq_R)$
shows $((\leq_L) \equiv_{pre} (\leq_R))$ *l r*
⟨*proof*⟩

lemma *partial-equivalence-rel-equivalenceI*:
assumes *partial-equivalence-rel* $(\leq_L)$
and *partial-equivalence-rel* $(\leq_R)$
shows $((\leq_L) \equiv_{PER} (\leq_R))$ *l r*
⟨*proof*⟩

end

locale *transport-reflexive-on-in-field-bijection* =
fixes *L* :: $'a \Rightarrow 'a \Rightarrow$ *bool*
and *R* :: $'b \Rightarrow 'b \Rightarrow$ *bool*
and *l* :: $'a \Rightarrow 'b$
and *r* :: $'b \Rightarrow 'a$
assumes *reflexive-on-in-field-left*: *reflexive-on* (*in-field L*) *L*
and *reflexive-on-in-field-right*: *reflexive-on* (*in-field R*) *R*
and *transport-bijection*: *transport-bijection L R l r*

begin

sublocale *tbij?* : *transport-bijection L R l r*
  rewrites *reflexive-on* (*in-field L*) *L* ≡ *True*
  and *reflexive-on* (*in-field R*) *R* ≡ *True*
  and $\bigwedge P.$ (*True* $\Longrightarrow$ *P*) ≡ *Trueprop P*
  ⟨*proof*⟩

lemmas *rel-equivalence-on-unit* = *rel-equivalence-on-unitI*
lemmas *rel-equivalence-on-counit* = *rel-equivalence-on-counitI*
lemmas *order-equivalence* = *order-equivalenceI*

end

locale *transport-preorder-on-in-field-bijection* =
  fixes *L* :: *'a* ⇒ *'a* ⇒ *bool*
  and *R* :: *'b* ⇒ *'b* ⇒ *bool*
  and *l* :: *'a* ⇒ *'b*
  and *r* :: *'b* ⇒ *'a*
  assumes *preorder-on-in-field-left*: *preorder-on* (*in-field L*) *L*
  and *preorder-on-in-field-right*: *preorder-on* (*in-field R*) *R*
  and *transport-bijection*: *transport-bijection L R l r*
begin

sublocale *trefl-bij?* : *transport-reflexive-on-in-field-bijection L R l r*
  rewrites *preorder-on* (*in-field L*) *L* ≡ *True*
  and *preorder-on* (*in-field R*) *R* ≡ *True*
  and $\bigwedge P.$ (*True* $\Longrightarrow$ *P*) ≡ *Trueprop P*
  ⟨*proof*⟩

lemmas *preorder-equivalence* = *preorder-equivalenceI*

end

locale *transport-partial-equivalence-rel-bijection* =
  fixes *L* :: *'a* ⇒ *'a* ⇒ *bool*
  and *R* :: *'b* ⇒ *'b* ⇒ *bool*
  and *l* :: *'a* ⇒ *'b*
  and *r* :: *'b* ⇒ *'a*
  assumes *partial-equivalence-rel-left*: *partial-equivalence-rel L*
  and *partial-equivalence-rel-right*: *partial-equivalence-rel R*
  and *transport-bijection*: *transport-bijection L R l r*
begin

sublocale *tpre-bij?* : *transport-preorder-on-in-field-bijection L R l r*
  rewrites *partial-equivalence-rel L* ≡ *True*
  and *partial-equivalence-rel R* ≡ *True*
  and $\bigwedge P.$ (*True* $\Longrightarrow$ *P*) ≡ *Trueprop P*
  ⟨*proof*⟩

lemmas *partial-equivalence-rel-equivalence* = *partial-equivalence-rel-equivalenceI*

end

locale *transport-eq-restrict-bijection* =
  fixes P :: $'a \Rightarrow bool$
  and Q :: $'b \Rightarrow bool$
  and l :: $'a \Rightarrow 'b$
  and r :: $'b \Rightarrow 'a$
  assumes *bijection-on-in-field*:
    bijection-on (*in-field* $((=_P)$ :: $'a \Rightarrow$ -)) (*in-field* $((=_Q)$ :: $'b \Rightarrow$ -)) l r
begin

interpretation *transport* $(=_P)$ $(=_Q)$ l r ⟨*proof*⟩

sublocale *tper-bij?* : *transport-partial-equivalence-rel-bijection* $(=_P)$ $(=_Q)$ l r
  ⟨*proof*⟩

lemma *left-Galois-eq-Galois-eq-eq-restrict*: $(_L\lessapprox) = ($*galois-rel.Galois* $(=)$ $(=)$ $r)\restriction_P\upharpoonright_Q$
  ⟨*proof*⟩

end

locale *transport-eq-bijection* =
  fixes l :: $'a \Rightarrow 'b$
  and r :: $'b \Rightarrow 'a$
  assumes *bijection-on-in-field*:
    bijection-on (*in-field* $((=)$ :: $'a \Rightarrow$ -)) (*in-field* $((=)$ :: $'b \Rightarrow$ -)) l r
begin

sublocale *teq-restr-bij?* : *transport-eq-restrict-bijection* $\top$ $\top$ l r
  rewrites $(=_{\top :: 'a \Rightarrow bool}) = ((=) :: 'a \Rightarrow$ -)
  and $(=_{\top :: 'b \Rightarrow bool}) = ((=) :: 'b \Rightarrow$ -)
  ⟨*proof*⟩

end

end

2.3 Compositions With Agreeing Relations

2.3.1 Basic Setup

theory *Transport-Compositions-Agree-Base*
  imports
    Transport-Base
begin

locale *transport-comp-agree* =
g1 : *galois L1 R1 l1 r1* + *g2* : *galois L2 R2 l2 r2*
for *L1* :: $'a \Rightarrow 'a \Rightarrow$ *bool*
and *R1* :: $'b \Rightarrow 'b \Rightarrow$ *bool*
and *l1* :: $'a \Rightarrow 'b$
and *r1* :: $'b \Rightarrow 'a$
and *L2* :: $'b \Rightarrow 'b \Rightarrow$ *bool*
and *R2* :: $'c \Rightarrow 'c \Rightarrow$ *bool*
and *l2* :: $'b \Rightarrow 'c$
and *r2* :: $'c \Rightarrow 'b$
begin

This locale collects results about the composition of transportable components under the assumption that the relations *R1* and *L2* agree (in one sense or another) whenever required. Such an agreement may not necessarily hold in practice, and the resulting theorems are not particularly pretty. However, in the special case where *R1* = *L2*, most side-conditions disappear and the results are very simple.

notation *L1* (**infix** $\le_{L1}$ *50*)
notation *R1* (**infix** $\le_{R1}$ *50*)
notation *L2* (**infix** $\le_{L2}$ *50*)
notation *R2* (**infix** $\le_{R2}$ *50*)

notation *g1.ge-left* (**infix** $\ge_{L1}$ *50*)
notation *g1.ge-right* (**infix** $\ge_{R1}$ *50*)
notation *g2.ge-left* (**infix** $\ge_{L2}$ *50*)
notation *g2.ge-right* (**infix** $\ge_{R2}$ *50*)

notation *g1.left-Galois* (**infix** ${}_{L1}\lessapprox$ *50*)
notation *g1.right-Galois* (**infix** ${}_{R1}\lessapprox$ *50*)
notation *g2.left-Galois* (**infix** ${}_{L2}\lessapprox$ *50*)
notation *g2.right-Galois* (**infix** ${}_{R2}\lessapprox$ *50*)

notation *g1.ge-Galois-left* (**infix** $\gtrapprox_{L1}$ *50*)
notation *g1.ge-Galois-right* (**infix** $\gtrapprox_{R1}$ *50*)
notation *g2.ge-Galois-left* (**infix** $\gtrapprox_{L2}$ *50*)
notation *g2.ge-Galois-right* (**infix** $\gtrapprox_{R2}$ *50*)

notation *g1.right-ge-Galois* (**infix** ${}_{R1}\gtrapprox$ *50*)
notation *g1.Galois-right* (**infix** $\lessapprox_{R1}$ *50*)
notation *g2.right-ge-Galois* (**infix** ${}_{R2}\gtrapprox$ *50*)
notation *g2.Galois-right* (**infix** $\lessapprox_{R2}$ *50*)

notation *g1.left-ge-Galois* (**infix** ${}_{L1}\gtrapprox$ *50*)
notation *g1.Galois-left* (**infix** $\lessapprox_{L1}$ *50*)
notation *g2.left-ge-Galois* (**infix** ${}_{L2}\gtrapprox$ *50*)
notation *g2.Galois-left* (**infix** $\lessapprox_{L2}$ *50*)

notation *g1.unit* (η_1)
notation *g1.counit* (ε_1)
notation *g2.unit* (η_2)
notation *g2.counit* (ε_2)

abbreviation (*input*) $L \equiv L1$

definition $l \equiv l2 \circ l1$

lemma *left-eq-comp*: $l = l2 \circ l1$
⟨*proof*⟩

lemma *left-eq* [*simp*]: $l\ x = l2\ (l1\ x)$
⟨*proof*⟩

context
begin

interpretation *flip* : *transport-comp-agree R2 L2 r2 l2 R1 L1 r1 l1* ⟨*proof*⟩

abbreviation (*input*) $R \equiv$ *flip.L*
abbreviation $r \equiv$ *flip.l*

lemma *right-eq-comp*: $r = r1 \circ r2$
⟨*proof*⟩

lemma *right-eq* [*simp*]: $r\ z = r1\ (r2\ z)$
⟨*proof*⟩

lemmas *transport-defs* = *left-eq-comp right-eq-comp*

end

sublocale *transport L R l r* ⟨*proof*⟩

notation *L* (**infix** $\leq_L$ *50*)
notation *R* (**infix** $\leq_R$ *50*)

end

locale *transport-comp-same* =
transport-comp-agree L1 R1 l1 r1 R1 R2 l2 r2
for $L1 :: 'a \Rightarrow 'a \Rightarrow bool$
and $R1 :: 'b \Rightarrow 'b \Rightarrow bool$
and $l1 :: 'a \Rightarrow 'b$
and $r1 :: 'b \Rightarrow 'a$
and $R2 :: 'c \Rightarrow 'c \Rightarrow bool$
and $l2 :: 'b \Rightarrow 'c$

and $r2$ :: $'c \Rightarrow 'b$
begin

This locale is a special case of *transport-comp-agree* where the left and right components both use $(\le_{R1})$ as their right and left relation, respectively. This is the special case that is most prominent in the literature. The resulting theorems are quite simple, but often not applicable in practice.

end

end

2.3.2 Monotonicity

theory *Transport-Compositions-Agree-Monotone*
imports
Transport-Compositions-Agree-Base
begin

context *transport-comp-agree*
begin

lemma *mono-wrt-rel-leftI*:
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1\ ((\le_{L2}) \Rrightarrow_m (\le_{R2}))\ l2$
and $\bigwedge x\ y.\ x \le_{L1} y \Longrightarrow l1\ x \le_{R1} l1\ y \Longrightarrow l1\ x \le_{L2} l1\ y$
shows $((\le_L) \Rrightarrow_m (\le_R))\ l$
⟨*proof*⟩

end

context *transport-comp-same*
begin

lemma *mono-wrt-rel-leftI*:
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1\ ((\le_{R1}) \Rrightarrow_m (\le_{R2}))\ l2$
shows $((\le_L) \Rrightarrow_m (\le_R))\ l$
⟨*proof*⟩

end

end

2.3.3 Galois Property

theory *Transport-Compositions-Agree-Galois-Property*
imports
Transport-Compositions-Agree-Base
begin

context *transport-comp-agree*
begin

lemma *galois-propI*:
assumes *galois1*: $((\le_{L1}) \trianglelefteq (\le_{R1}))$ *l1 r1*
and *galois2*: $((\le_{L2}) \trianglelefteq (\le_{R2}))$ *l2 r2*
and *mono-l1*: $([$*in-dom* $(\le_{L1})] \Rrightarrow_m$ *in-dom* $(\le_{L2}))$ *l1*
and *mono-r2*: $([$*in-codom* $(\le_{R2})] \Rrightarrow_m$ *in-codom* $(\le_{R1}))$ *r2*
and *agree*: $([$*in-dom* $(\le_{L1})] \Rrightarrow [$*in-codom* $(\le_{R2})] \Rrightarrow (\longleftrightarrow))$
(*rel-bimap l1 r2* $(\le_{R1})$) (*rel-bimap l1 r2* $(\le_{L2})$)
shows $((\le_L) \trianglelefteq (\le_R))$ *l r*
⟨*proof*⟩

end

context *transport-comp-same*
begin

corollary *galois-propI*:
assumes $((\le_{L1}) \trianglelefteq (\le_{R1}))$ *l1 r1*
and $((\le_{R1}) \trianglelefteq (\le_{R2}))$ *l2 r2*
and $([$*in-dom* $(\le_{L1})] \Rrightarrow_m$ *in-dom* $(\le_{R1}))$ *l1*
and $([$*in-codom* $(\le_{R2})] \Rrightarrow_m$ *in-codom* $(\le_{R1}))$ *r2*
shows $((\le_L) \trianglelefteq (\le_R))$ *l r*
⟨*proof*⟩

end

end

2.3.4 Galois Connection

theory *Transport-Compositions-Agree-Galois-Connection*
imports
Transport-Compositions-Agree-Monotone
Transport-Compositions-Agree-Galois-Property
begin

context *transport-comp-agree*
begin

interpretation *flip* : *transport-comp-agree R2 L2 r2 l2 R1 L1 r1 l1* ⟨*proof*⟩

lemma *galois-connectionI*:
assumes *galois*: $((\le_{L1}) \dashv (\le_{R1}))$ *l1 r1* $((\le_{L2}) \dashv (\le_{R2}))$ *l2 r2*
and *mono-L1-L2-l1*: $\bigwedge x\ y.\ x \le_{L1} y \Longrightarrow l1\ x \le_{R1} l1\ y \Longrightarrow l1\ x \le_{L2} l1\ y$
and *mono-R2-R1-r2*: $\bigwedge x\ y.\ x \le_{R2} y \Longrightarrow r2\ x \le_{L2} r2\ y \Longrightarrow r2\ x \le_{R1} r2\ y$

and ([*in-dom* $(\leq_{L1})$] $\Rrightarrow$ [*in-codom* $(\leq_{R2})$] $\Rrightarrow$ $(\longleftrightarrow)$)
(*rel-bimap l1 r2* $(\leq_{R1})$) (*rel-bimap l1 r2* $(\leq_{L2})$)
shows $((\leq_L) \dashv (\leq_R))$ *l r*
⟨*proof*⟩

lemma *galois-connectionI'*:
assumes $((\leq_{L1}) \dashv (\leq_{R1}))$ *l1 r1* $((\leq_{L2}) \dashv (\leq_{R2}))$ *l2 r2*
and $((\leq_{L1}) \Rrightarrow_m (\leq_{L2}))$ *l1* $((\leq_{R2}) \Rrightarrow_m (\leq_{R1}))$ *r2*
and ([*in-dom* $(\leq_{L1})$] $\Rrightarrow$ [*in-codom* $(\leq_{R2})$] $\Rrightarrow$ $(\longleftrightarrow)$)
(*rel-bimap l1 r2* $(\leq_{R1})$) (*rel-bimap l1 r2* $(\leq_{L2})$)
shows $((\leq_L) \dashv (\leq_R))$ *l r*
⟨*proof*⟩

end

context *transport-comp-same*
begin

corollary *galois-connectionI*:
assumes $((\leq_{L1}) \dashv (\leq_{R1}))$ *l1 r1* $((\leq_{R1}) \dashv (\leq_{R2}))$ *l2 r2*
shows $((\leq_L) \dashv (\leq_R))$ *l r*
⟨*proof*⟩

end

end

2.3.5 Galois Equivalence

theory *Transport-Compositions-Agree-Galois-Equivalence*
imports
Transport-Compositions-Agree-Galois-Connection
begin

context *transport-comp-agree*
begin

interpretation *flip* : *transport-comp-agree R2 L2 r2 l2 R1 L1 r1 l1* ⟨*proof*⟩

lemma *galois-equivalenceI*:
assumes *galois*: $((\leq_{L1}) \equiv_G (\leq_{R1}))$ *l1 r1* $((\leq_{L2}) \equiv_G (\leq_{R2}))$ *l2 r2*
and *mono-L1-L2-l1*: $\bigwedge x\ y.\ x \leq_{L1} y \Longrightarrow l1\ x \leq_{R1} l1\ y \Longrightarrow l1\ x \leq_{L2} l1\ y$
and *mono-R2-R1-r2*: $\bigwedge x\ y.\ x \leq_{R2} y \Longrightarrow r2\ x \leq_{L2} r2\ y \Longrightarrow r2\ x \leq_{R1} r2\ y$
and ([*in-dom* $(\leq_{L1})$] $\Rrightarrow$ [*in-codom* $(\leq_{R2})$] $\Rrightarrow$ $(\longleftrightarrow)$)
(*rel-bimap l1 r2* $(\leq_{R1})$) (*rel-bimap l1 r2* $(\leq_{L2})$)
and *mono-iff2*: ([*in-dom* $(\leq_{R2})$] $\Rrightarrow$ [*in-codom* $(\leq_{L1})$] $\Rrightarrow$ $(\longleftrightarrow)$)
(*rel-bimap r2 l1* $(\leq_{R1})$) (*rel-bimap r2 l1* $(\leq_{L2})$)
shows $((\leq_L) \equiv_G (\leq_R))$ *l r*

⟨*proof*⟩

lemma *galois-equivalenceI'*:
assumes $((\le_{L1}) \equiv_G (\le_{R1}))$ *l1 r1* $((\le_{L2}) \equiv_G (\le_{R2}))$ *l2 r2*
and $((\le_{L1}) \Rrightarrow_m (\le_{L2}))$ *l1* $((\le_{R2}) \Rrightarrow_m (\le_{R1}))$ *r2*
and ([*in-dom* $(\le_{L1})$] $\Rrightarrow$ [*in-codom* $(\le_{R2})$] $\Rrightarrow$ $(\longleftrightarrow)$)
(*rel-bimap l1 r2* $(\le_{R1})$) (*rel-bimap l1 r2* $(\le_{L2})$)
and ([*in-dom* $(\le_{R2})$] $\Rrightarrow$ [*in-codom* $(\le_{L1})$] $\Rrightarrow$ $(\longleftrightarrow)$)
(*rel-bimap r2 l1* $(\le_{R1})$) (*rel-bimap r2 l1* $(\le_{L2})$)
shows $((\le_L) \equiv_G (\le_R))$ *l r*
⟨*proof*⟩

end

context *transport-comp-same*
begin

lemma *galois-equivalenceI*:
assumes $((\le_{L1}) \equiv_G (\le_{R1}))$ *l1 r1* $((\le_{R1}) \equiv_G (\le_{R2}))$ *l2 r2*
shows $((\le_L) \equiv_G (\le_R))$ *l r*
⟨*proof*⟩

end

end

2.3.6 Galois Relator

theory *Transport-Compositions-Agree-Galois-Relator*
imports
Transport-Compositions-Agree-Base
begin

context *transport-comp-agree*
begin

lemma *left-Galois-le-comp-left-GaloisI*:
assumes *in-codom-mono-r2*: ([*in-codom* $(\le_{R2})$] $\Rrightarrow_m$ *in-codom* $(\le_{R1})$) *r2*
and *r2-L2-self-if-in-codom*: $\bigwedge z.$ *in-codom* $(\le_{R2})\ z \Longrightarrow r2\ z \le_{L2} r2\ z$
shows $({}_{L}\lessapprox) \le (({}_{L1}\lessapprox) \circ\circ ({}_{L2}\lessapprox))$
⟨*proof*⟩

lemma *comp-left-Galois-le-left-GaloisI*:
assumes *mono-r1*: $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*
and *trans-L1*: *transitive* $(\le_{L1})$
and *R1-r2-if-in-codom*: $\bigwedge y\ z.$ *in-codom* $(\le_{R2})\ z \Longrightarrow y \le_{L2} r2\ z \Longrightarrow y \le_{R1} r2\ z$
shows $(({}_{L1}\lessapprox) \circ\circ ({}_{L2}\lessapprox)) \le ({}_{L}\lessapprox)$
⟨*proof*⟩

corollary *left-Galois-eq-comp-left-GaloisI*:
assumes $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*
and *transitive* $(\le_{L1})$
and $\bigwedge z$. *in-codom* $(\le_{R2})$ $z \Longrightarrow r2\ z \le_{L2} r2\ z$
and $\bigwedge y\ z$. *in-codom* $(\le_{R2})$ $z \Longrightarrow y \le_{L2} r2\ z \Longrightarrow y \le_{R1} r2\ z$
shows $({}_{L}\lessapprox) = (({}_{L1}\lessapprox) \circ\circ ({}_{L2}\lessapprox))$
⟨*proof*⟩

corollary *left-Galois-eq-comp-left-GaloisI'*:
assumes $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*
and *transitive* $(\le_{L1})$
and $((\le_{R2}) \Rrightarrow_m (\le_{L2}))$ *r2*
and *reflexive-on* (*in-codom* $(\le_{R2})$) $(\le_{R2})$
and $\bigwedge y\ z$. *in-codom* $(\le_{R2})$ $z \Longrightarrow y \le_{L2} r2\ z \Longrightarrow y \le_{R1} r2\ z$
shows $({}_{L}\lessapprox) = (({}_{L1}\lessapprox) \circ\circ ({}_{L2}\lessapprox))$
⟨*proof*⟩

corollary *left-Galois-eq-comp-left-GaloisI''*:
assumes $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*
and *transitive* $(\le_{L1})$
and $((\le_{R2}) \Rrightarrow_m (\le_{L2}))$ *r2*
and *reflexive-on* (*in-codom* $(\le_{L2})$) $(\le_{L2})$
and $\bigwedge y\ z$. *in-codom* $(\le_{R2})$ $z \Longrightarrow y \le_{L2} r2\ z \Longrightarrow y \le_{R1} r2\ z$
shows $({}_{L}\lessapprox) = (({}_{L1}\lessapprox) \circ\circ ({}_{L2}\lessapprox))$
⟨*proof*⟩

end

context *transport-comp-same*
begin

lemma *left-Galois-eq-comp-left-GaloisI*:
assumes $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*
and *transitive* $(\le_{L1})$
and $((\le_{R2}) \Rrightarrow_m (\le_{R1}))$ *r2*
and *reflexive-on* (*in-codom* $(\le_{R2})$) $(\le_{R2})$
shows $({}_{L}\lessapprox) = (({}_{L1}\lessapprox) \circ\circ ({}_{L2}\lessapprox))$
⟨*proof*⟩

lemma *left-Galois-eq-comp-left-GaloisI'*:
assumes $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*
and *transitive* $(\le_{L1})$
and *reflexive-on* (*in-codom* $(\le_{R1})$) $(\le_{R1})$
and $((\le_{R2}) \Rrightarrow_m (\le_{R1}))$ *r2*
shows $({}_{L}\lessapprox) = (({}_{L1}\lessapprox) \circ\circ ({}_{L2}\lessapprox))$
⟨*proof*⟩

end

end

2.3.7 Order Equivalence

theory *Transport-Compositions-Agree-Order-Equivalence*
  imports
    Transport-Compositions-Agree-Monotone
begin

context *transport-comp-agree*
begin

Unit

Inflationary **lemma** *inflationary-on-unitI*:
  assumes *mono-l1*: $([P] \Rrightarrow_m P')$ *l1*
  and *mono-r1*: $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*
  and *inflationary-unit1*: *inflationary-on* P $(\le_{L1})$ η_1
  and *trans-L1*: *transitive* $(\le_{L1})$
  and *inflationary-unit2*: *inflationary-on* P' $(\le_{L2})$ η_2
  and *L2-le-R1*: $\bigwedge x.\ P\ x \Longrightarrow l1\ x \le_{L2} r2\ (l\ x) \Longrightarrow l1\ x \le_{R1} r2\ (l\ x)$
  shows *inflationary-on* P $(\le_L)$ η
⟨*proof*⟩

corollary *inflationary-on-in-field-unitI*:
  assumes $((\le_{L1}) \Rrightarrow_m (\le_{L2}))$ *l1*
  and $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*
  and *inflationary-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$ η_1
  and *transitive* $(\le_{L1})$
  and *inflationary-on* (*in-field* $(\le_{L2})$) $(\le_{L2})$ η_2
  and $\bigwedge x.$ *in-field* $(\le_{L1})\ x \Longrightarrow l1\ x \le_{L2} r2\ (l\ x) \Longrightarrow l1\ x \le_{R1} r2\ (l\ x)$
  shows *inflationary-on* (*in-field* $(\le_L)$) $(\le_L)$ η
  ⟨*proof*⟩

Deflationary **context**
begin

interpretation *inv* :
  transport-comp-agree $(\ge_{L1})$ $(\ge_{R1})$ *l1 r1* $(\ge_{L2})$ $(\ge_{R2})$ *l2 r2*
  rewrites $\bigwedge R\ S.\ (R^{-1} \Rrightarrow_m S^{-1}) \equiv (R \Rrightarrow_m S)$
  and $\bigwedge R.$ *inflationary-on* $P\ R^{-1} \equiv$ *deflationary-on* $P\ R$
  and $\bigwedge R.$ *transitive* $R^{-1} \equiv$ *transitive* R
  and $\bigwedge R.$ *in-field* $R^{-1} \equiv$ *in-field* R
  ⟨*proof*⟩

lemma *deflationary-on-in-field-unitI*:
  assumes $((\le_{L1}) \Rrightarrow_m (\le_{L2}))$ *l1*
  and $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*

and *deflationary-on* (*in-field* $(\leq_{L1})$) $(\leq_{L1})$ η_1
and *transitive* $(\leq_{L1})$
and *deflationary-on* (*in-field* $(\leq_{L2})$) $(\leq_{L2})$ η_2
and $\bigwedge x.$ *in-field* $(\leq_{L1})$ $x \Longrightarrow$ *r2* (*l x*) $\leq_{L2}$ *l1 x* $\Longrightarrow$ *r2* (*l x*) $\leq_{R1}$ *l1 x*
shows *deflationary-on* (*in-field* $(\leq_L)$) $(\leq_L)$ η
⟨*proof*⟩

end

Relational Equivalence

corollary *rel-equivalence-on-in-field-unitI*:
assumes $((\leq_{L1}) \Rrightarrow_m (\leq_{L2}))$ *l1*
and $((\leq_{R1}) \Rrightarrow_m (\leq_{L1}))$ *r1*
and *rel-equivalence-on* (*in-field* $(\leq_{L1})$) $(\leq_{L1})$ η_1
and *transitive* $(\leq_{L1})$
and *rel-equivalence-on* (*in-field* $(\leq_{L2})$) $(\leq_{L2})$ η_2
and $\bigwedge x.$ *in-field* $(\leq_{L1})$ $x \Longrightarrow$ *l1 x* $\leq_{L2}$ *r2* (*l x*) $\Longrightarrow$ *l1 x* $\leq_{R1}$ *r2* (*l x*)
and $\bigwedge x.$ *in-field* $(\leq_{L1})$ $x \Longrightarrow$ *r2* (*l x*) $\leq_{L2}$ *l1 x* $\Longrightarrow$ *r2* (*l x*) $\leq_{R1}$ *l1 x*
shows *rel-equivalence-on* (*in-field* $(\leq_L)$) $(\leq_L)$ η
⟨*proof*⟩

Counit

Corresponding lemmas for the counit can be obtained by flipping the interpretation of the locale.

Order Equivalence

interpretation *flip* : *transport-comp-agree R2 L2 r2 l2 R1 L1 r1 l1*
rewrites *flip.g1.unit* $\equiv \varepsilon_2$ **and** *flip.g2.unit* $\equiv \varepsilon_1$ **and** *flip.unit* $\equiv \varepsilon$
⟨*proof*⟩

lemma *order-equivalenceI*:
assumes $((\leq_{L1}) \equiv_o (\leq_{R1}))$ *l1 r1*
and *transitive* $(\leq_{L1})$
and $((\leq_{L2}) \equiv_o (\leq_{R2}))$ *l2 r2*
and *transitive* $(\leq_{R2})$
and $\bigwedge x\ y.\ x \leq_{L1} y \Longrightarrow$ *l1 x* $\leq_{R1}$ *l1 y* $\Longrightarrow$ *l1 x* $\leq_{L2}$ *l1 y*
and $\bigwedge x\ y.\ x \leq_{R2} y \Longrightarrow$ *r2 x* $\leq_{L2}$ *r2 y* $\Longrightarrow$ *r2 x* $\leq_{R1}$ *r2 y*
and $\bigwedge x.$ *in-field* $(\leq_{L1})$ $x \Longrightarrow$ *l1 x* $\leq_{L2}$ *r2* (*l x*) $\Longrightarrow$ *l1 x* $\leq_{R1}$ *r2* (*l x*)
and $\bigwedge x.$ *in-field* $(\leq_{L1})$ $x \Longrightarrow$ *r2* (*l x*) $\leq_{L2}$ *l1 x* $\Longrightarrow$ *r2* (*l x*) $\leq_{R1}$ *l1 x*
and $\bigwedge x.$ *in-field* $(\leq_{R2})$ $x \Longrightarrow$ *r2 x* $\leq_{R1}$ *l1* (*r x*) $\Longrightarrow$ *r2 x* $\leq_{L2}$ *l1* (*r x*)
and $\bigwedge x.$ *in-field* $(\leq_{R2})$ $x \Longrightarrow$ *l1* (*r x*) $\leq_{R1}$ *r2 x* $\Longrightarrow$ *l1* (*r x*) $\leq_{L2}$ *r2 x*
shows $((\leq_L) \equiv_o (\leq_R))$ *l r*
⟨*proof*⟩

end

context *transport-comp-same*

begin

lemma *order-equivalenceI*:
  assumes $((\leq_{L1}) \equiv_o (\leq_{R1}))$ *l1 r1*
  and *transitive* $(\leq_{L1})$
  and $((\leq_{R1}) \equiv_o (\leq_{R2}))$ *l2 r2*
  and *transitive* $(\leq_{R2})$
  shows $((\leq_{L}) \equiv_o (\leq_{R}))$ *l r*
  ⟨*proof*⟩

end

end

theory *Transport-Compositions-Agree*
  imports
    Transport-Compositions-Agree-Galois-Equivalence
    Transport-Compositions-Agree-Galois-Relator
    Transport-Compositions-Agree-Order-Equivalence
begin

Summary The general - though probably not very useful - results for the composition of transportable components under the condition of agreeing middle relations can be found in *transport-comp-agree*. The special case of a coinciding middle relation can be found in *transport-comp-same*. The latter corresponds to the well-know result in the literature, generalised to partial Galois connections and equivalences.

end

2.4 Generic Compositions

2.4.1 Basic Setup

theory *Transport-Compositions-Generic-Base*
  imports
    Transport-Base
begin

locale *transport-comp* =
  t1 : *transport L1 R1 l1 r1* + *t2* : *transport L2 R2 l2 r2*
  for *L1* :: $'a \Rightarrow 'a \Rightarrow$ *bool*
  and *R1* :: $'b \Rightarrow 'b \Rightarrow$ *bool*
  and *l1* :: $'a \Rightarrow 'b$
  and *r1* :: $'b \Rightarrow 'a$
  and *L2* :: $'b \Rightarrow 'b \Rightarrow$ *bool*
  and *R2* :: $'c \Rightarrow 'c \Rightarrow$ *bool*
  and *l2* :: $'b \Rightarrow 'c$

and $r2$ :: $'c \Rightarrow 'b$
begin

This locale collects results about the composition of transportable components under some generic compatibility conditions on $R1$ and $L2$ (cf. below). The composition is rather subtle, but in return can cover quite general cases.

Explanations and intuition about the construction can be found in [2].

notation $L1$ (**infix** $\le_{L1}$ *50*)
notation $R1$ (**infix** $\le_{R1}$ *50*)
notation $L2$ (**infix** $\le_{L2}$ *50*)
notation $R2$ (**infix** $\le_{R2}$ *50*)

notation *t1.ge-left* (**infix** $\ge_{L1}$ *50*)
notation *t1.ge-right* (**infix** $\ge_{R1}$ *50*)
notation *t2.ge-left* (**infix** $\ge_{L2}$ *50*)
notation *t2.ge-right* (**infix** $\ge_{R2}$ *50*)

notation *t1.left-Galois* (**infix** ${}_{L1}\lessapprox$ *50*)
notation *t1.right-Galois* (**infix** ${}_{R1}\lessapprox$ *50*)
notation *t2.left-Galois* (**infix** ${}_{L2}\lessapprox$ *50*)
notation *t2.right-Galois* (**infix** ${}_{R2}\lessapprox$ *50*)

notation *t1.ge-Galois-left* (**infix** $\gtrapprox_{L1}$ *50*)
notation *t1.ge-Galois-right* (**infix** $\gtrapprox_{R1}$ *50*)
notation *t2.ge-Galois-left* (**infix** $\gtrapprox_{L2}$ *50*)
notation *t2.ge-Galois-right* (**infix** $\gtrapprox_{R2}$ *50*)

notation *t1.right-ge-Galois* (**infix** ${}_{R1}\gtrapprox$ *50*)
notation *t1.Galois-right* (**infix** $\lessapprox_{R1}$ *50*)
notation *t2.right-ge-Galois* (**infix** ${}_{R2}\gtrapprox$ *50*)
notation *t2.Galois-right* (**infix** $\lessapprox_{R2}$ *50*)

notation *t1.left-ge-Galois* (**infix** ${}_{L1}\gtrapprox$ *50*)
notation *t1.Galois-left* (**infix** $\lessapprox_{L1}$ *50*)
notation *t2.left-ge-Galois* (**infix** ${}_{L2}\gtrapprox$ *50*)
notation *t2.Galois-left* (**infix** $\lessapprox_{L2}$ *50*)

notation *t1.unit* (η_1)
notation *t1.counit* (ε_1)
notation *t2.unit* (η_2)
notation *t2.counit* (ε_2)

definition $L \equiv ({}_{L1}\lessapprox) \circ\circ (\le_{L2}) \circ\circ ({}_{R1}\lessapprox)$

lemma *left-rel-eq-comp*: $L = ({}_{L1}\lessapprox) \circ\circ (\le_{L2}) \circ\circ ({}_{R1}\lessapprox)$
⟨*proof*⟩

definition $l \equiv l2 \circ l1$

lemma *left-eq-comp*: $l = l2 \circ l1$
⟨*proof*⟩

lemma *left-eq* [*simp*]: $l\ x = l2\ (l1\ x)$
⟨*proof*⟩

context
begin

interpretation *flip* : *transport-comp R2 L2 r2 l2 R1 L1 r1 l1* ⟨*proof*⟩

abbreviation $R \equiv$ *flip.L*
abbreviation $r \equiv$ *flip.l*

lemma *right-rel-eq-comp*: $R = ({}_{R2}\lessapprox) \circ\circ (\le_{R1}) \circ\circ ({}_{L2}\lessapprox)$
⟨*proof*⟩

lemma *right-eq-comp*: $r = r1 \circ r2$
⟨*proof*⟩

lemma *right-eq* [*simp*]: $r\ z = r1\ (r2\ z)$
⟨*proof*⟩

lemmas *transport-defs* = *left-rel-eq-comp left-eq-comp right-rel-eq-comp right-eq-comp*

end

sublocale *transport L R l r* ⟨*proof*⟩

notation *L* (**infix** $\le_L$ *50*)
notation *R* (**infix** $\le_R$ *50*)

lemma *left-relI* [*intro*]:
assumes $x\ {}_{L1}\lessapprox\ y$
and $y \le_{L2} y'$
and $y'\ {}_{R1}\lessapprox\ x'$
shows $x \le_L x'$
⟨*proof*⟩

lemma *left-relE* [*elim*]:
assumes $x \le_L x'$
obtains $y\ y'$ **where** $x\ {}_{L1}\lessapprox\ y$ $y \le_{L2} y'$ $y'\ {}_{R1}\lessapprox\ x'$
⟨*proof*⟩

context
begin

interpretation *flip* : *transport-comp R2 L2 r2 l2 R1 L1 r1 l1* ⟨*proof*⟩
interpretation *inv* : *transport-comp* $(\geq_{L1})$ $(\geq_{R1})$ *l1 r1* $(\geq_{L2})$ $(\geq_{R2})$ *l2 r2* ⟨*proof*⟩

lemma *ge-left-rel-eq-left-rel-inv-if-galois-prop* [*simp*]:
  assumes $((\leq_{L1}) \trianglelefteq (\leq_{R1}))$ *l1 r1* $((\leq_{R1}) \trianglelefteq (\leq_{L1}))$ *r1 l1*
  shows $(\geq_L) =$ *transport-comp.L* $(\geq_{L1})$ $(\geq_{R1})$ *l1 r1* $(\geq_{L2})$
  ⟨*proof*⟩

corollary *left-rel-inv-iff-left-rel-if-galois-prop* [*iff*]:
  assumes $((\leq_{L1}) \trianglelefteq (\leq_{R1}))$ *l1 r1* $((\leq_{R1}) \trianglelefteq (\leq_{L1}))$ *r1 l1*
  shows (*transport-comp.L* $(\geq_{L1})$ $(\geq_{R1})$ *l1 r1* $(\geq_{L2})$) $x\ x' \longleftrightarrow x' \leq_L x$
  ⟨*proof*⟩

Simplification of Relations

lemma *left-rel-le-left-rel1I*:
  assumes $((\leq_{L1}) \trianglelefteq_h (\leq_{R1}))$ *l1 r1*
  and $((\leq_{R1})\ {}_h\trianglelefteq (\leq_{L1}))$ *r1 l1*
  and *trans-L1*: *transitive* $(\leq_{L1})$
  and *mono-l1*: $((\leq_L) \Rrightarrow_m ((\leq_{R1}) \circ\circ (\leq_{R1})))$ *l1*
  shows $(\leq_L) \leq (\leq_{L1})$
⟨*proof*⟩

lemma *left-rel1-le-left-relI*:
  assumes $((\leq_{L1}) \trianglelefteq_h (\leq_{R1}))$ *l1 r1*
  and *mono-l1*: $((\leq_{L1}) \Rrightarrow_m ((\leq_{R1}) \circ\circ (\leq_{L2}) \circ\circ (\leq_{R1})))$ *l1*
  shows $(\leq_{L1}) \leq (\leq_L)$
⟨*proof*⟩

corollary *left-rel-eq-left-rel1I*:
  assumes $((\leq_{L1}) \trianglelefteq_h (\leq_{R1}))$ *l1 r1*
  and $((\leq_{R1})\ {}_h\trianglelefteq (\leq_{L1}))$ *r1 l1*
  and *transitive* $(\leq_{L1})$
  and $((\leq_L) \Rrightarrow_m ((\leq_{R1}) \circ\circ (\leq_{R1})))$ *l1*
  and $((\leq_{L1}) \Rrightarrow_m ((\leq_{R1}) \circ\circ (\leq_{L2}) \circ\circ (\leq_{R1})))$ *l1*
  shows $(\leq_L) = (\leq_{L1})$
  ⟨*proof*⟩

Note that we may not necessarily have *flip.R* $= (\leq_{L1})$, even in case of equivalence relations. Depending on the use case, one thus may wish to use an alternative composition operation.

lemma *ex-order-equiv-left-rel-neq-left-rel1*:
  ∃ (*L1* :: *bool* ⇒ -) (*R1* :: *bool* ⇒ -) *l1 r1*
    (*L2* :: *bool* ⇒ -) (*R2* :: *bool* ⇒ -) *l2 r2*.
    $(L1 \equiv_o R1)$ *l1 r1*
    ∧ *equivalence-rel L1* ∧ *equivalence-rel R1*
    ∧ $(L2 \equiv_o R2)$ *l2 r2*
    ∧ *equivalence-rel L2* ∧ *equivalence-rel R2*
    ∧ *transport-comp.L L1 R1 l1 r1 L2* ≠ *L1*

⟨*proof*⟩

end

Generic Left to Right Introduction Rules

The following lemmas generalise the proof outline used, for example, when proving monotonicity and the Galois property (cf. the paper [2]).

interpretation *flip* : *transport-comp R2 L2 r2 l2 R1 L1 r1 l1* ⟨*proof*⟩

lemma *right-rel-if-left-relI*:
  assumes $x \le_L x'$
  and *l1-R1-if-L1-r1*: $\bigwedge y.\ \textit{in-codom}\ (\le_{R1})\ y \Longrightarrow x \le_{L1} r1\ y \Longrightarrow l1\ x \le_{R1} y$
  and *t-R1-if-l1-R1*: $\bigwedge y.\ l1\ x \le_{R1} y \Longrightarrow t\ y \le_{R1} y$
  and *R2-l2-if-t-L2-if-l1-R1*:
    $\bigwedge y\ y'.\ l1\ x \le_{R1} y \Longrightarrow t\ y \le_{L2} y' \Longrightarrow z \le_{R2} l2\ y'$
  and *R1-b-if-R1-l1-if-R1-l1*:
    $\bigwedge y\ y'.\ y \le_{R1} l1\ x' \Longrightarrow y' \le_{R1} l1\ x' \Longrightarrow y' \le_{R1} b\ y$
  and *b-L2-r2-if-in-codom-L2-b-if-R1-l1*:
    $\bigwedge y.\ y \le_{R1} l1\ x' \Longrightarrow \textit{in-codom}\ (\le_{L2})\ (b\ y) \Longrightarrow b\ y \le_{L2} r2\ z'$
  and *in-codom-R2-if-in-codom-L2-b-if-R1-l1*:
    $\bigwedge y.\ y \le_{R1} l1\ x' \Longrightarrow \textit{in-codom}\ (\le_{L2})\ (b\ y) \Longrightarrow \textit{in-codom}\ (\le_{R2})\ z'$
  and *rel-comp-le*: $(\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1}) \le (\le_{L2}) \circ\circ (\le_{R1})$
  and *in-codom-rel-comp-le*: $\textit{in-codom}\ ((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le \textit{in-codom}\ (\le_{L2})$
  shows $z \le_R z'$
⟨*proof*⟩

lemma *right-rel-if-left-relI'*:
  assumes $x \le_L x'$
  and *l1-R1-if-L1-r1*: $\bigwedge y.\ \textit{in-codom}\ (\le_{R1})\ y \Longrightarrow x \le_{L1} r1\ y \Longrightarrow l1\ x \le_{R1} y$
  and *R1-b-if-R1-l1*: $\bigwedge y.\ y \le_{R1} l1\ x' \Longrightarrow y \le_{R1} b\ y$
  and *L2-r2-if-L2-b-if-R1-l1*:
    $\bigwedge y\ y'.\ y \le_{R1} l1\ x' \Longrightarrow y' \le_{L2} b\ y \Longrightarrow y' \le_{L2} r2\ z'$
  and *in-codom-R2-if-L2-b-if-R1-l1*:
    $\bigwedge y\ y'.\ y \le_{R1} l1\ x' \Longrightarrow y' \le_{L2} b\ y \Longrightarrow \textit{in-codom}\ (\le_{R2})\ z'$
  and *t-R1-if-R1-l1-if-l1-R1*:
    $\bigwedge y\ y'\ y''.\ l1\ x \le_{R1} y \Longrightarrow l1\ x \le_{R1} y' \Longrightarrow t\ y \le_{R1} y'$
  and *R2-l2-t-if-in-dom-L2-t-if-l1-R1*:
    $\bigwedge y\ y'.\ l1\ x \le_{R1} y \Longrightarrow \textit{in-dom}\ (\le_{L2})\ (t\ y) \Longrightarrow z \le_{R2} l2\ (t\ y)$
  and *in-codom-L2-t-if-in-dom-L2-t-if-l1-R1*:
    $\bigwedge y\ y'.\ l1\ x \le_{R1} y \Longrightarrow \textit{in-dom}\ (\le_{L2})\ (t\ y) \Longrightarrow \textit{in-codom}\ (\le_{L2})\ (t\ y)$
  and *rel-comp-le*: $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le ((\le_{R1}) \circ\circ (\le_{L2}))$
  and *in-dom-rel-comp-le*: $\textit{in-dom}\ ((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le \textit{in-dom}\ (\le_{L2})$
  shows $z \le_R z'$
⟨*proof*⟩

Simplification of Monotonicity Assumptions

Some sufficient conditions for monotonicity assumptions that repeatedly arise in various places.

lemma *mono-in-dom-left-rel-left1-if-in-dom-rel-comp-le*:
  assumes $((\le_{L1})\ {}_h\trianglelefteq\ (\le_{R1}))$ *l1 r1*
  and *in-dom* $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le$ *in-dom* $(\le_{L2})$
  shows $([$*in-dom* $(\le_L)] \Rrightarrow_m$ *in-dom* $(\le_{L2}))$ *l1*
  ⟨*proof*⟩

lemma *mono-in-codom-left-rel-left1-if-in-codom-rel-comp-le*:
  assumes $((\le_{L1})\ {}_h\trianglelefteq\ (\le_{R1}))$ *l1 r1*
  and *in-codom* $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le$ *in-codom* $(\le_{L2})$
  shows $([$*in-codom* $(\le_L)] \Rrightarrow_m$ *in-codom* $(\le_{L2}))$ *l1*
  ⟨*proof*⟩

Simplification of Compatibility Conditions

Most results will depend on certain compatibility conditions between $(\le_{R1})$ and $(\le_{L2})$. We next derive some sufficient assumptions for these conditions.

end

lemma *rel-comp-comp-le-rel-comp-if-rel-comp-comp-if-in-dom-leI*:
  assumes *trans-R*: *transitive R*
  and *refl-S*: *reflexive-on P S*
  and *in-dom-le*: *in-dom* $(R \circ\circ S \circ\circ R) \le P$
  and *rel-comp-le*: $(S \circ\circ R \circ\circ S) \le (S \circ\circ R)$
  shows $(R \circ\circ S \circ\circ R) \le (S \circ\circ R)$
⟨*proof*⟩

lemma *rel-comp-comp-le-rel-comp-if-rel-comp-comp-if-in-codom-leI*:
  assumes *trans-R*: *transitive R*
  and *refl-S*: *reflexive-on P S*
  and *in-codom-le*: *in-codom* $(R \circ\circ S \circ\circ R) \le P$
  and *rel-comp-le*: $(S \circ\circ R \circ\circ S) \le (R \circ\circ S)$
  shows $(R \circ\circ S \circ\circ R) \le (R \circ\circ S)$
⟨*proof*⟩

lemma *rel-comp-comp-le-rel-comp-if-rel-comp-le-if-transitive*:
  assumes *trans-R*: *transitive R*
  and *R-S-le*: $(R \circ\circ S) \le (S \circ\circ R)$
  shows $(R \circ\circ S \circ\circ R) \le (S \circ\circ R)$
⟨*proof*⟩

lemma *rel-comp-comp-le-rel-comp-if-rel-comp-le-if-transitive'*:
  assumes *trans-R*: *transitive R*
  and *S-R-le*: $(S \circ\circ R) \le (R \circ\circ S)$
  shows $(R \circ\circ S \circ\circ R) \le (R \circ\circ S)$
⟨*proof*⟩

lemma *rel-comp-eq-rel-comp-if-le-if-transitive-if-reflexive*:
  assumes *refl-R*: *reflexive-on* (*in-field* S) R
  and *trans-S*: *transitive* S
  and *R-le*: $R \le S \sqcup (=)$
  shows $(R \circ\circ S) = (S \circ\circ R)$
⟨*proof*⟩

lemma *rel-comp-eq-rel-comp-if-in-field-le-if-le-eq*:
  assumes *le-eq*: $R \le (=)$
  and *in-field-le*: *in-field* $S \le$ *in-field* R
  shows $(R \circ\circ S) = (S \circ\circ R)$
⟨*proof*⟩

context *transport-comp*
begin

lemma *left2-right1-left2-le-left2-right1-if-right1-left2-right1-le-left2-right1*:
  assumes *reflexive-on* (*in-codom* $(\le_{R1})$) $(\le_{R1})$
  and *transitive* $(\le_{L2})$
  and $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le ((\le_{L2}) \circ\circ (\le_{R1}))$
  and *in-codom* $((\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2})) \le$ *in-codom* $(\le_{R1})$
  shows $((\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2})) \le ((\le_{L2}) \circ\circ (\le_{R1}))$
  ⟨*proof*⟩

lemma *left2-right1-left2-le-right1-left2-if-right1-left2-right1-le-right1-left2I*:
  assumes *reflexive-on* (*in-dom* $(\le_{R1})$) $(\le_{R1})$
  and *transitive* $(\le_{L2})$
  and $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le ((\le_{R1}) \circ\circ (\le_{L2}))$
  and *in-dom* $((\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2})) \le$ *in-dom* $(\le_{R1})$
  shows $((\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2})) \le ((\le_{R1}) \circ\circ (\le_{L2}))$
  ⟨*proof*⟩

lemma *in-dom-right1-left2-right1-le-if-right1-left2-right1-le*:
  assumes $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le ((\le_{L2}) \circ\circ (\le_{R1}))$
  shows *in-dom* $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le$ *in-dom* $(\le_{L2})$
  ⟨*proof*⟩

lemma *in-codom-right1-left2-right1-le-if-right1-left2-right1-le*:
  assumes $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le ((\le_{R1}) \circ\circ (\le_{L2}))$
  shows *in-codom* $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le$ *in-codom* $(\le_{L2})$
  ⟨*proof*⟩

Our main results will be derivable for two different sets of compatibility conditions. The next two lemmas show the equivalence between those two sets under certain assumptions. In cases where these assumptions are met, we will only state the result for one of the two compatibility conditions. The other one will then be derivable using one of the following lemmas.

definition *middle-compatible-dom* ≡

$(\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1}) \le (\le_{R1}) \circ\circ (\le_{L2})$
$\wedge$ *in-dom* $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le$ *in-dom* $(\le_{L2})$
$\wedge ((\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2})) \le ((\le_{L2}) \circ\circ (\le_{R1}))$
$\wedge$ *in-dom* $((\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2})) \le$ *in-dom* $(\le_{R1})$

lemma *middle-compatible-domI* [*intro*]:
assumes $(\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1}) \le (\le_{R1}) \circ\circ (\le_{L2})$
and *in-dom* $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le$ *in-dom* $(\le_{L2})$
and $((\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2})) \le ((\le_{L2}) \circ\circ (\le_{R1}))$
and *in-dom* $((\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2})) \le$ *in-dom* $(\le_{R1})$
shows *middle-compatible-dom*
⟨*proof*⟩

lemma *middle-compatible-domE* [*elim*]:
assumes *middle-compatible-dom*
obtains $(\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1}) \le (\le_{R1}) \circ\circ (\le_{L2})$
and *in-dom* $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le$ *in-dom* $(\le_{L2})$
and $((\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2})) \le ((\le_{L2}) \circ\circ (\le_{R1}))$
and *in-dom* $((\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2})) \le$ *in-dom* $(\le_{R1})$
⟨*proof*⟩

definition *middle-compatible-codom* ≡
$((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le ((\le_{L2}) \circ\circ (\le_{R1}))$
$\wedge$ *in-codom* $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le$ *in-codom* $(\le_{L2})$
$\wedge (\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2}) \le (\le_{R1}) \circ\circ (\le_{L2})$
$\wedge$ *in-codom* $((\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2})) \le$ *in-codom* $(\le_{R1})$

lemma *middle-compatible-codomI* [*intro*]:
assumes $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le ((\le_{L2}) \circ\circ (\le_{R1}))$
and *in-codom* $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le$ *in-codom* $(\le_{L2})$
and $(\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2}) \le (\le_{R1}) \circ\circ (\le_{L2})$
and *in-codom* $((\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2})) \le$ *in-codom* $(\le_{R1})$
shows *middle-compatible-codom*
⟨*proof*⟩

lemma *middle-compatible-codomE* [*elim*]:
assumes *middle-compatible-codom*
obtains $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le ((\le_{L2}) \circ\circ (\le_{R1}))$
and *in-codom* $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le$ *in-codom* $(\le_{L2})$
and $(\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2}) \le (\le_{R1}) \circ\circ (\le_{L2})$
and *in-codom* $((\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2})) \le$ *in-codom* $(\le_{R1})$
⟨*proof*⟩

context
begin

interpretation *flip* : *transport-comp R2 L2 r2 l2 R1 L1 r1 l1* ⟨*proof*⟩

lemma *rel-comp-comp-le-assms-if-in-codom-rel-comp-comp-leI*:

assumes *preorder-on* (*in-field* ($\le_{R1}$)) ($\le_{R1}$)
and *preorder-on* (*in-field* ($\le_{L2}$)) ($\le_{L2}$)
and *middle-compatible-codom*
shows *middle-compatible-dom*
⟨*proof*⟩

lemma *rel-comp-comp-le-assms-if-in-dom-rel-comp-comp-leI*:
assumes *preorder-on* (*in-field* ($\le_{R1}$)) ($\le_{R1}$)
and *preorder-on* (*in-field* ($\le_{L2}$)) ($\le_{L2}$)
and *middle-compatible-dom*
shows *middle-compatible-codom*
⟨*proof*⟩

lemma *middle-compatible-dom-iff-middle-compatible-codom-if-preorder-on*:
assumes *preorder-on* (*in-field* ($\le_{R1}$)) ($\le_{R1}$)
and *preorder-on* (*in-field* ($\le_{L2}$)) ($\le_{L2}$)
shows *middle-compatible-dom* $\longleftrightarrow$ *middle-compatible-codom*
⟨*proof*⟩

end

Finally we derive some sufficient assumptions for the compatibility conditions.

lemma *right1-left2-right1-le-assms-if-right1-left2-eqI*:
assumes *transitive* ($\le_{R1}$)
and $((\le_{R1}) \circ\circ (\le_{L2})) = ((\le_{L2}) \circ\circ (\le_{R1}))$
shows $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le ((\le_{L2}) \circ\circ (\le_{R1}))$
and $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le ((\le_{R1}) \circ\circ (\le_{L2}))$
⟨*proof*⟩

interpretation *flip* : *transport-comp R2 L2 r2 l2 R1 L1 r1 l1*
rewrites $((\le_{L2}) \circ\circ (\le_{R1})) = ((\le_{R1}) \circ\circ (\le_{L2})) \equiv ((\le_{R1}) \circ\circ (\le_{L2})) = ((\le_{L2}) \circ\circ (\le_{R1}))$
⟨*proof*⟩

lemma *middle-compatible-codom-if-rel-comp-eq-if-transitive*:
assumes *transitive* ($\le_{R1}$) *transitive* ($\le_{L2}$)
and $((\le_{R1}) \circ\circ (\le_{L2})) = ((\le_{L2}) \circ\circ (\le_{R1}))$
shows *middle-compatible-codom*
⟨*proof*⟩

lemma *middle-compatible-codom-if-right1-le-left2-eqI*:
assumes *preorder-on* (*in-field* ($\le_{R1}$)) ($\le_{R1}$) *transitive* ($\le_{L2}$)
and $(\le_{R1}) \le (\le_{L2}) \sqcup (=)$
and *in-field* ($\le_{L2}$) $\le$ *in-field* ($\le_{R1}$)
shows *middle-compatible-codom*
⟨*proof*⟩

lemma *middle-compatible-codom-if-right1-le-eqI*:

assumes $(\leq_{R1}) \leq (=)$
and *transitive* $(\leq_{L2})$
and *in-field* $(\leq_{L2}) \leq$ *in-field* $(\leq_{R1})$
shows *middle-compatible-codom*
⟨*proof*⟩

end

end

2.4.2 Galois Property

theory *Transport-Compositions-Generic-Galois-Property*
imports
Transport-Compositions-Generic-Base
begin

context *transport-comp*
begin

interpretation *flip* : *transport-comp R2 L2 r2 l2 R1 L1 r1 l1*
rewrites *flip.t2.unit* $= \varepsilon_1$ **and** *flip.t1.counit* $\equiv \eta_2$
⟨*proof*⟩

lemma *half-galois-prop-left-left-rightI*:
assumes $((\leq_{L1})\ {}_h\trianglelefteq\ (\leq_{R1}))$ *l1 r1*
and *deflationary-counit1*: *deflationary-on* (*in-codom* $(\leq_{R1})$) $(\leq_{R1})\ \varepsilon_1$
and *trans-R1*: *transitive* $(\leq_{R1})$
and $((\leq_{L2}) \Rrightarrow_m (\leq_{R2}))$ *l2*
and *reflexive-on* (*in-codom* $(\leq_{L2})$) $(\leq_{L2})$
and $((\leq_{R1}) \circ\circ (\leq_{L2}) \circ\circ (\leq_{R1})) \leq ((\leq_{L2}) \circ\circ (\leq_{R1}))$
and *in-codom* $((\leq_{R1}) \circ\circ (\leq_{L2}) \circ\circ (\leq_{R1})) \leq$ *in-codom* $(\leq_{L2})$
and *mono-in-codom-r2*: ([*in-codom* $(\leq_R)$] $\Rrightarrow_m$ *in-codom* $(\leq_{R1})$) *r2*
shows $((\leq_L)\ {}_h\trianglelefteq\ (\leq_R))$ *l r*
⟨*proof*⟩

lemma *half-galois-prop-left-left-rightI'*:
assumes $((\leq_{L1})\ {}_h\trianglelefteq\ (\leq_{R1}))$ *l1 r1*
and *deflationary-counit1*: *deflationary-on* (*in-codom* $(\leq_{R1})$) $(\leq_{R1})\ \varepsilon_1$
and *trans-R1*: *transitive* $(\leq_{R1})$
and $((\leq_{L2}) \Rrightarrow_m (\leq_{R2}))$ *l2*
and *refl-L2*: *reflexive-on* (*in-dom* $(\leq_{L2})$) $(\leq_{L2})$
and $((\leq_{R1}) \circ\circ (\leq_{L2}) \circ\circ (\leq_{R1})) \leq ((\leq_{R1}) \circ\circ (\leq_{L2}))$
and *in-dom* $((\leq_{R1}) \circ\circ (\leq_{L2}) \circ\circ (\leq_{R1})) \leq$ *in-dom* $(\leq_{L2})$
and *mono-in-codom-r2*: ([*in-codom* $(\leq_R)$] $\Rrightarrow_m$ *in-codom* $(\leq_{R1})$) *r2*
shows $((\leq_L)\ {}_h\trianglelefteq\ (\leq_R))$ *l r*
⟨*proof*⟩

lemma *half-galois-prop-right-left-rightI*:
assumes $((\leq_{R1}) \Rrightarrow_m (\leq_{L1}))$ *r1*
and $((\leq_{L1}) \trianglelefteq_h (\leq_{R1}))$ *l1 r1*
and *inflationary-counit1*: *inflationary-on* (*in-codom* $(\leq_{R1})$) $(\leq_{R1})$ ε_1
and $((\leq_{R2})\ {}_h\trianglelefteq (\leq_{L2}))$ *r2 l2*
and *inflationary-unit2*: *inflationary-on* (*in-dom* $(\leq_{L2})$) $(\leq_{L2})$ η_2
and *trans-L2*: *transitive* $(\leq_{L2})$
and *mono-in-dom-l1*: ([*in-dom* $(\leq_L)$] $\Rrightarrow_m$ *in-dom* $(\leq_{L2})$) *l1*
and $((\leq_{L2}) \circ\circ (\leq_{R1}) \circ\circ (\leq_{L2})) \leq ((\leq_{R1}) \circ\circ (\leq_{L2}))$
and *in-codom* $((\leq_{L2}) \circ\circ (\leq_{R1}) \circ\circ (\leq_{L2})) \leq$ *in-codom* $(\leq_{R1})$
shows $((\leq_L) \trianglelefteq_h (\leq_R))$ *l r*
⟨*proof*⟩

lemma *half-galois-prop-right-left-rightI'*:
assumes $((\leq_{R1}) \Rrightarrow_m (\leq_{L1}))$ *r1*
and *inflationary-unit1*: *inflationary-on* (*in-dom* $(\leq_{L1})$) $(\leq_{L1})$ η_1
and *inflationary-counit1*: $\bigwedge y\ z.\ y \leq_{R1} r2\ z \Longrightarrow y \leq_{R1} l1\ (r\ z)$
and *in-dom* $(\leq_{R1}) \leq$ *in-codom* $(\leq_{R1})$
and $((\leq_{R2})\ {}_h\trianglelefteq (\leq_{L2}))$ *r2 l2*
and *inflationary-unit2*: *inflationary-on* (*in-dom* $(\leq_{L2})$) $(\leq_{L2})$ η_2
and *trans-L2*: *transitive* $(\leq_{L2})$
and *mono-in-dom-l1*: ([*in-dom* $(\leq_L)$] $\Rrightarrow_m$ *in-dom* $(\leq_{L2})$) *l1*
and $((\leq_{L2}) \circ\circ (\leq_{R1}) \circ\circ (\leq_{L2})) \leq ((\leq_{L2}) \circ\circ (\leq_{R1}))$
and *in-dom* $((\leq_{L2}) \circ\circ (\leq_{R1}) \circ\circ (\leq_{L2})) \leq$ *in-dom* $(\leq_{R1})$
shows $((\leq_L) \trianglelefteq_h (\leq_R))$ *l r*
⟨*proof*⟩

lemma *galois-prop-left-rightI*:
assumes $((\leq_{R1}) \Rrightarrow_m (\leq_{L1}))$ *r1*
and $((\leq_{L1}) \trianglelefteq (\leq_{R1}))$ *l1 r1*
and *rel-equivalence-on* (*in-codom* $(\leq_{R1})$) $(\leq_{R1})$ ε_1
and *transitive* $(\leq_{R1})$
and $((\leq_{L2}) \Rrightarrow_m (\leq_{R2}))$ *l2*
and $((\leq_{R2})\ {}_h\trianglelefteq (\leq_{L2}))$ *r2 l2*
and *inflationary-on* (*in-dom* $(\leq_{L2})$) $(\leq_{L2})$ η_2
and *preorder-on* (*in-field* $(\leq_{L2})$) $(\leq_{L2})$
and *middle-compatible-codom*
shows $((\leq_L) \trianglelefteq (\leq_R))$ *l r*
⟨*proof*⟩

lemma *galois-prop-left-rightI'*:
assumes $((\leq_{R1}) \Rrightarrow_m (\leq_{L1}))$ *r1*
and $((\leq_{L1})\ {}_h\trianglelefteq (\leq_{R1}))$ *l1 r1*
and *inflationary-on* (*in-dom* $(\leq_{L1})$) $(\leq_{L1})$ η_1
and *rel-equiv-counit1*: *rel-equivalence-on* (*in-field* $(\leq_{R1})$) $(\leq_{R1})$ ε_1
and *trans-R1*: *transitive* $(\leq_{R1})$
and $((\leq_{L2}) \Rrightarrow_m (\leq_{R2}))$ *l2*
and $((\leq_{R2})\ {}_h\trianglelefteq (\leq_{L2}))$ *r2 l2*
and *inflationary-on* (*in-dom* $(\leq_{L2})$) $(\leq_{L2})$ η_2

and *preorder-on* (*in-field* $(\le_{L2})$) $(\le_{L2})$
and *middle-compatible-dom*
shows $((\le_L) \trianglelefteq (\le_R))$ *l r*
⟨*proof*⟩

end

end

2.4.3 Monotonicity

theory *Transport-Compositions-Generic-Monotone*
imports
Transport-Compositions-Generic-Base
begin

context *transport-comp*
begin

lemma *mono-wrt-rel-leftI*:
assumes $((\le_{L1})\ {}_h\trianglelefteq\ (\le_{R1}))$ *l1 r1*
and $((\le_{L2}) \Rrightarrow_m (\le_{R2}))$ *l2*
and *inflationary-unit2*: *inflationary-on* (*in-codom* $(\le_{L2})$) $(\le_{L2})$ η_2
and $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le ((\le_{L2}) \circ\circ (\le_{R1}))$
and *in-codom* $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le$ *in-codom* $(\le_{L2})$
shows $((\le_L) \Rrightarrow_m (\le_R))$ *l*
⟨*proof*⟩

lemma *mono-wrt-rel-leftI'*:
assumes $((\le_{L1})\ {}_h\trianglelefteq\ (\le_{R1}))$ *l1 r1*
and $((\le_{L2}) \Rrightarrow_m (\le_{R2}))$ *l2*
and $((\le_{L2}) \trianglelefteq_h (\le_{R2}))$ *l2 r2*
and *refl-L2*: *reflexive-on* (*in-dom* $(\le_{L2})$) $(\le_{L2})$
and $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le ((\le_{R1}) \circ\circ (\le_{L2}))$
and *in-dom* $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le$ *in-dom* $(\le_{L2})$
shows $((\le_L) \Rrightarrow_m (\le_R))$ *l*
⟨*proof*⟩

end

end

2.4.4 Galois Connection

theory *Transport-Compositions-Generic-Galois-Connection*
imports
Transport-Compositions-Generic-Galois-Property
Transport-Compositions-Generic-Monotone

begin

context *transport-comp*
begin

interpretation *flip* : *transport-comp R2 L2 r2 l2 R1 L1 r1 l1*
rewrites *flip.t2.unit* $= \varepsilon_1$ **and** *flip.t1.counit* $\equiv \eta_2$
⟨*proof*⟩

lemma *galois-connection-left-rightI*:
assumes $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*
and $((\le_{L1}) \trianglelefteq (\le_{R1}))$ *l1 r1*
and *rel-equivalence-on* (*in-codom* $(\le_{R1})$) $(\le_{R1})$ ε_1
and *transitive* $(\le_{R1})$
and $((\le_{L2}) \Rrightarrow_m (\le_{R2}))$ *l2*
and $((\le_{R2})\ {}_h\trianglelefteq (\le_{L2}))$ *r2 l2*
and *inflationary-on* (*in-field* $(\le_{L2})$) $(\le_{L2})$ η_2
and *preorder-on* (*in-field* $(\le_{L2})$) $(\le_{L2})$
and *middle-compatible-codom*
shows $((\le_L) \dashv (\le_R))$ *l r*
⟨*proof*⟩

lemma *galois-connection-left-rightI'*:
assumes $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*
and $((\le_{L1})\ {}_h\trianglelefteq (\le_{R1}))$ *l1 r1*
and $((\le_{R1}) \trianglelefteq_h (\le_{L1}))$ *r1 l1*
and *inflationary-on* (*in-dom* $(\le_{L1})$) $(\le_{L1})$ η_1
and *rel-equivalence-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$ ε_1
and *transitive* $(\le_{R1})$
and $((\le_{L2}) \Rrightarrow_m (\le_{R2}))$ *l2*
and $((\le_{L2}) \trianglelefteq_h (\le_{R2}))$ *l2 r2*
and $((\le_{R2})\ {}_h\trianglelefteq (\le_{L2}))$ *r2 l2*
and *inflationary-on* (*in-dom* $(\le_{L2})$) $(\le_{L2})$ η_2
and *preorder-on* (*in-field* $(\le_{L2})$) $(\le_{L2})$
and *middle-compatible-dom*
shows $((\le_L) \dashv (\le_R))$ *l r*
⟨*proof*⟩

corollary *galois-connection-left-right-if-galois-equivalenceI*:
assumes $((\le_{L1}) \equiv_G (\le_{R1}))$ *l1 r1*
and *preorder-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$
and $((\le_{L2}) \equiv_G (\le_{R2}))$ *l2 r2*
and *preorder-on* (*in-field* $(\le_{L2})$) $(\le_{L2})$
and *middle-compatible-codom*
shows $((\le_L) \dashv (\le_R))$ *l r*
⟨*proof*⟩

corollary *galois-connection-left-right-if-order-equivalenceI*:
assumes $((\le_{L1}) \equiv_o (\le_{R1}))$ *l1 r1*

and *transitive* $(\le_{R1})$
and $((\le_{L2}) \equiv_o (\le_{R2}))$ *l2 r2*
and *transitive* $(\le_{L2})$
and *middle-compatible-codom*
shows $((\le_L) \dashv (\le_R))$ *l r*
⟨*proof*⟩

end

end

2.4.5 Galois Equivalence

theory *Transport-Compositions-Generic-Galois-Equivalence*
imports
Transport-Compositions-Generic-Galois-Connection
begin

context *transport-comp*
begin

interpretation *flip* : *transport-comp R2 L2 r2 l2 R1 L1 r1 l1*
rewrites *flip.t2.unit* $= \varepsilon_1$ **and** *flip.t1.counit* $\equiv \eta_2$ **and** *flip.t1.unit* $\equiv \varepsilon_2$
⟨*proof*⟩

lemma *galois-equivalenceI*:
assumes $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*
and $((\le_{L1}) \trianglelefteq (\le_{R1}))$ *l1 r1*
and *rel-equivalence-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$ ε_1
and *transitive* $(\le_{R1})$
and $((\le_{L2}) \Rrightarrow_m (\le_{R2}))$ *l2*
and $((\le_{R2}) \trianglelefteq (\le_{L2}))$ *r2 l2*
and *rel-equivalence-on* (*in-field* $(\le_{L2})$) $(\le_{L2})$ η_2
and *transitive* $(\le_{L2})$
and *middle-compatible-codom*
shows $((\le_L) \equiv_G (\le_R))$ *l r*
⟨*proof*⟩

lemma *galois-equivalenceI'*:
assumes $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*
and $((\le_{L1})\ {}_h\trianglelefteq (\le_{R1}))$ *l1 r1*
and $((\le_{R1}) \trianglelefteq_h (\le_{L1}))$ *r1 l1*
and *inflationary-on* (*in-dom* $(\le_{L1})$) $(\le_{L1})$ η_1
and *rel-equivalence-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$ ε_1
and *transitive* $(\le_{R1})$
and $((\le_{L2}) \Rrightarrow_m (\le_{R2}))$ *l2*
and $((\le_{L2}) \trianglelefteq_h (\le_{R2}))$ *l2 r2*
and $((\le_{R2})\ {}_h\trianglelefteq (\le_{L2}))$ *r2 l2*

and *rel-equivalence-on* (*in-field* ($\le_{L2}$)) ($\le_{L2}$) η_2
and *inflationary-on* (*in-dom* ($\le_{R2}$)) ($\le_{R2}$) ε_2
and *transitive* ($\le_{L2}$)
and *middle-compatible-dom*
shows (($\le_L$) $\equiv_G$ ($\le_R$)) *l r*
⟨*proof*⟩

corollary *galois-equivalence-if-galois-equivalenceI*:
assumes (($\le_{L1}$) $\equiv_G$ ($\le_{R1}$)) *l1 r1*
and *preorder-on* (*in-field* ($\le_{R1}$)) ($\le_{R1}$)
and (($\le_{L2}$) $\equiv_G$ ($\le_{R2}$)) *l2 r2*
and *preorder-on* (*in-field* ($\le_{L2}$)) ($\le_{L2}$)
and *middle-compatible-codom*
shows (($\le_L$) $\equiv_G$ ($\le_R$)) *l r*
⟨*proof*⟩

corollary *galois-equivalence-if-order-equivalenceI*:
assumes (($\le_{L1}$) $\equiv_o$ ($\le_{R1}$)) *l1 r1*
and *transitive* ($\le_{R1}$)
and (($\le_{L2}$) $\equiv_o$ ($\le_{R2}$)) *l2 r2*
and *transitive* ($\le_{L2}$)
and *middle-compatible-codom*
shows (($\le_L$) $\equiv_G$ ($\le_R$)) *l r*
⟨*proof*⟩

end

end

2.4.6 Galois Relator

theory *Transport-Compositions-Generic-Galois-Relator*
imports
Transport-Compositions-Generic-Base
begin

context *transport-comp*
begin

interpretation *flip* : *transport-comp R2 L2 r2 l2 R1 L1 r1 l1*
rewrites *flip.t2.unit* $\equiv \varepsilon_1$
⟨*proof*⟩

lemma *left-Galois-le-comp-left-GaloisI*:
assumes *mono-r1*: (($\le_{R1}$) $\Rrightarrow_m$ ($\le_{L1}$)) *r1*
and *galois-prop1*: (($\le_{L1}$) $\trianglelefteq$ ($\le_{R1}$)) *l1 r1*
and *preorder-R1*: *preorder-on* (*in-field* ($\le_{R1}$)) ($\le_{R1}$)
and *rel-comp-le*: (($\le_{R1}$) ∘∘ ($\le_{L2}$) ∘∘ ($\le_{R1}$)) $\le$ (($\le_{R1}$) ∘∘ ($\le_{L2}$))

and *mono-in-codom-r2*: ([*in-codom* ($\leq_R$)] $\Rrightarrow_m$ *in-codom* ($\leq_{R1}$)) *r2*
shows ($_L\lessapprox$) $\leq$ (($_{L1}\lessapprox$) $\circ\circ$ ($_{L2}\lessapprox$))
⟨*proof*⟩

lemma *comp-left-Galois-le-left-GaloisI*:
assumes *mono-r1*: (($\leq_{R1}$) $\Rrightarrow_m$ ($\leq_{L1}$)) *r1*
and *half-galois-prop-left1*: (($\leq_{L1}$) $_h\trianglelefteq$ ($\leq_{R1}$)) *l1 r1*
and *half-galois-prop-right1*: (($\leq_{R1}$) $\trianglelefteq_h$ ($\leq_{L1}$)) *r1 l1*
and *refl-R1*: *reflexive-on* (*in-codom* ($\leq_{R1}$)) ($\leq_{R1}$)
and *mono-l2*: (($\leq_{L2}$) $\Rrightarrow_m$ ($\leq_{R2}$)) *l2*
and *refl-L2*: *reflexive-on* (*in-dom* ($\leq_{L2}$)) ($\leq_{L2}$)
and *in-codom-rel-comp-le*: *in-codom* (($\leq_{L2}$) $\circ\circ$ ($\leq_{R1}$) $\circ\circ$ ($\leq_{L2}$)) $\leq$ *in-codom* ($\leq_{R1}$)
shows (($_{L1}\lessapprox$) $\circ\circ$ ($_{L2}\lessapprox$)) $\leq$ ($_L\lessapprox$)
⟨*proof*⟩

corollary *left-Galois-eq-comp-left-GaloisI*:
assumes (($\leq_{R1}$) $\Rrightarrow_m$ ($\leq_{L1}$)) *r1*
and (($\leq_{L1}$) $\trianglelefteq$ ($\leq_{R1}$)) *l1 r1*
and (($\leq_{R1}$) $\trianglelefteq_h$ ($\leq_{L1}$)) *r1 l1*
and *preorder-on* (*in-field* ($\leq_{R1}$)) ($\leq_{R1}$)
and (($\leq_{L2}$) $\Rrightarrow_m$ ($\leq_{R2}$)) *l2*
and *reflexive-on* (*in-dom* ($\leq_{L2}$)) ($\leq_{L2}$)
and (($\leq_{R1}$) $\circ\circ$ ($\leq_{L2}$) $\circ\circ$ ($\leq_{R1}$)) $\leq$ (($\leq_{R1}$) $\circ\circ$ ($\leq_{L2}$))
and ([*in-codom* ($\leq_R$)] $\Rrightarrow_m$ *in-codom* ($\leq_{R1}$)) *r2*
and *in-codom* (($\leq_{L2}$) $\circ\circ$ ($\leq_{R1}$) $\circ\circ$ ($\leq_{L2}$)) $\leq$ *in-codom* ($\leq_{R1}$)
shows ($_L\lessapprox$) = (($_{L1}\lessapprox$) $\circ\circ$ ($_{L2}\lessapprox$))
⟨*proof*⟩

corollary *left-Galois-eq-comp-left-GaloisI'*:
assumes (($\leq_{R1}$) $\Rrightarrow_m$ ($\leq_{L1}$)) *r1*
and (($\leq_{L1}$) $\trianglelefteq$ ($\leq_{R1}$)) *l1 r1*
and (($\leq_{R1}$) $\trianglelefteq_h$ ($\leq_{L1}$)) *r1 l1*
and *preorder-on* (*in-field* ($\leq_{R1}$)) ($\leq_{R1}$)
and (($\leq_{L2}$) $\Rrightarrow_m$ ($\leq_{R2}$)) *l2*
and (($\leq_{R2}$) $_h\trianglelefteq$ ($\leq_{L2}$)) *r2 l2*
and *reflexive-on* (*in-dom* ($\leq_{L2}$)) ($\leq_{L2}$)
and (($\leq_{R1}$) $\circ\circ$ ($\leq_{L2}$) $\circ\circ$ ($\leq_{R1}$)) $\leq$ (($\leq_{R1}$) $\circ\circ$ ($\leq_{L2}$))
and *in-codom* (($\leq_{L2}$) $\circ\circ$ ($\leq_{R1}$) $\circ\circ$ ($\leq_{L2}$)) $\leq$ *in-codom* ($\leq_{R1}$)
shows ($_L\lessapprox$) = (($_{L1}\lessapprox$) $\circ\circ$ ($_{L2}\lessapprox$))
⟨*proof*⟩

theorem *left-Galois-eq-comp-left-Galois-if-galois-connection-if-galois-equivalenceI'*:
assumes (($\leq_{L1}$) $\equiv_G$ ($\leq_{R1}$)) *l1 r1*
and *preorder-on* (*in-field* ($\leq_{R1}$)) ($\leq_{R1}$)
and (($\leq_{R2}$) $\dashv$ ($\leq_{L2}$)) *r2 l2*
and *reflexive-on* (*in-dom* ($\leq_{L2}$)) ($\leq_{L2}$)
and (($\leq_{R1}$) $\circ\circ$ ($\leq_{L2}$) $\circ\circ$ ($\leq_{R1}$)) $\leq$ (($\leq_{R1}$) $\circ\circ$ ($\leq_{L2}$))
and *in-codom* (($\leq_{L2}$) $\circ\circ$ ($\leq_{R1}$) $\circ\circ$ ($\leq_{L2}$)) $\leq$ *in-codom* ($\leq_{R1}$)

shows $({}_{L}\lessapprox) = (({}_{L1}\lessapprox) \circ\circ ({}_{L2}\lessapprox))$
⟨*proof*⟩

corollary *left-Galois-eq-comp-left-Galois-if-galois-connection-if-galois-equivalenceI*:
assumes $((\le_{L1}) \equiv_G (\le_{R1}))$ *l1 r1*
and *preorder-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$
and $((\le_{R2}) \dashv (\le_{L2}))$ *r2 l2*
and *reflexive-on* (*in-field* $(\le_{L2})$) $(\le_{L2})$
and *in-codom* $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le$ *in-codom* $(\le_{L2})$
and $((\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2})) \le ((\le_{R1}) \circ\circ (\le_{L2}))$
and *in-codom* $((\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2})) \le$ *in-codom* $(\le_{R1})$
shows $({}_{L}\lessapprox) = (({}_{L1}\lessapprox) \circ\circ ({}_{L2}\lessapprox))$
⟨*proof*⟩

corollary *left-Galois-eq-comp-left-Galois-if-preorder-equivalenceI*:
assumes $((\le_{L1}) \equiv_{pre} (\le_{R1}))$ *l1 r1*
and $((\le_{R2}) \equiv_{pre} (\le_{L2}))$ *r2 l2*
and *in-codom* $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le$ *in-codom* $(\le_{L2})$
and $(\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2}) \le (\le_{R1}) \circ\circ (\le_{L2})$
and *in-codom* $((\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2})) \le$ *in-codom* $(\le_{R1})$
shows $({}_{L}\lessapprox) = (({}_{L1}\lessapprox) \circ\circ ({}_{L2}\lessapprox))$
⟨*proof*⟩

end

end

2.4.7 Basic Order Properties

theory *Transport-Compositions-Generic-Order-Base*
imports
Transport-Compositions-Generic-Base
begin

context *transport-comp*
begin

interpretation *flip1* : *galois R1 L1 r1 l1* ⟨*proof*⟩

Reflexivity

lemma *reflexive-on-in-dom-leftI*:
assumes *galois-prop*: $((\le_{L1}) \trianglelefteq (\le_{R1}))$ *l1 r1*
and *in-dom-L1-le*: *in-dom* $(\le_{L1}) \le$ *in-codom* $(\le_{L1})$
and *refl-R1*: *reflexive-on* (*in-dom* $(\le_{R1})$) $(\le_{R1})$
and *refl-L2*: *reflexive-on* (*in-dom* $(\le_{L2})$) $(\le_{L2})$
and *mono-in-dom-l1*: ([*in-dom* $(\le_L)$] $\Rrightarrow_m$ *in-dom* $(\le_{L2})$) *l1*
shows *reflexive-on* (*in-dom* $(\le_L)$) $(\le_L)$
⟨*proof*⟩

lemma *reflexive-on-in-codom-leftI*:
assumes *L1-r1-l1I*: $\bigwedge x.$ *in-dom* $(\le_{L1})$ $x \Longrightarrow$ *l1* $x \le_{R1}$ *l1* $x \Longrightarrow x \le_{L1}$ *r1* (*l1* x)
and *in-codom-L1-le*: *in-codom* $(\le_{L1}) \le$ *in-dom* $(\le_{L1})$
and *refl-R1*: *reflexive-on* (*in-codom* $(\le_{R1})$) $(\le_{R1})$
and *refl-L2*: *reflexive-on* (*in-codom* $(\le_{L2})$) $(\le_{L2})$
and *mono-in-codom-l1*: ([*in-codom* $(\le_{L})$] $\Rrightarrow_m$ *in-codom* $(\le_{L2})$) *l1*
shows *reflexive-on* (*in-codom* $(\le_{L})$) $(\le_{L})$
⟨*proof*⟩

corollary *reflexive-on-in-field-leftI*:
assumes $((\le_{L1}) \trianglelefteq (\le_{R1}))$ *l1 r1*
and *in-codom* $(\le_{L1})$ = *in-dom* $(\le_{L1})$
and *reflexive-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$
and *reflexive-on* (*in-field* $(\le_{L2})$) $(\le_{L2})$
and ([*in-field* $(\le_{L})$] $\Rrightarrow_m$ *in-field* $(\le_{L2})$) *l1*
shows *reflexive-on* (*in-field* $(\le_{L})$) $(\le_{L})$
⟨*proof*⟩

Transitivity

There are many similar proofs for transitivity. They slightly differ in their assumptions, particularly which of $(\le_{R1})$ and $(\le_{L2})$ has to be transitive and the order of commutativity for the relations.

In the following, we just give two of them that suffice for many purposes.

lemma *transitive-leftI*:
assumes $((\le_{L1})\ {}_h\!\trianglelefteq (\le_{R1}))$ *l1 r1*
and *trans-L2*: *transitive* $(\le_{L2})$
and *R1-L2-R1-le*: $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le ((\le_{L2}) \circ\circ (\le_{R1}))$
shows *transitive* $(\le_{L})$
⟨*proof*⟩

lemma *transitive-leftI'*:
assumes *galois-prop*: $((\le_{L1}) \trianglelefteq (\le_{R1}))$ *l1 r1*
and *trans-L2*: *transitive* $(\le_{L2})$
and *R1-L2-R1-le*: $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le ((\le_{R1}) \circ\circ (\le_{L2}))$
shows *transitive* $(\le_{L})$
⟨*proof*⟩

Preorders

lemma *preorder-on-in-field-leftI*:
assumes $((\le_{L1}) \trianglelefteq (\le_{R1}))$ *l1 r1*
and *in-codom* $(\le_{L1})$ = *in-dom* $(\le_{L1})$
and *reflexive-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$
and *preorder-on* (*in-field* $(\le_{L2})$) $(\le_{L2})$
and *mono-in-codom-l1*: ([*in-codom* $(\le_{L})$] $\Rrightarrow_m$ *in-codom* $(\le_{L2})$) *l1*
and *R1-L2-R1-le*: $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le ((\le_{L2}) \circ\circ (\le_{R1}))$
shows *preorder-on* (*in-field* $(\le_{L})$) $(\le_{L})$

⟨*proof*⟩

lemma *preorder-on-in-field-leftI'*:
assumes $((\le_{L1}) \trianglelefteq (\le_{R1}))$ *l1 r1*
and *in-codom* $(\le_{L1})$ = *in-dom* $(\le_{L1})$
and *reflexive-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$
and *preorder-on* (*in-field* $(\le_{L2})$) $(\le_{L2})$
and *mono-in-dom-l1*: ([*in-dom* $(\le_L)$] $\Rrightarrow_m$ *in-dom* $(\le_{L2})$) *l1*
and *R1-L2-R1-le*: $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le ((\le_{R1}) \circ\circ (\le_{L2}))$
shows *preorder-on* (*in-field* $(\le_L)$) $(\le_L)$
⟨*proof*⟩

Symmetry

lemma *symmetric-leftI*:
assumes $((\le_{L1}) \trianglelefteq (\le_{R1}))$ *l1 r1*
and *in-codom* $(\le_{L1})$ = *in-dom* $(\le_{L1})$
and *symmetric* $(\le_{R1})$
and *symmetric* $(\le_{L2})$
shows *symmetric* $(\le_L)$
⟨*proof*⟩

lemma *partial-equivalence-rel-leftI*:
assumes $((\le_{L1}) \trianglelefteq (\le_{R1}))$ *l1 r1*
and *in-codom* $(\le_{L1})$ = *in-dom* $(\le_{L1})$
and *symmetric* $(\le_{R1})$
and *partial-equivalence-rel* $(\le_{L2})$
and $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le ((\le_{L2}) \circ\circ (\le_{R1}))$
shows *partial-equivalence-rel* $(\le_L)$
⟨*proof*⟩

lemma *partial-equivalence-rel-leftI'*:
assumes $((\le_{L1}) \trianglelefteq (\le_{R1}))$ *l1 r1*
and *in-codom* $(\le_{L1})$ = *in-dom* $(\le_{L1})$
and *symmetric* $(\le_{R1})$
and *partial-equivalence-rel* $(\le_{L2})$
and $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le ((\le_{R1}) \circ\circ (\le_{L2}))$
shows *partial-equivalence-rel* $(\le_L)$
⟨*proof*⟩

end

end

2.4.8 Order Equivalence

theory *Transport-Compositions-Generic-Order-Equivalence*
imports
Transport-Compositions-Generic-Monotone

begin

context *transport-comp*
begin

context
begin

interpretation *flip* : *transport-comp R2 L2 r2 l2 R1 L1 r1 l1* ⟨*proof*⟩

Unit

Inflationary **lemma** *inflationary-on-in-dom-unitI*:
assumes $((\leq_{R1}) \Rrightarrow_m (\leq_{L1}))$ *r1*
and $((\leq_{L1})\ _h\trianglelefteq (\leq_{R1}))$ *l1 r1*
and *inflationary-unit1*: *inflationary-on* (*in-dom* $(\leq_{L1})$) $(\leq_{L1})$ η_1
and *inflationary-counit1*: *inflationary-on* (*in-codom* $(\leq_{R1})$) $(\leq_{R1})$ ε_1
and *refl-R1*: *reflexive-on* (*in-dom* $(\leq_{R1})$) $(\leq_{R1})$
and *inflationary-unit2*: *inflationary-on* (*in-dom* $(\leq_{L2})$) $(\leq_{L2})$ η_2
and *refl-L2*: *reflexive-on* (*in-dom* $(\leq_{L2})$) $(\leq_{L2})$
and *mono-in-dom-l1*: ([*in-dom* $(\leq_L)$] $\Rrightarrow_m$ *in-dom* $(\leq_{L2})$) *l1*
and *in-codom-rel-comp-le*: *in-codom* $((\leq_{L2}) \circ\circ (\leq_{R1}) \circ\circ (\leq_{L2})) \leq$ *in-codom* $((\leq_{R1}))$
shows *inflationary-on* (*in-dom* $(\leq_L)$) $(\leq_L)$ η
⟨*proof*⟩

lemma *inflationary-on-in-codom-unitI*:
assumes $((\leq_{R1}) \Rrightarrow_m (\leq_{L1}))$ *r1*
and *inflationary-unit1*: *inflationary-on* (*in-codom* $(\leq_{L1})$) $(\leq_{L1})$ η_1
and *inflationary-counit1*: *inflationary-on* (*in-codom* $(\leq_{R1})$) $(\leq_{R1})$ ε_1
and *refl-R1*: *reflexive-on* (*in-codom* $(\leq_{R1})$) $(\leq_{R1})$
and *inflationary-unit2*: *inflationary-on* (*in-codom* $(\leq_{L2})$) $(\leq_{L2})$ η_2
and *refl-L2*: *reflexive-on* (*in-codom* $(\leq_{L2})$) $(\leq_{L2})$
and *mono-in-codom-l1*: ([*in-codom* $(\leq_L)$] $\Rrightarrow_m$ *in-codom* $(\leq_{L2})$) *l1*
and *in-codom-rel-comp-le*: *in-codom* $((\leq_{L2}) \circ\circ (\leq_{R1}) \circ\circ (\leq_{L2})) \leq$ *in-codom* $((\leq_{R1}))$
shows *inflationary-on* (*in-codom* $(\leq_L)$) $(\leq_L)$ η
⟨*proof*⟩

corollary *inflationary-on-in-field-unitI*:
assumes $((\leq_{R1}) \Rrightarrow_m (\leq_{L1}))$ *r1*
and $((\leq_{L1})\ _h\trianglelefteq (\leq_{R1}))$ *l1 r1*
and *inflationary-on* (*in-field* $(\leq_{L1})$) $(\leq_{L1})$ η_1
and *inflationary-on* (*in-codom* $(\leq_{R1})$) $(\leq_{R1})$ ε_1
and *reflexive-on* (*in-field* $(\leq_{R1})$) $(\leq_{R1})$
and *inflationary-on* (*in-field* $(\leq_{L2})$) $(\leq_{L2})$ η_2
and *reflexive-on* (*in-field* $(\leq_{L2})$) $(\leq_{L2})$
and ([*in-dom* $(\leq_L)$] $\Rrightarrow_m$ *in-dom* $(\leq_{L2})$) *l1*
and ([*in-codom* $(\leq_L)$] $\Rrightarrow_m$ *in-codom* $(\leq_{L2})$) *l1*

and *in-codom* $((\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2})) \le$ *in-codom* $((\le_{R1}))$
shows *inflationary-on* (*in-field* $(\le_L)$) $(\le_L)$ η
⟨*proof*⟩

Deflationary

lemma *deflationary-on-in-dom-unitI*:
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))$ *l1* $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*
and *refl-L1*: *reflexive-on* (*in-dom* $(\le_{L1})$) $(\le_{L1})$
and *in-dom-R1-le-in-codom-R1*: *in-dom* $(\le_{R1}) \le$ *in-codom* $(\le_{R1})$
and *deflationary-L2*: *deflationary-on* (*in-dom* $(\le_{L2})$) $(\le_{L2})$ η_2
and *refl-L2*: *reflexive-on* (*in-dom* $(\le_{L2})$) $(\le_{L2})$
and *mono-in-dom-l1*: ([*in-dom* $(\le_L)$] $\Rrightarrow_m$ *in-dom* $(\le_{L2})$) *l1*
and *in-dom-rel-comp-le*: *in-dom* $((\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2})) \le$ *in-dom* $((\le_{R1}))$
shows *deflationary-on* (*in-dom* $(\le_L)$) $(\le_L)$ η
⟨*proof*⟩

lemma *deflationary-on-in-codom-unitI*:
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))$ *l1* $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*
and *refl-L1*: *reflexive-on* (*in-codom* $(\le_{L1})$) $(\le_{L1})$
and *in-dom-R1-le-in-codom-R1*: *in-dom* $(\le_{R1}) \le$ *in-codom* $(\le_{R1})$
and *deflationary-L2*: *deflationary-on* (*in-codom* $(\le_{L2})$) $(\le_{L2})$ η_2
and *refl-L2*: *reflexive-on* (*in-codom* $(\le_{L2})$) $(\le_{L2})$
and *mono-in-codom-l1*: ([*in-codom* $(\le_L)$] $\Rrightarrow_m$ *in-codom* $(\le_{L2})$) *l1*
and *in-dom-rel-comp-le*: *in-dom* $((\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2})) \le$ *in-dom* $((\le_{R1}))$
shows *deflationary-on* (*in-codom* $(\le_L)$) $(\le_L)$ η
⟨*proof*⟩

corollary *deflationary-on-in-field-unitI*:
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))$ *l1* $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*
and *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and *in-dom* $(\le_{R1}) \le$ *in-codom* $(\le_{R1})$
and *deflationary-on* (*in-field* $(\le_{L2})$) $(\le_{L2})$ η_2
and *reflexive-on* (*in-field* $(\le_{L2})$) $(\le_{L2})$
and ([*in-dom* $(\le_L)$] $\Rrightarrow_m$ *in-dom* $(\le_{L2})$) *l1*
and ([*in-codom* $(\le_L)$] $\Rrightarrow_m$ *in-codom* $(\le_{L2})$) *l1*
and *in-dom* $((\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2})) \le$ *in-dom* $((\le_{R1}))$
shows *deflationary-on* (*in-field* $(\le_L)$) $(\le_L)$ η
⟨*proof*⟩

Relational Equivalence

corollary *rel-equivalence-on-in-field-unitI*:
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))$ *l1* $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*
and $((\le_{L1})\ {}_h\trianglelefteq\ (\le_{R1}))$ *l1 r1*
and *inflationary-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$ η_1
and *inflationary-on* (*in-codom* $(\le_{R1})$) $(\le_{R1})$ ε_1
and *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and *reflexive-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$
and *rel-equivalence-on* (*in-field* $(\le_{L2})$) $(\le_{L2})$ η_2
and *reflexive-on* (*in-field* $(\le_{L2})$) $(\le_{L2})$

and $([in\text{-}dom\ (\le_{L})] \Rrightarrow_m in\text{-}dom\ (\le_{L2}))\ l1$
and $([in\text{-}codom\ (\le_{L})] \Rrightarrow_m in\text{-}codom\ (\le_{L2}))\ l1$
and $in\text{-}dom\ ((\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2})) \le in\text{-}dom\ ((\le_{R1}))$
and $in\text{-}codom\ ((\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2})) \le in\text{-}codom\ ((\le_{R1}))$
shows $rel\text{-}equivalence\text{-}on\ (in\text{-}field\ (\le_{L}))\ (\le_{L})\ \eta$
⟨*proof*⟩

Counit

Corresponding lemmas for the counit can be obtained by flipping the interpretation of the locale, i.e. *interpretation flip : transport-comp R2 L2 r2 l2 R1 L1 r1 l1 rewrites flip.t2.unit* $\equiv \varepsilon_1$ *and flip.t2.counit* $\equiv \eta_1$ *and flip.t1.unit* $\equiv \varepsilon_2$ *and flip.t1.counit* $\equiv \eta_2$ *and flip.unit* $\equiv \varepsilon$ *and flip.counit* $\equiv \eta$ *unfolding transport-comp.transport-defs by* (*auto simp*: *order-functors.flip-counit-eq-unit*)

end

Order Equivalence

interpretation *flip* : *transport-comp R2 L2 r2 l2 R1 L1 r1 l1*
rewrites *flip.t2.unit* $\equiv \varepsilon_1$ **and** *flip.t2.counit* $\equiv \eta_1$
and *flip.t1.unit* $\equiv \varepsilon_2$ **and** *flip.t1.counit* $\equiv \eta_2$
and *flip.counit* $\equiv \eta$ **and** *flip.unit* $\equiv \varepsilon$
⟨*proof*⟩

lemma *order-equivalenceI*:
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1\ ((\le_{R1}) \Rrightarrow_m (\le_{L1}))\ r1$
and $((\le_{L1})\ {}_h\trianglelefteq\ (\le_{R1}))\ l1\ r1$
and $inflationary\text{-}on\ (in\text{-}field\ (\le_{L1}))\ (\le_{L1})\ \eta_1$
and *rel-equiv-counit1*: $rel\text{-}equivalence\text{-}on\ (in\text{-}field\ (\le_{R1}))\ (\le_{R1})\ \varepsilon_1$
and $reflexive\text{-}on\ (in\text{-}field\ (\le_{L1}))\ (\le_{L1})$
and $reflexive\text{-}on\ (in\text{-}field\ (\le_{R1}))\ (\le_{R1})$
and $((\le_{R2}) \Rrightarrow_m (\le_{L2}))\ r2\ ((\le_{L2}) \Rrightarrow_m (\le_{R2}))\ l2$
and $((\le_{R2})\ {}_h\trianglelefteq\ (\le_{L2}))\ r2\ l2$
and *rel-equiv-unit2*: $rel\text{-}equivalence\text{-}on\ (in\text{-}field\ (\le_{L2}))\ (\le_{L2})\ \eta_2$
and $inflationary\text{-}on\ (in\text{-}field\ (\le_{R2}))\ (\le_{R2})\ \varepsilon_2$
and $reflexive\text{-}on\ (in\text{-}field\ (\le_{L2}))\ (\le_{L2})$
and $reflexive\text{-}on\ (in\text{-}field\ (\le_{R2}))\ (\le_{R2})$
and *middle-compatible*: *middle-compatible-codom*
shows $((\le_{L}) \equiv_o (\le_{R}))\ l\ r$
⟨*proof*⟩

corollary *order-equivalence-if-order-equivalenceI*:
assumes $((\le_{L1}) \equiv_o (\le_{R1}))\ l1\ r1$
and $reflexive\text{-}on\ (in\text{-}field\ (\le_{L1}))\ (\le_{L1})$
and $transitive\ (\le_{R1})$
and $((\le_{L2}) \equiv_o (\le_{R2}))\ l2\ r2$
and $transitive\ (\le_{L2})$
and $reflexive\text{-}on\ (in\text{-}field\ (\le_{R2}))\ (\le_{R2})$
and *middle-compatible-codom*

shows $((\le_L) \equiv_o (\le_R))$ *l r*
⟨*proof*⟩

corollary *order-equivalence-if-galois-equivalenceI*:
assumes $((\le_{L1}) \equiv_G (\le_{R1}))$ *l1 r1*
and *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and *reflexive-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$
and $((\le_{L2}) \equiv_G (\le_{R2}))$ *l2 r2*
and *reflexive-on* (*in-field* $(\le_{L2})$) $(\le_{L2})$
and *reflexive-on* (*in-field* $(\le_{R2})$) $(\le_{R2})$
and *middle-compatible-codom*
shows $((\le_L) \equiv_o (\le_R))$ *l r*
⟨*proof*⟩

end

end

theory *Transport-Compositions-Generic*
imports
Transport-Compositions-Generic-Galois-Equivalence
Transport-Compositions-Generic-Galois-Relator
Transport-Compositions-Generic-Order-Base
Transport-Compositions-Generic-Order-Equivalence
begin

Summary of Main Results

Closure of Order and Galois Concepts **context** *transport-comp*
begin

interpretation *flip* : *transport-comp R2 L2 r2 l2 R1 L1 r1 l1* ⟨*proof*⟩

lemma *preorder-galois-connection-if-galois-equivalenceI*:
assumes $((\le_{L1}) \equiv_G (\le_{R1}))$ *l1 r1*
and *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and *preorder-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$
and $((\le_{L2}) \equiv_G (\le_{R2}))$ *l2 r2*
and *preorder-on* (*in-field* $(\le_{L2})$) $(\le_{L2})$
and *reflexive-on* (*in-field* $(\le_{R2})$) $(\le_{R2})$
and *middle-compatible-codom*
shows $((\le_L) \dashv_{pre} (\le_R))$ *l r*
⟨*proof*⟩

theorem *preorder-galois-connection-if-preorder-equivalenceI*:
assumes $((\le_{L1}) \equiv_{pre} (\le_{R1}))$ *l1 r1*
and $((\le_{L2}) \equiv_{pre} (\le_{R2}))$ *l2 r2*
and *middle-compatible-codom*

shows $((\le_L) \dashv_{pre} (\le_R))$ *l r*
⟨*proof*⟩

lemma *preorder-equivalence-if-galois-equivalenceI*:
assumes $((\le_{L1}) \equiv_G (\le_{R1}))$ *l1 r1*
and *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and *preorder-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$
and $((\le_{L2}) \equiv_G (\le_{R2}))$ *l2 r2*
and *preorder-on* (*in-field* $(\le_{L2})$) $(\le_{L2})$
and *reflexive-on* (*in-field* $(\le_{R2})$) $(\le_{R2})$
and *middle-compatible-codom*
shows $((\le_L) \equiv_{pre} (\le_R))$ *l r*
⟨*proof*⟩

theorem *preorder-equivalenceI*:
assumes $((\le_{L1}) \equiv_{pre} (\le_{R1}))$ *l1 r1*
and $((\le_{L2}) \equiv_{pre} (\le_{R2}))$ *l2 r2*
and *middle-compatible-codom*
shows $((\le_L) \equiv_{pre} (\le_R))$ *l r*
⟨*proof*⟩

theorem *partial-equivalence-rel-equivalenceI*:
assumes $((\le_{L1}) \equiv_{PER} (\le_{R1}))$ *l1 r1*
and $((\le_{L2}) \equiv_{PER} (\le_{R2}))$ *l2 r2*
and *middle-compatible-codom*
shows $((\le_L) \equiv_{PER} (\le_R))$ *l r*
⟨*proof*⟩

Simplification of Galois relator **theorem** *left-Galois-eq-comp-left-GaloisI*:
assumes $((\le_{L1}) \equiv_{pre} (\le_{R1}))$ *l1 r1*
and $((\le_{R2}) \dashv_{pre} (\le_{L2}))$ *r2 l2*
and *middle-compatible-codom*
shows $({}_{L}\lessapprox) = (({}_{L1}\lessapprox) \circ\circ ({}_{L2}\lessapprox))$
⟨*proof*⟩

For theorems with weaker assumptions, see $[\![((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*; *t1.galois-prop l1 r1*; *flip.t2.half-galois-prop-right*; *preorder-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$; $((\le_{L2}) \Rrightarrow_m (\le_{R2}))$ *l2*; *flip.t1.half-galois-prop-left*; *reflexive-on* (*in-dom* $(\le_{L2})$) $(\le_{L2})$; $(\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1}) \le (\le_{R1}) \circ\circ (\le_{L2})$; *in-codom* $((\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2})) \le$ *in-codom* $(\le_{R1})]\!] \Longrightarrow$ *flip.right-Galois* = *flip.t2.right-Galois* $\circ\circ$ *flip.t1.right-Galois*

$[\![$*t1.galois-equivalence*; *preorder-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$; *flip.t1.galois-connection*; *reflexive-on* (*in-field* $(\le_{L2})$) $(\le_{L2})$; *in-codom* $((\le_{R1}) \circ\circ (\le_{L2}) \circ\circ (\le_{R1})) \le$ *in-codom* $(\le_{L2})$; $(\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2}) \le (\le_{R1}) \circ\circ (\le_{L2})$; *in-codom* $((\le_{L2}) \circ\circ (\le_{R1}) \circ\circ (\le_{L2})) \le$ *in-codom* $(\le_{R1})]\!] \Longrightarrow$ *flip.right-Galois* = *flip.t2.right-Galois* $\circ\circ$ *flip.t1.right-Galois*.

Simplification of Compatibility Assumption See *Transport.Transport-Compositions-Gener*

end

end

2.5 Transport For Compositions

theory *Transport-Compositions*
  imports
    Transport-Compositions-Agree
    Transport-Compositions-Generic
begin

Summary We provide two ways to compose transportable components: a slightly intricate, generic one in *transport-comp* and another straightforward but less general one in *transport-comp-agree*. As a special case from the latter, we obtain *transport-comp-same*, which includes the cases most prominently covered in the literature.

Refer to [2] for more details.

end

2.6 Reflexive Relator

theory *Reflexive-Relator*
  imports
    Galois-Equivalences
    Galois-Relator
begin

definition *Refl-Rel* $R\ x\ y \equiv R\ x\ x \wedge R\ y\ y \wedge R\ x\ y$

bundle *Refl-Rel-syntax* **begin notation** *Refl-Rel* ((-$^{\oplus}$) [*1000*]) **end**
bundle *no-Refl-Rel-syntax* **begin no-notation** *Refl-Rel* ((-$^{\oplus}$) [*1000*]) **end**
unbundle *Refl-Rel-syntax*

lemma *Refl-RelI* [*intro*]:
  assumes $R\ x\ x$
  and $R\ y\ y$
  and $R\ x\ y$
  shows $R^{\oplus}\ x\ y$
  ⟨*proof*⟩

lemma *Refl-Rel-selfI* [*intro*]:
  assumes $R\ x\ x$
  shows $R^{\oplus}\ x\ x$
  ⟨*proof*⟩

lemma *Refl-RelE* [*elim*]:
assumes $R^{\oplus}$ *x y*
obtains *R x x R y y R x y*
⟨*proof*⟩

lemma *Refl-Rel-reflexive-on-in-field* [*iff*]:
reflexive-on (*in-field* $R^{\oplus}$) $R^{\oplus}$
⟨*proof*⟩

lemma *Refl-Rel-le-self* [*iff*]: $R^{\oplus} \leq R$ ⟨*proof*⟩

lemma *Refl-Rel-eq-self-if-reflexive-on* [*simp*]:
assumes *reflexive-on* (*in-field R*) *R*
shows $R^{\oplus} = R$
⟨*proof*⟩

lemma *reflexive-on-in-field-if-Refl-Rel-eq-self*:
assumes $R^{\oplus} = R$
shows *reflexive-on* (*in-field R*) *R*
⟨*proof*⟩

corollary *Refl-Rel-eq-self-iff-reflexive-on*:
$R^{\oplus} = R \longleftrightarrow$ *reflexive-on* (*in-field R*) *R*
⟨*proof*⟩

lemma *Refl-Rel-Refl-Rel-eq* [*simp*]: $(R^{\oplus})^{\oplus} = R^{\oplus}$
⟨*proof*⟩

lemma *rel-inv-Refl-Rel-eq* [*simp*]: $(R^{\oplus})^{-1} = (R^{-1})^{\oplus}$
⟨*proof*⟩

lemma *Refl-Rel-transitive-onI* [*intro*]:
assumes *transitive-on* (*P* :: *'a* ⇒ *bool*) (*R* :: *'a* ⇒ -)
shows *transitive-on P* $R^{\oplus}$
⟨*proof*⟩

corollary *Refl-Rel-transitiveI* [*intro*]:
assumes *transitive R*
shows *transitive* $R^{\oplus}$
⟨*proof*⟩

corollary *Refl-Rel-preorder-onI*:
assumes *transitive-on P R*
and $P \leq$ *in-field* $R^{\oplus}$
shows *preorder-on P* $R^{\oplus}$
⟨*proof*⟩

corollary *Refl-Rel-preorder-on-in-fieldI* [*intro*]:
assumes *transitive R*

shows *preorder-on* (*in-field* $R^{\oplus}$) $R^{\oplus}$
⟨*proof*⟩

lemma *Refl-Rel-symmetric-onI* [*intro*]:
assumes *symmetric-on* (P :: $'a \Rightarrow bool$) (R :: $'a \Rightarrow$ -)
shows *symmetric-on* P $R^{\oplus}$
⟨*proof*⟩

lemma *Refl-Rel-symmetricI* [*intro*]:
assumes *symmetric* R
shows *symmetric* $R^{\oplus}$
⟨*proof*⟩

lemma *Refl-Rel-partial-equivalence-rel-onI* [*intro*]:
assumes *partial-equivalence-rel-on* (P :: $'a \Rightarrow bool$) (R :: $'a \Rightarrow$ -)
shows *partial-equivalence-rel-on* P $R^{\oplus}$
⟨*proof*⟩

lemma *Refl-Rel-partial-equivalence-relI* [*intro*]:
assumes *partial-equivalence-rel* R
shows *partial-equivalence-rel* $R^{\oplus}$
⟨*proof*⟩

lemma *Refl-Rel-app-leftI*:
assumes R (f x) y
and *in-field* $S^{\oplus}$ x
and *in-field* $R^{\oplus}$ y
and ($S \Rrightarrow_m R$) f
shows $R^{\oplus}$ (f x) y
⟨*proof*⟩

corollary *Refl-Rel-app-rightI*:
assumes R x (f y)
and *in-field* $S^{\oplus}$ y
and *in-field* $R^{\oplus}$ x
and ($S \Rrightarrow_m R$) f
shows $R^{\oplus}$ x (f y)
⟨*proof*⟩

lemma *mono-wrt-rel-Refl-Rel-Refl-Rel-if-mono-wrt-rel* [*intro*]:
assumes ($R \Rrightarrow_m S$) f
shows ($R^{\oplus} \Rrightarrow_m S^{\oplus}$) f
⟨*proof*⟩

context *galois*
begin

interpretation *gR* : *galois* $(\leq_L)^{\oplus}$ $(\leq_R)^{\oplus}$ l r ⟨*proof*⟩

lemma *Galois-Refl-RelI*:
assumes $((\leq_R) \Rrightarrow_m (\leq_L))\ r$
and *in-field* $(\leq_L)^\oplus\ x$
and *in-field* $(\leq_R)^\oplus\ y$
and *in-codom* $(\leq_R)\ y \Longrightarrow x\ {}_L\lessapprox\ y$
shows $(galois\text{-}rel.Galois\ ((\leq_L)^\oplus)\ ((\leq_R)^\oplus)\ r)\ x\ y$
⟨*proof*⟩

lemma *half-galois-prop-left-Refl-Rel-left-rightI*:
assumes $((\leq_L) \Rrightarrow_m (\leq_R))\ l$
and $((\leq_L)\ {}_h\unlhd\ (\leq_R))\ l\ r$
shows $((\leq_L)^\oplus\ {}_h\unlhd\ (\leq_R)^\oplus)\ l\ r$
⟨*proof*⟩

interpretation *flip-inv* : *galois* $(\geq_R)\ (\geq_L)\ r\ l$
rewrites $((\geq_R) \Rrightarrow_m (\geq_L)) \equiv ((\leq_R) \Rrightarrow_m (\leq_L))$
and $\bigwedge R.\ (R^{-1})^\oplus \equiv (R^\oplus)^{-1}$
and $\bigwedge R\ S\ f\ g.\ (R^{-1}\ {}_h\unlhd\ S^{-1})\ f\ g \equiv (S \unlhd_h R)\ g\ f$
⟨*proof*⟩

lemma *half-galois-prop-right-Refl-Rel-right-leftI*:
assumes $((\leq_R) \Rrightarrow_m (\leq_L))\ r$
and $((\leq_L) \unlhd_h (\leq_R))\ l\ r$
shows $((\leq_L)^\oplus \unlhd_h (\leq_R)^\oplus)\ l\ r$
⟨*proof*⟩

corollary *galois-prop-Refl-Rel-left-rightI*:
assumes $((\leq_L) \dashv (\leq_R))\ l\ r$
shows $((\leq_L)^\oplus \unlhd (\leq_R)^\oplus)\ l\ r$
⟨*proof*⟩

lemma *galois-connection-Refl-Rel-left-rightI*:
assumes $((\leq_L) \dashv (\leq_R))\ l\ r$
shows $((\leq_L)^\oplus \dashv (\leq_R)^\oplus)\ l\ r$
⟨*proof*⟩

lemma *galois-equivalence-Refl-RelI*:
assumes $((\leq_L) \equiv_G (\leq_R))\ l\ r$
shows $((\leq_L)^\oplus \equiv_G (\leq_R)^\oplus)\ l\ r$
⟨*proof*⟩

end

context *order-functors*
begin

lemma *inflationary-on-in-field-Refl-Rel-left*:
assumes $((\leq_L) \Rrightarrow_m (\leq_R))\ l$
and $((\leq_R) \Rrightarrow_m (\leq_L))\ r$

and *inflationary-on* (*in-dom* $(\leq_L)$) $(\leq_L)$ η
shows *inflationary-on* (*in-field* $(\leq_L)^{\oplus}$) $(\leq_L)^{\oplus}$ η
⟨*proof*⟩

lemma *inflationary-on-in-field-Refl-Rel-left'*:
assumes $((\leq_L) \Rrightarrow_m (\leq_R))$ l
and $((\leq_R) \Rrightarrow_m (\leq_L))$ r
and *inflationary-on* (*in-codom* $(\leq_L)$) $(\leq_L)$ η
shows *inflationary-on* (*in-field* $(\leq_L)^{\oplus}$) $(\leq_L)^{\oplus}$ η
⟨*proof*⟩

interpretation *inv* : *galois* $(\geq_L)$ $(\geq_R)$ l r
rewrites $((\geq_L) \Rrightarrow_m (\geq_R)) \equiv ((\leq_L) \Rrightarrow_m (\leq_R))$
and $((\geq_R) \Rrightarrow_m (\geq_L)) \equiv ((\leq_R) \Rrightarrow_m (\leq_L))$
and $\bigwedge R.\ (R^{-1})^{\oplus} \equiv (R^{\oplus})^{-1}$
and $\bigwedge R.$ *in-dom* $R^{-1} \equiv$ *in-codom* R
and $\bigwedge R.$ *in-codom* $R^{-1} \equiv$ *in-dom* R
and $\bigwedge R.$ *in-field* $R^{-1} \equiv$ *in-field* R
and $\bigwedge P\ R.$ *inflationary-on* P $R^{-1} \equiv$ *deflationary-on* P R
⟨*proof*⟩

lemma *deflationary-on-in-field-Refl-Rel-leftI*:
assumes $((\leq_L) \Rrightarrow_m (\leq_R))$ l
and $((\leq_R) \Rrightarrow_m (\leq_L))$ r
and *deflationary-on* (*in-dom* $(\leq_L)$) $(\leq_L)$ η
shows *deflationary-on* (*in-field* $(\leq_L)^{\oplus}$) $(\leq_L)^{\oplus}$ η
⟨*proof*⟩

lemma *deflationary-on-in-field-Refl-RelI-left'*:
assumes $((\leq_L) \Rrightarrow_m (\leq_R))$ l
and $((\leq_R) \Rrightarrow_m (\leq_L))$ r
and *deflationary-on* (*in-codom* $(\leq_L)$) $(\leq_L)$ η
shows *deflationary-on* (*in-field* $(\leq_L)^{\oplus}$) $(\leq_L)^{\oplus}$ η
⟨*proof*⟩

lemma *rel-equivalence-on-in-field-Refl-Rel-leftI*:
assumes $((\leq_L) \Rrightarrow_m (\leq_R))$ l
and $((\leq_R) \Rrightarrow_m (\leq_L))$ r
and *rel-equivalence-on* (*in-dom* $(\leq_L)$) $(\leq_L)$ η
shows *rel-equivalence-on* (*in-field* $(\leq_L)^{\oplus}$) $(\leq_L)^{\oplus}$ η
⟨*proof*⟩

lemma *rel-equivalence-on-in-field-Refl-Rel-leftI'*:
assumes $((\leq_L) \Rrightarrow_m (\leq_R))$ l
and $((\leq_R) \Rrightarrow_m (\leq_L))$ r
and *rel-equivalence-on* (*in-codom* $(\leq_L)$) $(\leq_L)$ η
shows *rel-equivalence-on* (*in-field* $(\leq_L)^{\oplus}$) $(\leq_L)^{\oplus}$ η
⟨*proof*⟩

interpretation *oR* : *order-functors* $(\le_L)^{\oplus}$ $(\le_R)^{\oplus}$ *l r* ⟨*proof*⟩

lemma *order-equivalence-Refl-RelI*:
assumes $((\le_L) \equiv_o (\le_R))$ *l r*
shows $((\le_L)^{\oplus} \equiv_o (\le_R)^{\oplus})$ *l r*
⟨*proof*⟩

end

end

2.7 Monotone Function Relator

theory *Monotone-Function-Relator*
imports
Reflexive-Relator
begin

abbreviation *Mono-Dep-Fun-Rel R S* ≡ ([*x y* :: *R*] ⇛ *S x y*)$^{\oplus}$
abbreviation *Mono-Fun-Rel R S* ≡ *Mono-Dep-Fun-Rel R* (λ- -. *S*)

bundle *Mono-Dep-Fun-Rel-syntax* **begin**
syntax
-Mono-Fun-Rel-rel :: (′*a* ⇒ ′*b* ⇒ *bool*) ⇒ (′*c* ⇒ ′*d* ⇒ *bool*) ⇒ (′*a* ⇒ ′*c*) ⇒
(′*b* ⇒ ′*d*) ⇒ *bool* ((-) ⇛⊕ (-) [*41*, *40*] *40*)
-Mono-Dep-Fun-Rel-rel :: *idt* ⇒ *idt* ⇒ (′*a* ⇒ ′*b* ⇒ *bool*) ⇒ (′*c* ⇒ ′*d* ⇒ *bool*) ⇒
(′*a* ⇒ ′*c*) ⇒ (′*b* ⇒ ′*d*) ⇒ *bool* ([-/ -/ ::/ -] ⇛⊕ (-) [*41*, *41*, *41*, *40*] *40*)
-Mono-Dep-Fun-Rel-rel-if :: *idt* ⇒ *idt* ⇒ (′*a* ⇒ ′*b* ⇒ *bool*) ⇒ *bool* ⇒ (′*c* ⇒ ′*d*
⇒ *bool*) ⇒
(′*a* ⇒ ′*c*) ⇒ (′*b* ⇒ ′*d*) ⇒ *bool* ([-/ -/ ::/ -/ |/ -] ⇛⊕ (-) [*41*, *41*, *41*, *41*, *40*]
40)
end
bundle *no-Mono-Dep-Fun-Rel-syntax* **begin**
no-syntax
-Mono-Fun-Rel-rel :: (′*a* ⇒ ′*b* ⇒ *bool*) ⇒ (′*c* ⇒ ′*d* ⇒ *bool*) ⇒ (′*a* ⇒ ′*c*) ⇒
(′*b* ⇒ ′*d*) ⇒ *bool* ((-) ⇛⊕ (-) [*41*, *40*] *40*)
-Mono-Dep-Fun-Rel-rel :: *idt* ⇒ *idt* ⇒ (′*a* ⇒ ′*b* ⇒ *bool*) ⇒ (′*c* ⇒ ′*d* ⇒ *bool*) ⇒
(′*a* ⇒ ′*c*) ⇒ (′*b* ⇒ ′*d*) ⇒ *bool* ([-/ -/ ::/ -] ⇛⊕ (-) [*41*, *41*, *41*, *40*] *40*)
-Mono-Dep-Fun-Rel-rel-if :: *idt* ⇒ *idt* ⇒ (′*a* ⇒ ′*b* ⇒ *bool*) ⇒ *bool* ⇒ (′*c* ⇒ ′*d*
⇒ *bool*) ⇒
(′*a* ⇒ ′*c*) ⇒ (′*b* ⇒ ′*d*) ⇒ *bool* ([-/ -/ ::/ -/ |/ -] ⇛⊕ (-) [*41*, *41*, *41*, *41*, *40*]
40)
end
unbundle *Mono-Dep-Fun-Rel-syntax*

translations
R ⇛⊕ *S* ⇌ *CONST Mono-Fun-Rel R S*
[*x y* :: *R*] ⇛⊕ *S* ⇌ *CONST Mono-Dep-Fun-Rel R* (λ*x y*. *S*)

$[x\ y :: R \mid B] \Rrightarrow\oplus S \rightleftharpoons$ *CONST Mono-Dep-Fun-Rel R* ($\lambda x\ y$. *CONST rel-if B S*)

locale *Dep-Fun-Rel-orders* =
  fixes $L :: 'a \Rightarrow 'b \Rightarrow bool$
  and $R :: 'a \Rightarrow 'b \Rightarrow 'c \Rightarrow 'd \Rightarrow bool$
begin

sublocale *o* : *orders L R a b* **for** *a b* ⟨*proof*⟩

notation *L* (**infix** $\le_L$ *50*)
notation *o.ge-left* (**infix** $\ge_L$ *50*)

notation *R* (($\le_{R\ (-)\ (-)}$) *50*)
abbreviation *right-infix c a b d* $\equiv$ ($\le_{R\ a\ b}$) *c d*
notation *right-infix* ((-) $\le_{R\ (-)\ (-)}$ (-) [*51*,*51*,*51*,*51*] *50*)

notation *o.ge-right* (($\ge_{R\ (-)\ (-)}$) *50*)

abbreviation (*input*) *ge-right-infix d a b c* $\equiv$ ($\ge_{R\ a\ b}$) *d c*
notation *ge-right-infix* ((-) $\ge_{R\ (-)\ (-)}$ (-) [*51*,*51*,*51*,*51*] *50*)

abbreviation (*input*) *DFR* $\equiv$ ([*a b* :: *L*] $\Rrightarrow$ *R a b*)

end

locale *hom-Dep-Fun-Rel-orders* = *Dep-Fun-Rel-orders L R*
  for $L :: 'a \Rightarrow 'a \Rightarrow bool$
  and $R :: 'a \Rightarrow 'a \Rightarrow 'b \Rightarrow 'b \Rightarrow bool$
begin

sublocale *ho* : *hom-orders L R a b* **for** *a b* ⟨*proof*⟩

lemma *Mono-Dep-Fun-Refl-Rel-right-eq-Mono-Dep-Fun-if-le-if-reflexive-onI*:
  assumes *refl-L*: *reflexive-on* (*in-field* ($\le_L$)) ($\le_L$)
  and $\bigwedge x1\ x2.\ x1 \le_L x2 \Longrightarrow (\le_{R\ x2\ x2}) \le (\le_{R\ x1\ x2})$
  and $\bigwedge x1\ x2.\ x1 \le_L x2 \Longrightarrow (\le_{R\ x1\ x1}) \le (\le_{R\ x1\ x2})$
  shows $([x\ y :: (\le_L)] \Rrightarrow\oplus (\le_{R\ x\ y})^{\oplus}) = ([x\ y :: (\le_L)] \Rrightarrow\oplus (\le_{R\ x\ y}))$
⟨*proof*⟩

lemma *Mono-Dep-Fun-Refl-Rel-right-eq-Mono-Dep-Fun-if-mono-if-reflexive-onI*:
  assumes *reflexive-on* (*in-field* ($\le_L$)) ($\le_L$)
  and $([x1\ x2 :: (\ge_L)] \Rrightarrow_m [x3\ x4 :: (\le_L) \mid x1 \le_L x3] \Rrightarrow (\le))\ R$
  shows $([x\ y :: (\le_L)] \Rrightarrow\oplus (\le_{R\ x\ y})^{\oplus}) = ([x\ y :: (\le_L)] \Rrightarrow\oplus (\le_{R\ x\ y}))$
  ⟨*proof*⟩

end

context *hom-orders*
begin

sublocale *fro* : *hom-Dep-Fun-Rel-orders L λ- -. R* ⟨*proof*⟩

corollary *Mono-Fun-Rel-Refl-Rel-right-eq-Mono-Fun-RelI*:
assumes *reflexive-on* (*in-field* ($\le_L$)) ($\le_L$)
shows (($\le_L$) ⇛⊕ ($\le_R$)$^{\oplus}$) = (($\le_L$) ⇛⊕ ($\le_R$))
⟨*proof*⟩

end

end

2.8 Transport For Functions

2.8.1 Basic Setup

theory *Transport-Functions-Base*
imports
Monotone-Function-Relator
Transport-Base
begin

Summary Basic setup for closure proofs. We introduce locales for the syntax, the dependent relator, the non-dependent relator, the monotone dependent relator, and the monotone non-dependent relator.

definition *flip2 f x1 x2 x3 x4* ≡ *f x2 x1 x4 x3*

lemma *flip2-eq*: *flip2 f x1 x2 x3 x4* = *f x2 x1 x4 x3*
⟨*proof*⟩

lemma *flip2-eq-rel-inv* [*simp*]: *flip2 R x y* = $(R\ y\ x)^{-1}$
⟨*proof*⟩

lemma *flip2-flip2-eq-self* [*simp*]: *flip2* (*flip2 f*) = *f*
⟨*proof*⟩

lemma *flip2-eq-flip2-iff-eq* [*iff*]: *flip2 f* = *flip2 g* ⟷ *f* = *g*
⟨*proof*⟩

Dependent Function Relator **locale** *transport-Dep-Fun-Rel-syntax* =
t1 : *transport L1 R1 l1 r1* +
dfro1 : *hom-Dep-Fun-Rel-orders L1 L2* +
dfro2 : *hom-Dep-Fun-Rel-orders R1 R2*
for *L1* :: *'a1* ⇒ *'a1* ⇒ *bool*
and *R1* :: *'a2* ⇒ *'a2* ⇒ *bool*
and *l1* :: *'a1* ⇒ *'a2*
and *r1* :: *'a2* ⇒ *'a1*

and $L2$ :: $'a1 \Rightarrow 'a1 \Rightarrow 'b1 \Rightarrow 'b1 \Rightarrow bool$
and $R2$ :: $'a2 \Rightarrow 'a2 \Rightarrow 'b2 \Rightarrow 'b2 \Rightarrow bool$
and $l2$ :: $'a2 \Rightarrow 'a1 \Rightarrow 'b1 \Rightarrow 'b2$
and $r2$ :: $'a1 \Rightarrow 'a2 \Rightarrow 'b2 \Rightarrow 'b1$
begin

notation *L1* (**infix** $\le_{L1}$ *50*)
notation *R1* (**infix** $\le_{R1}$ *50*)

notation *t1.ge-left* (**infix** $\ge_{L1}$ *50*)
notation *t1.ge-right* (**infix** $\ge_{R1}$ *50*)

notation *t1.left-Galois* (**infix** $_{L1}\lessapprox$ *50*)
notation *t1.ge-Galois-left* (**infix** $\gtrapprox_{L1}$ *50*)
notation *t1.right-Galois* (**infix** $_{R1}\lessapprox$ *50*)
notation *t1.ge-Galois-right* (**infix** $\gtrapprox_{R1}$ *50*)
notation *t1.right-ge-Galois* (**infix** $_{R1}\gtrapprox$ *50*)
notation *t1.Galois-right* (**infix** $\lessapprox_{R1}$ *50*)
notation *t1.left-ge-Galois* (**infix** $_{L1}\gtrapprox$ *50*)
notation *t1.Galois-left* (**infix** $\lessapprox_{L1}$ *50*)

notation *t1.unit* (η_1)
notation *t1.counit* (ε_1)

notation *L2* (($\le_{L2\ (-)\ (-)}$) *50*)
notation *R2* (($\le_{R2\ (-)\ (-)}$) *50*)

notation *dfro1.right-infix* ((-) $\le_{L2\ (-)\ (-)}$ (-) [*51*,*51*,*51*,*51*] *50*)
notation *dfro2.right-infix* ((-) $\le_{R2\ (-)\ (-)}$ (-) [*51*,*51*,*51*,*51*] *50*)

notation *dfro1.o.ge-right* (($\ge_{L2\ (-)\ (-)}$) *50*)
notation *dfro2.o.ge-right* (($\ge_{R2\ (-)\ (-)}$) *50*)

notation *dfro1.ge-right-infix* ((-) $\ge_{L2\ (-)\ (-)}$ (-) [*51*,*51*,*51*,*51*] *50*)
notation *dfro2.ge-right-infix* ((-) $\ge_{R2\ (-)\ (-)}$ (-) [*51*,*51*,*51*,*51*] *50*)

notation *l2* ($l2_{(-)\ (-)}$)
notation *r2* ($r2_{(-)\ (-)}$)

sublocale *t2* : *transport* ($\le_{L2\ x\ (r1\ x')}$) ($\le_{R2\ (l1\ x)\ x'}$) $l2_{x'\ x}$ $r2_{x\ x'}$ **for** x x'
⟨*proof*⟩

notation *t2.left-Galois* (($_{L2\ (-)\ (-)}\lessapprox$) *50*)
notation *t2.right-Galois* (($_{R2\ (-)\ (-)}\lessapprox$) *50*)

abbreviation *left2-Galois-infix* $y\ x\ x'\ y' \equiv (_{L2\ x\ x'}\lessapprox)\ y\ y'$
notation *left2-Galois-infix* ((-) $_{L2\ (-)\ (-)}\lessapprox$ (-) [*51*,*51*,*51*,*51*] *50*)
abbreviation *right2-Galois-infix* $y'\ x\ x'\ y \equiv (_{R2\ x\ x'}\lessapprox)\ y'\ y$

notation *right2-Galois-infix* ((-) $_{R2\ (-)\ (-)}\lessapprox$ (-) [*51*,*51*,*51*,*51*] *50*)

notation *t2.ge-Galois-left* (($\gtrapprox_{L2\ (-)\ (-)}$) *50*)
notation *t2.ge-Galois-right* (($\gtrapprox_{R2\ (-)\ (-)}$) *50*)

abbreviation (*input*) *ge-Galois-left-left2-infix* y' x x' y ≡ ($\gtrapprox_{L2\ x\ x'}$) y' y
notation *ge-Galois-left-left2-infix* ((-) $\gtrapprox_{L2\ (-)\ (-)}$ (-) [*51*,*51*,*51*,*51*] *50*)
abbreviation (*input*) *ge-Galois-left-right2-infix* y x x' y' ≡ ($\gtrapprox_{R2\ x\ x'}$) y y'
notation *ge-Galois-left-right2-infix* ((-) $\gtrapprox_{R2\ (-)\ (-)}$ (-) [*51*,*51*,*51*,*51*] *50*)

notation *t2.right-ge-Galois* (($_{R2\ (-)\ (-)}\gtrapprox$) *50*)
notation *t2.left-ge-Galois* (($_{L2\ (-)\ (-)}\gtrapprox$) *50*)

abbreviation *left2-ge-Galois-left-infix* y x x' y' ≡ ($_{L2\ x\ x'}\gtrapprox$) y y'
notation *left2-ge-Galois-left-infix* ((-) $_{L2\ (-)\ (-)}\gtrapprox$ (-) [*51*,*51*,*51*,*51*] *50*)
abbreviation *right2-ge-Galois-left-infix* y' x x' y ≡ ($_{R2\ x\ x'}\gtrapprox$) y' y
notation *right2-ge-Galois-left-infix* ((-) $_{R2\ (-)\ (-)}\gtrapprox$ (-) [*51*,*51*,*51*,*51*] *50*)

notation *t2.Galois-right* (($\lessapprox_{R2\ (-)\ (-)}$) *50*)
notation *t2.Galois-left* (($\lessapprox_{L2\ (-)\ (-)}$) *50*)

abbreviation (*input*) *Galois-left2-infix* y' x x' y ≡ ($\lessapprox_{L2\ x\ x'}$) y' y
notation *Galois-left2-infix* ((-) $\lessapprox_{L2\ (-)\ (-)}$ (-) [*51*,*51*,*51*,*51*] *50*)
abbreviation (*input*) *Galois-right2-infix* y x x' y' ≡ ($\lessapprox_{R2\ x\ x'}$) y y'
notation *Galois-right2-infix* ((-) $\lessapprox_{R2\ (-)\ (-)}$ (-) [*51*,*51*,*51*,*51*] *50*)

abbreviation *t2-unit* x x' ≡ *t2.unit* x' x
notation *t2-unit* ($\eta_{2\ (-)\ (-)}$)
abbreviation *t2-counit* x x' ≡ *t2.counit* x' x
notation *t2-counit* ($\varepsilon_{2\ (-)\ (-)}$)

end

locale *transport-Dep-Fun-Rel* =
  transport-Dep-Fun-Rel-syntax L1 R1 l1 r1 L2 R2 l2 r2
  for *L1* :: *'a1* ⇒ *'a1* ⇒ *bool*
  and *R1* :: *'a2* ⇒ *'a2* ⇒ *bool*
  and *l1* :: *'a1* ⇒ *'a2*
  and *r1* :: *'a2* ⇒ *'a1*
  and *L2* :: *'a1* ⇒ *'a1* ⇒ *'b1* ⇒ *'b1* ⇒ *bool*
  and *R2* :: *'a2* ⇒ *'a2* ⇒ *'b2* ⇒ *'b2* ⇒ *bool*
  and *l2* :: *'a2* ⇒ *'a1* ⇒ *'b1* ⇒ *'b2*
  and *r2* :: *'a1* ⇒ *'a2* ⇒ *'b2* ⇒ *'b1*
begin

definition $L \equiv [x1\ x2 :: (\le_{L1})] \Rrightarrow (\le_{L2\ x1\ x2})$

lemma *left-rel-eq-Dep-Fun-Rel*: $L = ([x1\ x2 :: (\le_{L1})] \Rrightarrow (\le_{L2\ x1\ x2}))$

⟨*proof*⟩

definition $l \equiv ([x' : r1] \rightarrow l2\ x')$

lemma *left-eq-dep-fun-map*: $l = ([x' : r1] \rightarrow l2\ x')$
⟨*proof*⟩

lemma *left-eq* [*simp*]: $l\ f\ x' = l2_{x'\ (r1\ x')}\ (f\ (r1\ x'))$
⟨*proof*⟩

context
begin

interpretation *flip* : *transport-Dep-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2* ⟨*proof*⟩

abbreviation $R \equiv flip.L$
abbreviation $r \equiv flip.l$

lemma *right-rel-eq-Dep-Fun-Rel*: $R = ([x1'\ x2' :: (\le_{R1})] \Rrightarrow (\le_{R2\ x1'\ x2'}))$
⟨*proof*⟩

lemma *right-eq-dep-fun-map*: $r = ([x : l1] \rightarrow r2\ x)$
⟨*proof*⟩

end

lemma *right-eq* [*simp*]: $r\ g\ x = r2_{x\ (l1\ x)}\ (g\ (l1\ x))$
⟨*proof*⟩

lemmas *transport-defs* = *left-rel-eq-Dep-Fun-Rel left-eq-dep-fun-map right-rel-eq-Dep-Fun-Rel right-eq-dep-fun-map*

sublocale *transport L R l r* ⟨*proof*⟩

notation L (**infix** $\le_L$ *50*)
notation R (**infix** $\le_R$ *50*)

lemma *left-relI* [*intro*]:
assumes $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow f\ x1 \le_{L2\ x1\ x2} f'\ x2$
shows $f \le_L f'$
⟨*proof*⟩

lemma *left-relE* [*elim*]:
assumes $f \le_L f'$
and $x1 \le_{L1} x2$
obtains $f\ x1 \le_{L2\ x1\ x2} f'\ x2$
⟨*proof*⟩

interpretation *flip-inv* :
transport-Dep-Fun-Rel $(\geq_{R1})$ $(\geq_{L1})$ *r1 l1 flip2 R2 flip2 L2 r2 l2* ⟨*proof*⟩

lemma *flip-inv-right-eq-ge-left*: *flip-inv.R* = $(\geq_L)$
⟨*proof*⟩

interpretation *flip* : *transport-Dep-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2* ⟨*proof*⟩

lemma *flip-inv-left-eq-ge-right*: *flip-inv.L* ≡ $(\geq_R)$
⟨*proof*⟩

Useful Rewritings for Dependent Relation **lemma** *left-rel2-unit-eqs-left-rel2I*:
assumes $\bigwedge x1\ x2.\ x1 \leq_{L1} x2 \Longrightarrow (\leq_{L2\ x2\ x2}) \leq (\leq_{L2\ x1\ x2})$
and $\bigwedge x.\ x \leq_{L1} x \Longrightarrow (\leq_{L2\ (\eta_1\ x)\ x}) \leq (\leq_{L2\ x\ x})$
and $\bigwedge x1\ x2.\ x1 \leq_{L1} x2 \Longrightarrow (\leq_{L2\ x1\ x1}) \leq (\leq_{L2\ x1\ x2})$
and $\bigwedge x.\ x \leq_{L1} x \Longrightarrow (\leq_{L2\ x\ (\eta_1\ x)}) \leq (\leq_{L2\ x\ x})$
and $x \leq_{L1} x$
and $x \equiv_{L1} \eta_1\ x$
shows $(\leq_{L2\ (\eta_1\ x)\ x}) = (\leq_{L2\ x\ x})$
and $(\leq_{L2\ x\ (\eta_1\ x)}) = (\leq_{L2\ x\ x})$
⟨*proof*⟩

lemma *left2-eq-if-bi-related-if-monoI*:
assumes *mono-L2*: $([x1\ x2 :: (\geq_{L1})] \Rrightarrow_m [x3\ x4 :: (\leq_{L1}) \mid x1 \leq_{L1} x3] \Rrightarrow (\leq))$
L2
and $x1 \leq_{L1} x2$
and $x1 \equiv_{L1} x3$
and $x2 \equiv_{L1} x4$
and *trans-L1*: *transitive* $(\leq_{L1})$
shows $(\leq_{L2\ x1\ x2}) = (\leq_{L2\ x3\ x4})$
⟨*proof*⟩

end

Function Relator **locale** *transport-Fun-Rel-syntax* =
tdfrs : *transport-Dep-Fun-Rel-syntax L1 R1 l1 r1* λ- -. *L2* λ- -. *R2*
λ- -. *l2* λ- -. *r2*
for *L1* :: *'a1* ⇒ *'a1* ⇒ *bool*
and *R1* :: *'a2* ⇒ *'a2* ⇒ *bool*
and *l1* :: *'a1* ⇒ *'a2*
and *r1* :: *'a2* ⇒ *'a1*
and *L2* :: *'b1* ⇒ *'b1* ⇒ *bool*
and *R2* :: *'b2* ⇒ *'b2* ⇒ *bool*
and *l2* :: *'b1* ⇒ *'b2*
and *r2* :: *'b2* ⇒ *'b1*
begin

notation *L1* (**infix** $\leq_{L1}$ *50*)

notation *R1* (**infix** $\le_{R1}$ *50*)

notation *tdfrs.t1.ge-left* (**infix** $\ge_{L1}$ *50*)
notation *tdfrs.t1.ge-right* (**infix** $\ge_{R1}$ *50*)

notation *tdfrs.t1.left-Galois* (**infix** ${}_{L1}\lessapprox$ *50*)
notation *tdfrs.t1.ge-Galois-left* (**infix** $\gtrapprox_{L1}$ *50*)
notation *tdfrs.t1.right-Galois* (**infix** ${}_{R1}\lessapprox$ *50*)
notation *tdfrs.t1.ge-Galois-right* (**infix** $\gtrapprox_{R1}$ *50*)
notation *tdfrs.t1.right-ge-Galois* (**infix** ${}_{R1}\gtrapprox$ *50*)
notation *tdfrs.t1.Galois-right* (**infix** $\lessapprox_{R1}$ *50*)
notation *tdfrs.t1.left-ge-Galois* (**infix** ${}_{L1}\gtrapprox$ *50*)
notation *tdfrs.t1.Galois-left* (**infix** $\lessapprox_{L1}$ *50*)

notation *tdfrs.t1.unit* (η_1)
notation *tdfrs.t1.counit* (ε_1)

notation *L2* (**infix** $\le_{L2}$ *50*)
notation *R2* (**infix** $\le_{R2}$ *50*)

notation *tdfrs.t2.ge-left* (**infix** $\ge_{L2}$ *50*)
notation *tdfrs.t2.ge-right* (**infix** $\ge_{R2}$ *50*)

notation *tdfrs.t2.left-Galois* (**infix** ${}_{L2}\lessapprox$ *50*)
notation *tdfrs.t2.ge-Galois-left* (**infix** $\gtrapprox_{L2}$ *50*)
notation *tdfrs.t2.right-Galois* (**infix** ${}_{R2}\lessapprox$ *50*)
notation *tdfrs.t2.ge-Galois-right* (**infix** $\gtrapprox_{R2}$ *50*)
notation *tdfrs.t2.right-ge-Galois* (**infix** ${}_{R2}\gtrapprox$ *50*)
notation *tdfrs.t2.Galois-right* (**infix** $\lessapprox_{R2}$ *50*)
notation *tdfrs.t2.left-ge-Galois* (**infix** ${}_{L2}\gtrapprox$ *50*)
notation *tdfrs.t2.Galois-left* (**infix** $\lessapprox_{L2}$ *50*)

notation *tdfrs.t2.unit* (η_2)
notation *tdfrs.t2.counit* (ε_2)

end

locale *transport-Fun-Rel* =
  transport-Fun-Rel-syntax L1 R1 l1 r1 L2 R2 l2 r2 +
  tdfr : *transport-Dep-Fun-Rel L1 R1 l1 r1* λ- -. *L2* λ- -. *R2*
    λ- -. *l2* λ- -. *r2*
  for *L1* :: *'a1* ⇒ *'a1* ⇒ *bool*
  and *R1* :: *'a2* ⇒ *'a2* ⇒ *bool*
  and *l1* :: *'a1* ⇒ *'a2*
  and *r1* :: *'a2* ⇒ *'a1*
  and *L2* :: *'b1* ⇒ *'b1* ⇒ *bool*
  and *R2* :: *'b2* ⇒ *'b2* ⇒ *bool*
  and *l2* :: *'b1* ⇒ *'b2*
  and *r2* :: *'b2* ⇒ *'b1*

begin

notation *tdfr.L* (*L*)
notation *tdfr.R* (*R*)

abbreviation $l \equiv$ *tdfr.l*
abbreviation $r \equiv$ *tdfr.r*

notation *tdfr.L* (**infix** $\leq_L$ *50*)
notation *tdfr.R* (**infix** $\leq_R$ *50*)

notation *tdfr.ge-left* (**infix** $\geq_L$ *50*)
notation *tdfr.ge-right* (**infix** $\geq_R$ *50*)

notation *tdfr.left-Galois* (**infix** ${}_L\lessapprox$ *50*)
notation *tdfr.ge-Galois-left* (**infix** $\gtrapprox_L$ *50*)
notation *tdfr.right-Galois* (**infix** ${}_R\lessapprox$ *50*)
notation *tdfr.ge-Galois-right* (**infix** $\gtrapprox_R$ *50*)
notation *tdfr.right-ge-Galois* (**infix** ${}_R\gtrapprox$ *50*)
notation *tdfr.Galois-right* (**infix** $\lessapprox_R$ *50*)
notation *tdfr.left-ge-Galois* (**infix** ${}_L\gtrapprox$ *50*)
notation *tdfr.Galois-left* (**infix** $\lessapprox_L$ *50*)

notation *tdfr.unit* (η)
notation *tdfr.counit* (ε)

lemma *left-rel-eq-Fun-Rel*: $(\leq_L) = ((\leq_{L1}) \Rrightarrow (\leq_{L2}))$
⟨*proof*⟩

lemma *left-eq-fun-map*: $l = (r1 \rightarrow l2)$
⟨*proof*⟩

interpretation *flip* : *transport-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2* ⟨*proof*⟩

lemma *right-rel-eq-Fun-Rel*: $(\leq_R) = ((\leq_{R1}) \Rrightarrow (\leq_{R2}))$
⟨*proof*⟩

lemma *right-eq-fun-map*: $r = (l1 \rightarrow r2)$
⟨*proof*⟩

lemmas *transport-defs* = *left-rel-eq-Fun-Rel right-rel-eq-Fun-Rel left-eq-fun-map right-eq-fun-map*

end

Monotone Dependent Function Relator **locale** *transport-Mono-Dep-Fun-Rel* =
transport-Dep-Fun-Rel-syntax L1 R1 l1 r1 L2 R2 l2 r2

+ *tdfr* : *transport-Dep-Fun-Rel L1 R1 l1 r1 L2 R2 l2 r2*
for *L1* :: *'a1* ⇒ *'a1* ⇒ *bool*
and *R1* :: *'a2* ⇒ *'a2* ⇒ *bool*
and *l1* :: *'a1* ⇒ *'a2*
and *r1* :: *'a2* ⇒ *'a1*
and *L2* :: *'a1* ⇒ *'a1* ⇒ *'b1* ⇒ *'b1* ⇒ *bool*
and *R2* :: *'a2* ⇒ *'a2* ⇒ *'b2* ⇒ *'b2* ⇒ *bool*
and *l2* :: *'a2* ⇒ *'a1* ⇒ *'b1* ⇒ *'b2*
and *r2* :: *'a1* ⇒ *'a2* ⇒ *'b2* ⇒ *'b1*
begin

definition $L \equiv tdfr.L^{\oplus}$

lemma *left-rel-eq-tdfr-left-Refl-Rel*: $L = tdfr.L^{\oplus}$
⟨*proof*⟩

lemma *left-rel-eq-Mono-Dep-Fun-Rel*: $L = ([x1\ x2 :: (\leq_{L1})] \Rrightarrow\oplus (\leq_{L2\ x1\ x2}))$
⟨*proof*⟩

lemma *left-rel-eq-tdfr-left-rel-if-reflexive-on*:
assumes *reflexive-on* (*in-field tdfr.L*) *tdfr.L*
shows $L = tdfr.L$
⟨*proof*⟩

abbreviation $l \equiv tdfr.l$

lemma *left-eq-tdfr-left*: $l = tdfr.l$ ⟨*proof*⟩

interpretation *flip* : *transport-Mono-Dep-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2* ⟨*proof*⟩

abbreviation $R \equiv flip.L$

lemma *right-rel-eq-tdfr-right-Refl-Rel*: $R = tdfr.R^{\oplus}$
⟨*proof*⟩

lemma *right-rel-eq-Mono-Dep-Fun-Rel*: $R = ([y1\ y2 :: (\leq_{R1})] \Rrightarrow\oplus (\leq_{R2\ y1\ y2}))$
⟨*proof*⟩

lemma *right-rel-eq-tdfr-right-rel-if-reflexive-on*:
assumes *reflexive-on* (*in-field tdfr.R*) *tdfr.R*
shows $R = tdfr.R$
⟨*proof*⟩

abbreviation $r \equiv tdfr.r$

lemma *right-eq-tdfr-right*: $r = tdfr.r$ ⟨*proof*⟩

lemmas *transport-defs* = *left-rel-eq-tdfr-left-Refl-Rel*
right-rel-eq-tdfr-right-Refl-Rel

sublocale *transport L R l r* ⟨*proof*⟩

notation *L* (**infix** $\leq_L$ *50*)
notation *R* (**infix** $\leq_R$ *50*)

end

Monotone Function Relator **locale** *transport-Mono-Fun-Rel* =
transport-Fun-Rel-syntax L1 R1 l1 r1 L2 R2 l2 r2 +
tfr : *transport-Fun-Rel L1 R1 l1 r1 L2 R2 l2 r2* +
tpdfr : *transport-Mono-Dep-Fun-Rel L1 R1 l1 r1* λ- -. *L2* λ- -. *R2*
λ- -. *l2* λ- -. *r2*
for *L1* :: *'a1* ⇒ *'a1* ⇒ *bool*
and *R1* :: *'a2* ⇒ *'a2* ⇒ *bool*
and *l1* :: *'a1* ⇒ *'a2*
and *r1* :: *'a2* ⇒ *'a1*
and *L2* :: *'b1* ⇒ *'b1* ⇒ *bool*
and *R2* :: *'b2* ⇒ *'b2* ⇒ *bool*
and *l2* :: *'b1* ⇒ *'b2*
and *r2* :: *'b2* ⇒ *'b1*
begin

notation *tpdfr.L* (*L*)
notation *tpdfr.R* (*R*)

abbreviation *l* ≡ *tpdfr.l*
abbreviation *r* ≡ *tpdfr.r*

notation *tpdfr.L* (**infix** $\leq_L$ *50*)
notation *tpdfr.R* (**infix** $\leq_R$ *50*)

notation *tpdfr.ge-left* (**infix** $\geq_L$ *50*)
notation *tpdfr.ge-right* (**infix** $\geq_R$ *50*)

notation *tpdfr.left-Galois* (**infix** ${}_L\lessapprox$ *50*)
notation *tpdfr.ge-Galois-left* (**infix** $\gtrapprox_L$ *50*)
notation *tpdfr.right-Galois* (**infix** ${}_R\lessapprox$ *50*)
notation *tpdfr.ge-Galois-right* (**infix** $\gtrapprox_R$ *50*)
notation *tpdfr.right-ge-Galois* (**infix** ${}_R\gtrapprox$ *50*)
notation *tpdfr.Galois-right* (**infix** $\lessapprox_R$ *50*)
notation *tpdfr.left-ge-Galois* (**infix** ${}_L\gtrapprox$ *50*)
notation *tpdfr.Galois-left* (**infix** $\lessapprox_L$ *50*)

notation *tpdfr.unit* (η)
notation *tpdfr.counit* (ε)

lemma *left-rel-eq-Mono-Fun-Rel*: $(\le_L) = ((\le_{L1}) \Rrightarrow\oplus (\le_{L2}))$
⟨*proof*⟩

lemma *left-eq-fun-map*: $l = (r1 \to l2)$
⟨*proof*⟩

interpretation *flip* : *transport-Mono-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2* ⟨*proof*⟩

lemma *right-rel-eq-Mono-Fun-Rel*: $(\le_R) = ((\le_{R1}) \Rrightarrow\oplus (\le_{R2}))$
⟨*proof*⟩

lemma *right-eq-fun-map*: $r = (l1 \to r2)$
⟨*proof*⟩

lemmas *transport-defs* = *tpdfr.transport-defs*

end

end

2.8.2 Monotonicity

theory *Transport-Functions-Monotone*
imports
Transport-Functions-Base
begin

Dependent Function Relator **context** *transport-Dep-Fun-Rel*
begin

interpretation *flip* : *transport-Dep-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2* ⟨*proof*⟩

lemma *mono-wrt-rel-leftI*:
assumes *mono-r1*: $((\le_{R1}) \Rrightarrow_m (\le_{L1}))\ r1$
and *mono-l2*: $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$
$((\le_{L2\ (r1\ x1')\ (r1\ x2')}) \Rrightarrow_m (\le_{R2\ (\varepsilon_1\ x1')\ x2'}))\ (l2_{x2'\ (r1\ x1')})$
and *R2-le1*: $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x1')\ x2'}) \le (\le_{R2\ x1'\ x2'})$
and *R2-l2-le1*: $\bigwedge x1'\ x2'\ y.\ x1' \le_{R1} x2' \Longrightarrow$ *in-dom* $(\le_{L2\ (r1\ x1')\ (r1\ x2')})\ y$
$\Longrightarrow$
$(\le_{R2\ x1'\ x2'})\ (l2_{x2'\ (r1\ x1')}\ y) \le (\le_{R2\ x1'\ x2'})\ (l2_{x1'\ (r1\ x1')}\ y)$
and *ge-R2-l2-le2*: $\bigwedge x1'\ x2'\ y.\ x1' \le_{R1} x2' \Longrightarrow$ *in-codom* $(\le_{L2\ (r1\ x1')\ (r1\ x2')})$
$y \Longrightarrow$
$(\ge_{R2\ x1'\ x2'})\ (l2_{x2'\ (r1\ x1')}\ y) \le (\ge_{R2\ x1'\ x2'})\ (l2_{x2'\ (r1\ x2')}\ y)$
shows $((\le_L) \Rrightarrow_m (\le_R))\ l$
⟨*proof*⟩

lemma *mono-wrt-rel-left-in-dom-mono-left-assm*:

assumes $([in\text{-}dom\ (\le_{L2\ (r1\ x1')\ (r1\ x2')})] \Rrightarrow (\le_{R2\ x1'\ x2'}))$
$(l2_{x1'\ (r1\ x1')})\ (l2_{x2'\ (r1\ x1')})$
and $transitive\ (\le_{R2\ x1'\ x2'})$
and $x1' \le_{R1} x2'$
and $in\text{-}dom\ (\le_{L2\ (r1\ x1')\ (r1\ x2')})\ y$
shows $(\le_{R2\ x1'\ x2'})\ (l2_{x2'\ (r1\ x1')}\ y) \le (\le_{R2\ x1'\ x2'})\ (l2_{x1'\ (r1\ x1')}\ y)$
⟨*proof*⟩

lemma *mono-wrt-rel-left-in-codom-mono-left-assm*:
assumes $([in\text{-}codom\ (\le_{L2\ (r1\ x1')\ (r1\ x2')})] \Rrightarrow (\le_{R2\ x1'\ x2'}))$
$(l2_{x2'\ (r1\ x1')})\ (l2_{x2'\ (r1\ x2')})$
and $transitive\ (\le_{R2\ x1'\ x2'})$
and $x1' \le_{R1} x2'$
and $in\text{-}codom\ (\le_{L2\ (r1\ x1')\ (r1\ x2')})\ y$
shows $(\ge_{R2\ x1'\ x2'})\ (l2_{x2'\ (r1\ x1')}\ y) \le (\ge_{R2\ x1'\ x2'})\ (l2_{x2'\ (r1\ x2')}\ y)$
⟨*proof*⟩

lemma *mono-wrt-rel-left-if-transitiveI*:
assumes $((\le_{R1}) \Rrightarrow_m (\le_{L1}))\ r1$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$
$((\le_{L2\ (r1\ x1')\ (r1\ x2')}) \Rrightarrow_m (\le_{R2\ (\varepsilon_1\ x1')\ x2'}))\ (l2_{x2'\ (r1\ x1')})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x1')\ x2'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$
$([in\text{-}dom\ (\le_{L2\ (r1\ x1')\ (r1\ x2')})] \Rrightarrow (\le_{R2\ x1'\ x2'}))\ (l2_{x1'\ (r1\ x1')})\ (l2_{x2'\ (r1\ x1')})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$
$([in\text{-}codom\ (\le_{L2\ (r1\ x1')\ (r1\ x2')})] \Rrightarrow (\le_{R2\ x1'\ x2'}))\ (l2_{x2'\ (r1\ x1')})\ (l2_{x2'\ (r1\ x2')})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow transitive\ (\le_{R2\ x1'\ x2'})$
shows $((\le_L) \Rrightarrow_m (\le_R))\ l$
⟨*proof*⟩

lemma *mono-wrt-rel-left2-if-mono-wrt-rel-left2-if-left-GaloisI*:
assumes $((\le_{R1}) \Rrightarrow_m (\le_{L1}))\ r1$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \Rrightarrow_m (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})$
shows $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$
$((\le_{L2\ (r1\ x1')\ (r1\ x2')}) \Rrightarrow_m (\le_{R2\ (\varepsilon_1\ x1')\ x2'}))\ (l2_{x2'\ (r1\ x1')})$
⟨*proof*⟩

interpretation *flip-inv* :
$transport\text{-}Dep\text{-}Fun\text{-}Rel\ (\ge_{R1})\ (\ge_{L1})\ r1\ l1\ flip2\ R2\ flip2\ L2\ r2\ l2$
rewrites $flip\text{-}inv.R \equiv (\ge_L)$ **and** $flip\text{-}inv.L \equiv (\ge_R)$
and $flip\text{-}inv.t1.counit \equiv \eta_1$
and $\bigwedge R\ x\ y.\ (flip2\ R\ x\ y)^{-1} \equiv R\ y\ x$
and $\bigwedge R\ x1\ x2.\ in\text{-}dom\ (flip2\ R\ x1\ x2) \equiv in\text{-}codom\ (R\ x2\ x1)$
and $\bigwedge R\ x1\ x2.\ in\text{-}codom\ (flip2\ R\ x1\ x2) \equiv in\text{-}dom\ (R\ x2\ x1)$
and $\bigwedge R\ S.\ (R^{-1} \Rrightarrow_m S^{-1}) \equiv (R \Rrightarrow_m S)$
and $\bigwedge x1\ x2\ x1'\ x2'.\ (flip2\ R2\ x1'\ x2' \Rrightarrow_m flip2\ L2\ x1\ x2) \equiv$

$((\le_{R2\ x2'\ x1'}) \Rrightarrow_m (\le_{L2\ x2\ x1}))$
and $\bigwedge x1\ x2\ x3\ x4.\ flip2\ L2\ x1\ x2 \le flip2\ L2\ x3\ x4 \equiv (\le_{L2\ x2\ x1}) \le (\le_{L2\ x4\ x3})$
and $\bigwedge x1'\ x2'\ y1\ y2.$
$flip\text{-}inv.dfro2.right\text{-}infix\ y1\ x1'\ x2' \le flip\text{-}inv.dfro2.right\text{-}infix\ y2\ x1'\ x2' \equiv$
$(\ge_{L2\ x2'\ x1'})\ y1 \le (\ge_{L2\ x2'\ x1'})\ y2$
and $\bigwedge P\ x1\ x2.\ ([P] \Rrightarrow flip2\ L2\ x1\ x2) \equiv ([P] \Rrightarrow (\ge_{L2\ x2\ x1}))$
and $\bigwedge P\ R.\ ([P] \Rrightarrow R^{-1}) \equiv ([P] \Rrightarrow R)^{-1}$
and $\bigwedge x1\ x2.\ transitive\ (flip2\ L2\ x1\ x2) \equiv transitive\ (\le_{L2\ x2\ x1})$
⟨*proof*⟩

lemma *mono-wrt-rel-rightI*:
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow ((\le_{R2\ (l1\ x1)\ (l1\ x2)}) \Rrightarrow_m (\le_{L2\ x1\ (\eta_1\ x2)}))\ (r2_{x1\ (l1\ x2)})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2\ y'.\ x1 \le_{L1} x2 \Longrightarrow in\text{-}codom\ (\le_{R2\ (l1\ x1)\ (l1\ x2)})\ y' \Longrightarrow$
$(\ge_{L2\ x1\ x2})\ (r2_{x1\ (l1\ x2)}\ y') \le (\ge_{L2\ x1\ x2})\ (r2_{x2\ (l1\ x2)}\ y')$
and $\bigwedge x1\ x2\ y'.\ x1 \le_{L1} x2 \Longrightarrow in\text{-}dom\ (\le_{R2\ (l1\ x1)\ (l1\ x2)})\ y' \Longrightarrow$
$(\le_{L2\ x1\ x2})\ (r2_{x1\ (l1\ x2)}\ y') \le (\le_{L2\ x1\ x2})\ (r2_{x1\ (l1\ x1)}\ y')$
shows $((\le_R) \Rrightarrow_m (\le_L))\ r$
⟨*proof*⟩

lemma *mono-wrt-rel-right-if-transitiveI*:
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow ((\le_{R2\ (l1\ x1)\ (l1\ x2)}) \Rrightarrow_m (\le_{L2\ x1\ (\eta_1\ x2)}))\ (r2_{x1\ (l1\ x2)})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$
$([in\text{-}codom\ (\le_{R2\ (l1\ x1)\ (l1\ x2)})] \Rrightarrow (\le_{L2\ x1\ x2}))\ (r2_{x1\ (l1\ x2)})\ (r2_{x2\ (l1\ x2)})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$
$([in\text{-}dom\ (\le_{R2\ (l1\ x1)\ (l1\ x2)})] \Rrightarrow (\le_{L2\ x1\ x2}))\ (r2_{x1\ (l1\ x1)})\ (r2_{x1\ (l1\ x2)})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow transitive\ (\le_{L2\ x1\ x2})$
shows $((\le_R) \Rrightarrow_m (\le_L))\ r$
⟨*proof*⟩

lemma *mono-wrt-rel-right2-if-mono-wrt-rel-right2-if-left-GaloisI*:
assumes *assms1*: $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1\ ((\le_{L1}) \trianglelefteq_h (\le_{R1}))\ l1\ r1$
and *mono-r2*: $\bigwedge x\ x'.\ x\ _{L1}\lessapprox x' \Longrightarrow ((\le_{R2\ (l1\ x)\ x'}) \Rrightarrow_m (\le_{L2\ x\ (r1\ x')}))\ (r2_{x\ x'})$
shows $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow ((\le_{R2\ (l1\ x1)\ (l1\ x2)}) \Rrightarrow_m (\le_{L2\ x1\ (\eta_1\ x2)}))$
$(r2_{x1\ (l1\ x2)})$
⟨*proof*⟩

end

Function Relator **context** *transport-Fun-Rel*
begin

lemma *mono-wrt-rel-leftI*:
assumes $((\le_{R1}) \Rrightarrow_m (\le_{L1}))\ r1$

and $((\le_{L2}) \Rrightarrow_m (\le_{R2}))$ *l2*
shows $((\le_L) \Rrightarrow_m (\le_R))$ *l*
⟨*proof*⟩

end

Monotone Dependent Function Relator **context** *transport-Mono-Dep-Fun-Rel*
begin

lemmas *mono-wrt-rel-leftI* = *mono-wrt-rel-Refl-Rel-Refl-Rel-if-mono-wrt-rel*
[*of tdfr.L tdfr.R l*, *folded transport-defs*]

end

Monotone Function Relator **context** *transport-Mono-Fun-Rel*
begin

lemmas *mono-wrt-rel-leftI* = *tpdfr.mono-wrt-rel-leftI*[*OF tfr.mono-wrt-rel-leftI*]

end

end

2.8.3 Galois Property

theory *Transport-Functions-Galois-Property*
imports
Transport-Functions-Monotone
begin

Dependent Function Relator **context** *transport-Dep-Fun-Rel*
begin

context
begin

interpretation *flip* : *transport-Dep-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2* ⟨*proof*⟩

lemma *left-right-rel-if-left-rel-rightI*:
assumes *mono-r1*: $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*
and *half-galois-prop-left1*: $((\le_{L1}) \ {}_h\trianglelefteq (\le_{R1}))$ *l1 r1*
and *refl-R1*: *reflexive-on* (*in-dom* $(\le_{R1})$) $(\le_{R1})$
and *half-galois-prop-left2*: $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow$
$((\le_{L2\ (r1\ x')\ (r1\ x')})\ {}_h\trianglelefteq\ (\le_{R2\ (\varepsilon_1\ x')\ x'}))\ (l2_{\ x'\ (r1\ x')})\ (r2_{(r1\ x')\ x'})$
and *R2-le1*: $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x')\ x'}) \le (\le_{R2\ x'\ x'})$
and *R2-le2*: $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x1'\ x1'}) \le (\le_{R2\ x1'\ x2'})$
and *ge-L2-r2-le2*: $\bigwedge x'\ y'.\ x' \le_{R1} x' \Longrightarrow$ *in-codom* $(\le_{R2\ (\varepsilon_1\ x')\ x'})\ y' \Longrightarrow$

$(\geq_{L2\ (r1\ x')\ (r1\ x')})\ (r2_{(r1\ x')\ (\varepsilon_1\ x')}\ y') \leq (\geq_{L2\ (r1\ x')\ (r1\ x')})\ (r2_{(r1\ x')\ x'}\ y')$
and *trans-R2*: $\bigwedge x1'\ x2'.\ x1' \leq_{R1} x2' \Longrightarrow transitive\ (\leq_{R2\ x1'\ x2'})$
and $g \leq_R g$
and $f \leq_L r\ g$
shows $l\ f \leq_R g$
⟨*proof*⟩

lemma *left-right-rel-if-left-rel-right-ge-left2-assmI*:
assumes *mono-r1*: $((\leq_{R1}) \Rrightarrow_m (\leq_{L1}))\ r1$
and $((\leq_{L1})\ {}_h\trianglelefteq (\leq_{R1}))\ l1\ r1$
and $([in\text{-}codom\ (\leq_{R2\ (\varepsilon_1\ x')\ x'})] \Rrightarrow (\leq_{L2\ (r1\ x')\ (r1\ x')}))$
$(r2_{(r1\ x')\ (\varepsilon_1\ x')})\ (r2_{(r1\ x')\ x'})$
and $\bigwedge x1\ x2.\ x1 \leq_{L1} x2 \Longrightarrow transitive\ (\leq_{L2\ x1\ x2})$
and $x' \leq_{R1} x'$
and $in\text{-}codom\ (\leq_{R2\ (\varepsilon_1\ x')\ x'})\ y'$
shows $(\geq_{L2\ (r1\ x')\ (r1\ x')})\ (r2_{(r1\ x')\ (\varepsilon_1\ x')}\ y') \leq (\geq_{L2\ (r1\ x')\ (r1\ x')})\ (r2_{(r1\ x')\ x'}\ y')$
⟨*proof*⟩

interpretation *flip-inv* :
transport-Dep-Fun-Rel $(\geq_{R1})\ (\geq_{L1})$ *r1 l1 flip2 R2 flip2 L2 r2 l2*
rewrites $flip\text{-}inv.L \equiv (\geq_R)$ **and** $flip\text{-}inv.R \equiv (\geq_L)$
and $flip\text{-}inv.t1.counit \equiv \eta_1$
and $\bigwedge R\ x\ y.\ (flip2\ R\ x\ y)^{-1} \equiv R\ y\ x$
and $\bigwedge R.\ in\text{-}dom\ R^{-1} \equiv in\text{-}codom\ R$
and $\bigwedge R\ x1\ x2.\ in\text{-}codom\ (flip2\ R\ x1\ x2) \equiv in\text{-}dom\ (R\ x2\ x1)$
and $\bigwedge R\ S.\ (R^{-1} \Rrightarrow_m S^{-1}) \equiv (R \Rrightarrow_m S)$
and $\bigwedge R\ S\ x1\ x2\ x1'\ x2'.\ (flip2\ R\ x1\ x2\ {}_h\trianglelefteq flip2\ S\ x1'\ x2') \equiv (S\ x2'\ x1' \trianglelefteq_h R\ x2\ x1)^{-1}$
and $\bigwedge R\ S.\ (R^{-1}\ {}_h\trianglelefteq S^{-1}) \equiv (S \trianglelefteq_h R)^{-1}$
and $\bigwedge x1\ x2\ x3\ x4.\ flip2\ L2\ x1\ x2 \leq flip2\ L2\ x3\ x4 \equiv (\leq_{L2\ x2\ x1}) \leq (\leq_{L2\ x4\ x3})$
and $\bigwedge(R :: 'z \Rightarrow \text{-})\ (P :: 'z \Rightarrow bool).\ reflexive\text{-}on\ P\ R^{-1} \equiv reflexive\text{-}on\ P\ R$
and $\bigwedge R\ x1\ x2.\ transitive\ (flip2\ R\ x1\ x2) \equiv transitive\ (R\ x2\ x1)$
and $\bigwedge x\ x.\ ([in\text{-}dom\ (\leq_{L2\ x'\ \eta_1\ x'})] \Rrightarrow flip2\ R2\ (l1\ x')\ (l1\ x'))$
$\equiv ([in\text{-}dom\ (\leq_{L2\ x'\ \eta_1\ x'})] \Rrightarrow (\leq_{R2\ (l1\ x')\ (l1\ x')}))^{-1}$
⟨*proof*⟩

lemma *left-rel-right-if-left-right-relI*:
assumes $((\leq_{L1}) \Rrightarrow_m (\leq_{R1}))\ l1$
and $((\leq_{L1}) \trianglelefteq_h (\leq_{R1}))\ l1\ r1$
and $reflexive\text{-}on\ (in\text{-}codom\ (\leq_{L1}))\ (\leq_{L1})$
and $\bigwedge x.\ x \leq_{L1} x \Longrightarrow ((\leq_{L2\ x\ (\eta_1\ x)}) \trianglelefteq_h (\leq_{R2\ (l1\ x)\ (l1\ x)}))\ (l2_{(l1\ x)\ x})\ (r2_{x\ (l1\ x)})$
and $\bigwedge x1\ x2.\ x1 \leq_{L1} x2 \Longrightarrow (\leq_{L2\ x2\ x2}) \leq (\leq_{L2\ x1\ x2})$
and $\bigwedge x.\ x \leq_{L1} x \Longrightarrow (\leq_{L2\ x\ (\eta_1\ x)}) \leq (\leq_{L2\ x\ x})$
and $\bigwedge x\ y.\ x \leq_{L1} x \Longrightarrow in\text{-}dom\ (\leq_{L2\ x\ (\eta_1\ x)})\ y \Longrightarrow$
$(\leq_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ (\eta_1\ x)}\ y) \leq (\leq_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ x}\ y)$

and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow \textit{transitive}\ (\le_{L2\ x1\ x2})$
and $f \le_L f$
and $l\ f \le_R g$
shows $f \le_L r\ g$
⟨*proof*⟩

lemma *left-rel-right-if-left-right-rel-le-right2-assmI*:
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1$
and $((\le_{L1}) \trianglelefteq_h (\le_{R1}))^{-1}\ r1\ l1$
and $([\textit{in-dom}\ (\le_{L2\ x\ (\eta_1\ x)})] \Rrightarrow (\le_{R2\ (l1\ x)\ (l1\ x)}))\ (l2_{(l1\ x)\ x})\ (l2_{(l1\ x)\ (\eta_1\ x)})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow \textit{transitive}\ (\le_{R2\ x1'\ x2'})$
and $x \le_{L1} x$
and $\textit{in-dom}\ (\le_{L2\ x\ (\eta_1\ x)})\ y$
shows $(\le_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ (\eta_1\ x)}\ y) \le (\le_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ x}\ y)$
⟨*proof*⟩

end

lemma *left-rel-right-iff-left-right-relI*:
assumes $((\le_{L1}) \dashv (\le_{R1}))\ l1\ r1$
and $\textit{reflexive-on}\ (\textit{in-codom}\ (\le_{L1}))\ (\le_{L1})$
and $\textit{reflexive-on}\ (\textit{in-dom}\ (\le_{R1}))\ (\le_{R1})$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow$
$((\le_{L2\ (r1\ x')\ (r1\ x')})\ {}_h\trianglelefteq\ (\le_{R2\ (\varepsilon_1\ x')\ x'}))\ (l2\ {}_{x'\ (r1\ x')})\ (r2_{(r1\ x')\ x'})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow ((\le_{L2\ x\ (\eta_1\ x)}) \trianglelefteq_h (\le_{R2\ (l1\ x)\ (l1\ x)}))\ (l2_{(l1\ x)\ x})\ (r2_{x\ (l1\ x)})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x2\ x2}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow (\le_{L2\ x\ (\eta_1\ x)}) \le (\le_{L2\ x\ x})$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x')\ x'}) \le (\le_{R2\ x'\ x'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x1'\ x1'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x\ y.\ x \le_{L1} x \Longrightarrow \textit{in-dom}\ (\le_{L2\ x\ (\eta_1\ x)})\ y \Longrightarrow$
$(\le_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ (\eta_1\ x)}\ y) \le (\le_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ x}\ y)$
and $\bigwedge x'\ y'.\ x' \le_{R1} x' \Longrightarrow \textit{in-codom}\ (\le_{R2\ (\varepsilon_1\ x')\ x'})\ y' \Longrightarrow$
$(\ge_{L2\ (r1\ x')\ (r1\ x')})\ (r2_{(r1\ x')\ (\varepsilon_1\ x')}\ y') \le (\ge_{L2\ (r1\ x')\ (r1\ x')})\ (r2_{(r1\ x')\ x'}$
$y')$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow \textit{transitive}\ (\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow \textit{transitive}\ (\le_{R2\ x1'\ x2'})$
and $f \le_L f$
and $g \le_R g$
shows $f \le_L r\ g \longleftrightarrow l\ f \le_R g$
⟨*proof*⟩

lemma *half-galois-prop-left2-if-half-galois-prop-left2-if-left-GaloisI*:
assumes $((\le_{R1}) \Rrightarrow_m (\le_{L1}))\ r1$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')})\ {}_h\trianglelefteq\ (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $x' \le_{R1} x'$
shows $((\le_{L2\ (r1\ x')\ (r1\ x')})\ {}_h\trianglelefteq\ (\le_{R2\ (\varepsilon_1\ x')\ x'}))\ (l2\ {}_{x'\ (r1\ x')})\ (r2_{(r1\ x')\ x'})$
⟨*proof*⟩

lemma *half-galois-prop-right2-if-half-galois-prop-right2-if-left-GaloisI*:
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))$ *l1*
and $((\le_{L1}) \trianglelefteq_h (\le_{R1}))$ *l1 r1*
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \trianglelefteq_h (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $x \le_{L1} x$
shows $((\le_{L2\ x\ (\eta_1\ x)}) \trianglelefteq_h (\le_{R2\ (l1\ x)\ (l1\ x)}))\ (l2_{(l1\ x)\ x})\ (r2_{x\ (l1\ x)})$
⟨*proof*⟩

lemma *left-rel-right-iff-left-right-relI′*:
assumes $((\le_{L1}) \dashv (\le_{R1}))$ *l1 r1*
and *reflexive-on* (*in-codom* $(\le_{L1})$) $(\le_{L1})$
and *reflexive-on* (*in-dom* $(\le_{R1})$) $(\le_{R1})$
and *galois-prop2*: $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow$
$((\le_{L2\ x\ (r1\ x')}) \trianglelefteq (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x2\ x2}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow (\le_{L2\ x\ (\eta_1\ x)}) \le (\le_{L2\ x\ x})$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x')\ x'}) \le (\le_{R2\ x'\ x'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x1'\ x1'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow$
$([\mathit{in\text{-}dom}\ (\le_{L2\ x\ (\eta_1\ x)})] \Rrightarrow (\le_{R2\ (l1\ x)\ (l1\ x)}))\ (l2_{(l1\ x)\ x})\ (l2_{(l1\ x)\ (\eta_1\ x)})$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow$
$([\mathit{in\text{-}codom}\ (\le_{R2\ (\varepsilon_1\ x')\ x'})] \Rrightarrow (\le_{L2\ (r1\ x')\ (r1\ x')}))\ (r2_{(r1\ x')\ (\varepsilon_1\ x')})\ (r2_{(r1\ x')\ x'})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$ *transitive* $(\le_{R2\ x1'\ x2'})$
and $f \le_L f$
and $g \le_R g$
shows $f \le_L r\ g \longleftrightarrow l\ f \le_R g$
⟨*proof*⟩

lemma *left-rel-right-iff-left-right-rel-if-galois-prop-le-assms-leftI*:
assumes *galois-conn1*: $((\le_{L1}) \dashv (\le_{R1}))$ *l1 r1*
and *refl-L1*: *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and *antimono-L2*:
$([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid (x2 \le_{L1} x3 \land x4 \le_{L1} \eta_1\ x3)] \Rrightarrow (\ge))$
L2
shows $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x2\ x2}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
⟨*proof*⟩

lemma *left-rel-right-iff-left-right-rel-if-galois-prop-le-assms-rightI*:
assumes *galois-conn1*: $((\le_{L1}) \dashv (\le_{R1}))$ *l1 r1*
and *refl-R1*: *reflexive-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$
and *mono-R2*:
$([x1'\ x2' :: (\le_{R1}) \mid \varepsilon_1\ x2' \le_{R1} x1'] \Rrightarrow_m [x3'\ x4' :: (\le_{R1}) \mid x2' \le_{R1} x3'] \Rrightarrow$
$(\le))$ *R2*
shows $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x1')\ x2'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x1'\ x1'}) \le (\le_{R2\ x1'\ x2'})$

⟨*proof*⟩

corollary *left-rel-right-iff-left-right-rel-if-monoI*:
assumes $((\le_{L1}) \dashv (\le_{R1}))$ *l1 r1*
and *reflexive-on* $($*in-field* $(\le_{L1}))\ (\le_{L1})$
and *reflexive-on* $($*in-field* $(\le_{R1}))\ (\le_{R1})$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \trianglelefteq (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid (x2 \le_{L1} x3 \wedge x4 \le_{L1} \eta_1\ x3)] \Rrightarrow (\ge))\ L2$
and $([x1'\ x2' :: (\le_{R1}) \mid \varepsilon_1\ x2' \le_{R1} x1'] \Rrightarrow_m [x3'\ x4' :: (\le_{R1}) \mid x2' \le_{R1} x3'] \Rrightarrow (\le))\ R2$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow$
$([in\text{-}dom\ (\le_{L2\ x\ (\eta_1\ x)})] \Rrightarrow (\le_{R2\ (l1\ x)\ (l1\ x)}))\ (l2_{(l1\ x)\ x})\ (l2_{(l1\ x)\ (\eta_1\ x)})$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow$
$([in\text{-}codom\ (\le_{R2\ (\varepsilon_1\ x')\ x'})] \Rrightarrow (\le_{L2\ (r1\ x')\ (r1\ x')}))\ (r2_{(r1\ x')\ (\varepsilon_1\ x')})\ (r2_{(r1\ x')\ x'})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$ *transitive* $(\le_{R2\ x1'\ x2'})$
and $f \le_L f$
and $g \le_R g$
shows $f \le_L r\ g \longleftrightarrow l\ f \le_R g$
⟨*proof*⟩

end

Function Relator **context** *transport-Fun-Rel*
begin

corollary *left-right-rel-if-left-rel-rightI*:
assumes $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*
and $((\le_{L1})\ {}_h\trianglelefteq\ (\le_{R1}))$ *l1 r1*
and *reflexive-on* $($*in-dom* $(\le_{R1}))\ (\le_{R1})$
and $((\le_{L2})\ {}_h\trianglelefteq\ (\le_{R2}))$ *l2 r2*
and *transitive* $(\le_{R2})$
and $g \le_R g$
and $f \le_L r\ g$
shows $l\ f \le_R g$
⟨*proof*⟩

corollary *left-rel-right-if-left-right-relI*:
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))$ *l1*
and $((\le_{L1}) \trianglelefteq_h (\le_{R1}))$ *l1 r1*
and *reflexive-on* $($*in-codom* $(\le_{L1}))\ (\le_{L1})$
and $((\le_{L2}) \trianglelefteq_h (\le_{R2}))$ *l2 r2*
and *transitive* $(\le_{L2})$
and $f \le_L f$
and $l\ f \le_R g$
shows $f \le_L r\ g$
⟨*proof*⟩

corollary *left-rel-right-iff-left-right-relI*:
  assumes $((\le_{L1}) \dashv (\le_{R1}))$ *l1 r1*
  and *reflexive-on* (*in-codom* $(\le_{L1})$) $(\le_{L1})$
  and *reflexive-on* (*in-dom* $(\le_{R1})$) $(\le_{R1})$
  and $((\le_{L2}) \trianglelefteq (\le_{R2}))$ *l2 r2*
  and *transitive* $(\le_{L2})$
  and *transitive* $(\le_{R2})$
  and $f \le_L f$
  and $g \le_R g$
  shows $f \le_L r\ g \longleftrightarrow l\ f \le_R g$
  ⟨*proof*⟩

end

Monotone Dependent Function Relator **context** *transport-Mono-Dep-Fun-Rel*
begin

lemma *half-galois-prop-left-left-rightI*:
  assumes $(tdfr.L \Rrightarrow_m tdfr.R)\ l$
  and $((\le_{R1}) \Rrightarrow_m (\le_{L1}))\ r1$
  and $((\le_{L1})\ {}_h\trianglelefteq (\le_{R1}))\ l1\ r1$
  and *reflexive-on* (*in-dom* $(\le_{R1})$) $(\le_{R1})$
  and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow$
    $((\le_{L2\ (r1\ x')\ (r1\ x')})\ {}_h\trianglelefteq (\le_{R2\ (\varepsilon_1\ x')\ x'}))\ (l2_{\ x'\ (r1\ x')})\ (r2_{(r1\ x')\ x'})$
  and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x')\ x'}) \le (\le_{R2\ x'\ x'})$
  and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x1'\ x1'}) \le (\le_{R2\ x1'\ x2'})$
  and $\bigwedge x'\ y'.\ x' \le_{R1} x' \Longrightarrow$ *in-codom* $(\le_{R2\ (\varepsilon_1\ x')\ x'})\ y' \Longrightarrow$
    $(\ge_{L2\ (r1\ x')\ (r1\ x')})\ (r2_{(r1\ x')\ (\varepsilon_1\ x')}\ y') \le (\ge_{L2\ (r1\ x')\ (r1\ x')})\ (r2_{(r1\ x')\ x'}$
$y')$
  and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$ *transitive* $(\le_{R2\ x1'\ x2'})$
  shows $((\le_L)\ {}_h\trianglelefteq (\le_R))\ l\ r$
  ⟨*proof*⟩

lemma *half-galois-prop-right-left-rightI*:
  assumes $(tdfr.R \Rrightarrow_m tdfr.L)\ r$
  and $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1$
  and $((\le_{L1}) \trianglelefteq_h (\le_{R1}))\ l1\ r1$
  and *reflexive-on* (*in-codom* $(\le_{L1})$) $(\le_{L1})$
  and $\bigwedge x.\ x \le_{L1} x \Longrightarrow ((\le_{L2\ x\ (\eta_1\ x)}) \trianglelefteq_h (\le_{R2\ (l1\ x)\ (l1\ x)}))\ (l2_{(l1\ x)\ x})\ (r2_{x\ (l1\ x)})$
  and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x2\ x2}) \le (\le_{L2\ x1\ x2})$
  and $\bigwedge x.\ x \le_{L1} x \Longrightarrow (\le_{L2\ x\ (\eta_1\ x)}) \le (\le_{L2\ x\ x})$
  and $\bigwedge x\ y.\ x \le_{L1} x \Longrightarrow$ *in-dom* $(\le_{L2\ x\ (\eta_1\ x)})\ y \Longrightarrow$
    $(\le_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ (\eta_1\ x)}\ y) \le (\le_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ x}\ y)$
  and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
  shows $((\le_L) \trianglelefteq_h (\le_R))\ l\ r$
  ⟨*proof*⟩

corollary *galois-prop-left-rightI*:

assumes $(tdfr.L \Rrightarrow_m tdfr.R)$ l **and** $(tdfr.R \Rrightarrow_m tdfr.L)$ r
and $((\le_{L1}) \dashv (\le_{R1}))$ $l1$ $r1$
and *reflexive-on* $(in\text{-}codom\ (\le_{L1}))\ (\le_{L1})$
and *reflexive-on* $(in\text{-}dom\ (\le_{R1}))\ (\le_{R1})$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow$
$((\le_{L2\ (r1\ x')\ (r1\ x')})\ {}_h\trianglelefteq\ (\le_{R2\ (\varepsilon_1\ x')\ x'}))\ (l2_{x'\ (r1\ x')})\ (r2_{(r1\ x')\ x'})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow ((\le_{L2\ x\ (\eta_1\ x)}) \trianglelefteq_h (\le_{R2\ (l1\ x)\ (l1\ x)}))\ (l2_{(l1\ x)\ x})\ (r2_{x\ (l1\ x)})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x2\ x2}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow (\le_{L2\ x\ (\eta_1\ x)}) \le (\le_{L2\ x\ x})$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x')\ x'}) \le (\le_{R2\ x'\ x'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x1'\ x1'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x\ y.\ x \le_{L1} x \Longrightarrow in\text{-}dom\ (\le_{L2\ x\ (\eta_1\ x)})\ y \Longrightarrow$
$(\le_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ (\eta_1\ x)}\ y) \le (\le_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ x}\ y)$
and $\bigwedge x'\ y'.\ x' \le_{R1} x' \Longrightarrow in\text{-}codom\ (\le_{R2\ (\varepsilon_1\ x')\ x'})\ y' \Longrightarrow$
$(\ge_{L2\ (r1\ x')\ (r1\ x')})\ (r2_{(r1\ x')\ (\varepsilon_1\ x')}\ y') \le (\ge_{L2\ (r1\ x')\ (r1\ x')})\ (r2_{(r1\ x')\ x'}$
$y')$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$ *transitive* $(\le_{R2\ x1'\ x2'})$
shows $((\le_L) \trianglelefteq (\le_R))$ l r
⟨*proof*⟩

corollary *galois-prop-left-rightI'*:
assumes $(tdfr.L \Rrightarrow_m tdfr.R)$ l **and** $(tdfr.R \Rrightarrow_m tdfr.L)$ r
and $((\le_{L1}) \dashv (\le_{R1}))$ $l1$ $r1$
and *reflexive-on* $(in\text{-}codom\ (\le_{L1}))\ (\le_{L1})$
and *reflexive-on* $(in\text{-}dom\ (\le_{R1}))\ (\le_{R1})$
and *galois-prop2*: $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow$
$((\le_{L2\ x\ (r1\ x')}) \trianglelefteq (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x2\ x2}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow (\le_{L2\ x\ (\eta_1\ x)}) \le (\le_{L2\ x\ x})$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x')\ x'}) \le (\le_{R2\ x'\ x'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x1'\ x1'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow$
$([in\text{-}dom\ (\le_{L2\ x\ (\eta_1\ x)})] \Rrightarrow (\le_{R2\ (l1\ x)\ (l1\ x)}))\ (l2_{(l1\ x)\ x})\ (l2_{(l1\ x)\ (\eta_1\ x)})$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow$
$([in\text{-}codom\ (\le_{R2\ (\varepsilon_1\ x')\ x'})] \Rrightarrow (\le_{L2\ (r1\ x')\ (r1\ x')}))\ (r2_{(r1\ x')\ (\varepsilon_1\ x')})\ (r2_{(r1\ x')\ x'})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$ *transitive* $(\le_{R2\ x1'\ x2'})$
shows $((\le_L) \trianglelefteq (\le_R))$ l r
⟨*proof*⟩

corollary *galois-prop-left-right-if-mono-if-galois-propI*:
assumes $(tdfr.L \Rrightarrow_m tdfr.R)$ l **and** $(tdfr.R \Rrightarrow_m tdfr.L)$ r
and $((\le_{L1}) \dashv (\le_{R1}))$ $l1$ $r1$
and *reflexive-on* $(in\text{-}field\ (\le_{L1}))\ (\le_{L1})$
and *reflexive-on* $(in\text{-}field\ (\le_{R1}))\ (\le_{R1})$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \trianglelefteq (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$

and $([x1\ x2 :: (\leq_{L1})] \Rrightarrow_m [x3\ x4 :: (\leq_{L1}) \mid (x2 \leq_{L1} x3 \wedge x4 \leq_{L1} \eta_1\ x3)] \Rrightarrow (\geq))\ L2$
and $([x1'\ x2' :: (\leq_{R1}) \mid \varepsilon_1\ x2' \leq_{R1} x1'] \Rrightarrow_m [x3'\ x4' :: (\leq_{R1}) \mid x2' \leq_{R1} x3'] \Rrightarrow (\leq))\ R2$
and $\bigwedge x.\ x \leq_{L1} x \Longrightarrow$
$([in\text{-}dom\ (\leq_{L2\ x\ (\eta_1\ x)})] \Rrightarrow (\leq_{R2\ (l1\ x)\ (l1\ x)}))\ (l2_{(l1\ x)\ x})\ (l2_{(l1\ x)\ (\eta_1\ x)})$
and $\bigwedge x'.\ x' \leq_{R1} x' \Longrightarrow$
$([in\text{-}codom\ (\leq_{R2\ (\varepsilon_1\ x')\ x'})] \Rrightarrow (\leq_{L2\ (r1\ x')\ (r1\ x')}))\ (r2_{(r1\ x')\ (\varepsilon_1\ x')})\ (r2_{(r1\ x')\ x'})$
and $\bigwedge x1\ x2.\ x1 \leq_{L1} x2 \Longrightarrow transitive\ (\leq_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \leq_{R1} x2' \Longrightarrow transitive\ (\leq_{R2\ x1'\ x2'})$
shows $((\leq_L) \trianglelefteq (\leq_R))\ l\ r$
$\langle proof \rangle$

Note that we could further rewrite $[\![(tdfr.L \Rrightarrow_m tdfr.R)\ l;\ (tdfr.R \Rrightarrow_m tdfr.L)\ r;\ t1.galois\text{-}connection;\ reflexive\text{-}on\ (in\text{-}field\ (\leq_{L1}))\ (\leq_{L1});\ reflexive\text{-}on\ (in\text{-}field\ (\leq_{R1}))\ (\leq_{R1});\ \bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow t2.galois\text{-}prop\ x\ x'\ l2_{x'\ x}\ r2_{x\ x'};\ ([x1\ x2 :: (\leq_{L1})] \Rrightarrow_m [x3\ x4 :: (\leq_{L1})] \Rrightarrow (x2 \leq_{L1} x3 \wedge x4 \leq_{L1} \eta_1\ x3) \longrightarrow (\lambda x\ y.\ y \leq x))\ L2;\ ([x1'\ x2' :: (\leq_{R1})] \Rrightarrow_m \varepsilon_1\ x2' \leq_{R1} x1' \longrightarrow ([x3'\ x4' :: (\leq_{R1})] \Rrightarrow x2' \leq_{R1} x3' \longrightarrow (\leq)))\ R2;\ \bigwedge x.\ x \leq_{L1} x \Longrightarrow ([in\text{-}dom\ (\leq_{L2\ x\ \eta_1\ x})] \Rrightarrow \leq_{R2\ l1\ x\ l1\ x})\ l2_{l1\ x\ x}\ l2_{l1\ x\ \eta_1\ x};\ \bigwedge x'.\ x' \leq_{R1} x' \Longrightarrow ([in\text{-}codom\ (\leq_{R2\ \varepsilon_1\ x'\ x'})] \Rrightarrow \leq_{L2\ r1\ x'\ r1\ x'})\ r2_{r1\ x'\ \varepsilon_1\ x'}\ r2_{r1\ x'\ x'};\ \bigwedge x1\ x2.\ x1 \leq_{L1} x2 \Longrightarrow transitive\ (\leq_{L2\ x1\ x2});\ \bigwedge x1'\ x2'.\ x1' \leq_{R1} x2' \Longrightarrow transitive\ (\leq_{R2\ x1'\ x2'})]\!] \Longrightarrow galois\text{-}prop\ l\ r$, as we will do later for Galois connections, by applying $[\![((\leq_{R1}) \Rrightarrow_m (\leq_{L1}))\ r1;\ \bigwedge x1'\ x2'.\ x1' \leq_{R1} x2' \Longrightarrow (\leq_{L2\ r1\ x1'\ r1\ x2'} \Rrightarrow_m \leq_{R2\ \varepsilon_1\ x1'\ x2'})\ l2_{x2'\ r1\ x1'};\ \bigwedge x1'\ x2'.\ x1' \leq_{R1} x2' \Longrightarrow (\leq_{R2\ \varepsilon_1\ x1'\ x2'}) \leq (\leq_{R2\ x1'\ x2'});\ \bigwedge x1'\ x2'\ y.\ [\![x1' \leq_{R1} x2';\ in\text{-}dom\ (\leq_{L2\ r1\ x1'\ r1\ x2'})\ y]\!] \Longrightarrow dfro2.right\text{-}infix\ (l2_{x2'\ r1\ x1'}\ y)\ x1'\ x2' \leq dfro2.right\text{-}infix\ (l2_{x1'\ r1\ x1'}\ y)\ x1'\ x2';\ \bigwedge x1'\ x2'\ y.\ [\![x1' \leq_{R1} x2';\ in\text{-}codom\ (\leq_{L2\ r1\ x1'\ r1\ x2'})\ y]\!] \Longrightarrow (\leq_{R2\ x1'\ x2'})^{-1}\ (l2_{x2'\ r1\ x1'}\ y) \leq (\leq_{R2\ x1'\ x2'})^{-1}\ (l2_{x2'\ r1\ x2'}\ y)]\!] \Longrightarrow (tdfr.L \Rrightarrow_m tdfr.R)\ l$ and $[\![((\leq_{L1}) \Rrightarrow_m (\leq_{R1}))\ l1;\ \bigwedge x1\ x2.\ x1 \leq_{L1} x2 \Longrightarrow (\leq_{R2\ l1\ x1\ l1\ x2} \Rrightarrow_m \leq_{L2\ x1\ \eta_1\ x2})\ r2_{x1\ l1\ x2};\ \bigwedge x1\ x2.\ x1 \leq_{L1} x2 \Longrightarrow (\leq_{L2\ x1\ \eta_1\ x2}) \leq (\leq_{L2\ x1\ x2});\ \bigwedge x1\ x2\ y'.\ [\![x1 \leq_{L1} x2;\ in\text{-}codom\ (\leq_{R2\ l1\ x1\ l1\ x2})\ y']\!] \Longrightarrow (\leq_{L2\ x1\ x2})^{-1}\ (r2_{x1\ l1\ x2}\ y') \leq (\leq_{L2\ x1\ x2})^{-1}\ (r2_{x2\ l1\ x2}\ y');\ \bigwedge x1\ x2\ y'.\ [\![x1 \leq_{L1} x2;\ in\text{-}dom\ (\leq_{R2\ l1\ x1\ l1\ x2})\ y']\!] \Longrightarrow dfro1.right\text{-}infix\ (r2_{x1\ l1\ x2}\ y')\ x1\ x2 \leq dfro1.right\text{-}infix\ (r2_{x1\ l1\ x1}\ y')\ x1\ x2]\!] \Longrightarrow (tdfr.R \Rrightarrow_m tdfr.L)\ r$ to the first premises. However, this is not really helpful here. Moreover, the resulting theorem will not result in a useful lemma for the flipped instance of *transport-Dep-Fun-Rel* since $[\![((\leq_{R1}) \Rrightarrow_m (\leq_{L1}))\ r1;\ \bigwedge x1'\ x2'.\ x1' \leq_{R1} x2' \Longrightarrow (\leq_{L2\ r1\ x1'\ r1\ x2'} \Rrightarrow_m \leq_{R2\ \varepsilon_1\ x1'\ x2'})\ l2_{x2'\ r1\ x1'};\ \bigwedge x1'\ x2'.\ x1' \leq_{R1} x2' \Longrightarrow (\leq_{R2\ \varepsilon_1\ x1'\ x2'}) \leq (\leq_{R2\ x1'\ x2'});\ \bigwedge x1'\ x2'\ y.\ [\![x1' \leq_{R1} x2';\ in\text{-}dom\ (\leq_{L2\ r1\ x1'\ r1\ x2'})\ y]\!] \Longrightarrow dfro2.right\text{-}infix\ (l2_{x2'\ r1\ x1'}\ y)\ x1'\ x2' \leq dfro2.right\text{-}infix\ (l2_{x1'\ r1\ x1'}\ y)\ x1'\ x2';\ \bigwedge x1'\ x2'\ y.\ [\![x1' \leq_{R1} x2';\ in\text{-}codom\ (\leq_{L2\ r1\ x1'\ r1\ x2'})\ y]\!] \Longrightarrow (\leq_{R2\ x1'\ x2'})^{-1}\ (l2_{x2'\ r1\ x1'}\ y) \leq (\leq_{R2\ x1'\ x2'})^{-1}\ (l2_{x2'\ r1\ x2'}\ y)]\!] \Longrightarrow (tdfr.L \Rrightarrow_m tdfr.R)\ l$ and $[\![((\leq_{L1}) \Rrightarrow_m (\leq_{R1}))\ l1;\ \bigwedge x1\ x2.\ x1 \leq_{L1} x2 \Longrightarrow (\leq_{R2\ l1\ x1\ l1\ x2} \Rrightarrow_m \leq_{L2\ x1\ \eta_1\ x2})$

$r2_{x1\ l1\ x2}$; $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ \eta_1\ x2}) \le (\le_{L2\ x1\ x2})$; $\bigwedge x1\ x2\ y'.$ $[\![x1 \le_{L1} x2;\ in\text{-}codom\ (\le_{R2\ l1\ x1\ l1\ x2})\ y']\!] \Longrightarrow (\le_{L2\ x1\ x2})^{-1}\ (r2_{x1\ l1\ x2}\ y')$ $\le (\le_{L2\ x1\ x2})^{-1}\ (r2_{x2\ l1\ x2}\ y')$; $\bigwedge x1\ x2\ y'.\ [\![x1 \le_{L1} x2;\ in\text{-}dom\ (\le_{R2\ l1\ x1\ l1\ x2})$ $y']\!] \Longrightarrow dfro1.right\text{-}infix\ (r2_{x1\ l1\ x2}\ y')\ x1\ x2 \le dfro1.right\text{-}infix\ (r2_{x1\ l1\ x1}$ $y')\ x1\ x2]\!] \Longrightarrow (tdfr.R \Rrightarrow_m tdfr.L)\ r$ are not flipped dual but only flipped-inversed dual.

end

Monotone Function Relator **context** *transport-Mono-Fun-Rel*
begin

lemma *half-galois-prop-left-left-rightI*:
  assumes $((\le_{R1}) \Rrightarrow_m (\le_{L1}))\ r1$
  and $((\le_{L1})\ {}_h\trianglelefteq\ (\le_{R1}))\ l1\ r1$
  and $reflexive\text{-}on\ (in\text{-}dom\ (\le_{R1}))\ (\le_{R1})$
  and $((\le_{L2}) \Rrightarrow_m (\le_{R2}))\ l2$
  and $((\le_{L2})\ {}_h\trianglelefteq\ (\le_{R2}))\ l2\ r2$
  and $transitive\ (\le_{R2})$
  shows $((\le_L)\ {}_h\trianglelefteq\ (\le_R))\ l\ r$
  ⟨*proof*⟩

interpretation *flip* : *transport-Mono-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2* ⟨*proof*⟩

lemma *half-galois-prop-right-left-rightI*:
  assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1$
  and $((\le_{L1}) \trianglelefteq_h (\le_{R1}))\ l1\ r1$
  and $reflexive\text{-}on\ (in\text{-}codom\ (\le_{L1}))\ (\le_{L1})$
  and $((\le_{R2}) \Rrightarrow_m (\le_{L2}))\ r2$
  and $((\le_{L2}) \trianglelefteq_h (\le_{R2}))\ l2\ r2$
  and $transitive\ (\le_{L2})$
  shows $((\le_L) \trianglelefteq_h (\le_R))\ l\ r$
  ⟨*proof*⟩

corollary *galois-prop-left-rightI*:
  assumes $((\le_{L1}) \dashv (\le_{R1}))\ l1\ r1$
  and $reflexive\text{-}on\ (in\text{-}codom\ (\le_{L1}))\ (\le_{L1})$
  and $reflexive\text{-}on\ (in\text{-}dom\ (\le_{R1}))\ (\le_{R1})$
  and $((\le_{L2}) \dashv (\le_{R2}))\ l2\ r2$
  and $transitive\ (\le_{L2})$
  and $transitive\ (\le_{R2})$
  shows $((\le_L) \trianglelefteq (\le_R))\ l\ r$
  ⟨*proof*⟩

end

end

2.8.4 Galois Connection

theory *Transport-Functions-Galois-Connection*
  imports
    Transport-Functions-Galois-Property
    Transport-Functions-Monotone
begin

Dependent Function Relator **context** *transport-Dep-Fun-Rel*
begin

Lemmas for Monotone Function Relator **lemma** *galois-connection-left-right-if-galois-connection-mo*
  assumes *galois-conn1*: $((\le_{L1}) \dashv (\le_{R1}))\ l1\ r1$
  and *refl-R1*: *reflexive-on* (*in-codom* $(\le_{R1})$) $(\le_{R1})$
  and *R2-le1*: $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x1')\ x2'}) \le (\le_{R2\ x1'\ x2'})$
  and *mono-l2-2*: $([x' :: in\text{-}codom\ (\le_{R1})] \Rrightarrow_m [x1\ x2 :: (\le_{L1}) \mid x2\ {}_{L1}\lessapprox x'] \Rrightarrow_m$
    $[in\text{-}field\ (\le_{L2\ x1\ (r1\ x')})] \Rrightarrow (\le_{R2\ (l1\ x1)\ x'}))\ l2$
  shows $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$
   $([in\text{-}codom\ (\le_{L2\ (r1\ x1')\ (r1\ x2')})] \Rrightarrow (\le_{R2\ x1'\ x2'}))\ (l2_{x2'\ (r1\ x1')})\ (l2_{x2'\ (r1\ x2')})$
  and $\bigwedge x.\ x \le_{L1} x \Longrightarrow$
    $([in\text{-}dom\ (\le_{L2\ x\ (\eta_1\ x)})] \Rrightarrow (\le_{R2\ (l1\ x)\ (l1\ x)}))\ (l2_{(l1\ x)\ x})\ (l2_{(l1\ x)\ (\eta_1\ x)})$
⟨*proof*⟩

lemma *galois-connection-left-right-if-galois-connection-mono-assms-leftI*:
  assumes *galois-conn1*: $((\le_{L1}) \dashv (\le_{R1}))\ l1\ r1$
  and *refl-R1*: *reflexive-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$
  and *R2-le1*: $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x1')\ x2'}) \le (\le_{R2\ x1'\ x2'})$
  and *mono-l2*: $([x1'\ x2' :: (\le_{R1})] \Rrightarrow_m [x1\ x2 :: (\le_{L1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
    $[in\text{-}field\ (\le_{L2\ x1\ (r1\ x2')})] \Rrightarrow (\le_{R2\ (l1\ x1)\ x2'}))\ l2$
  shows $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$
   $([in\text{-}dom\ (\le_{L2\ (r1\ x1')\ (r1\ x2')})] \Rrightarrow (\le_{R2\ x1'\ x2'}))\ (l2_{x1'\ (r1\ x1')})\ (l2_{x2'\ (r1\ x1')})$
  and $([x' :: in\text{-}codom\ (\le_{R1})] \Rrightarrow_m [x1\ x2 :: (\le_{L1}) \mid x2\ {}_{L1}\lessapprox x'] \Rrightarrow_m$
    $[in\text{-}field\ (\le_{L2\ x1\ (r1\ x')})] \Rrightarrow (\le_{R2\ (l1\ x1)\ x'}))\ l2$
⟨*proof*⟩

In theory, the following lemmas can be obtained by taking the flipped, inverse interpretation of the locale; however, rewriting the assumptions is more involved than simply copying and adapting above proofs.

lemma *galois-connection-left-right-if-galois-connection-mono-2-assms-rightI*:
  assumes *galois-conn1*: $((\le_{L1}) \dashv (\le_{R1}))\ l1\ r1$
  and *refl-L1*: *reflexive-on* (*in-dom* $(\le_{L1})$) $(\le_{L1})$
  and *L2-le2*: $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
  and *mono-r2-2*: $([x :: in\text{-}dom\ (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x\ {}_{L1}\lessapprox x1'] \Rrightarrow_m$
    $[in\text{-}field\ (\le_{R2\ (l1\ x)\ x2'})] \Rrightarrow (\le_{L2\ x\ (r1\ x2')}))\ r2$
  shows $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$
    $([in\text{-}dom\ (\le_{R2\ (l1\ x1)\ (l1\ x2)})] \Rrightarrow (\le_{L2\ x1\ x2}))\ (r2_{x1\ (l1\ x1)})\ (r2_{x1\ (l1\ x2)})$
  and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow$

$([\mathit{in\text{-}codom}\ (\le_{R2\ (\varepsilon_1\ x')\ x'})] \Rrightarrow (\le_{L2\ (r1\ x')\ (r1\ x')}))\ (r2_{(r1\ x')\ (\varepsilon_1\ x')})\ (r2_{(r1\ x')\ x'})$
⟨*proof*⟩

lemma *galois-connection-left-right-if-galois-connection-mono-assms-rightI*:
assumes *galois-conn1*: $((\le_{L1}) \dashv (\le_{R1}))\ l1\ r1$
and *refl-L1*: $\mathit{reflexive\text{-}on}\ (\mathit{in\text{-}field}\ (\le_{L1}))\ (\le_{L1})$
and *L2-le2*: $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
and *mono-r2*: $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
$[\mathit{in\text{-}field}\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
shows $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$
$([\mathit{in\text{-}codom}\ (\le_{R2\ (l1\ x1)\ (l1\ x2)})] \Rrightarrow (\le_{L2\ x1\ x2}))\ (r2_{x1\ (l1\ x2)})\ (r2_{x2\ (l1\ x2)})$
and $([x :: \mathit{in\text{-}dom}\ (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x\ {}_{L1}\lessapprox x1'] \Rrightarrow_m$
$[\mathit{in\text{-}field}\ (\le_{R2\ (l1\ x)\ x2'})] \Rrightarrow (\le_{L2\ x\ (r1\ x2')}))\ r2$
⟨*proof*⟩

end

Monotone Dependent Function Relator **context** *transport-Mono-Dep-Fun-Rel*
begin

interpretation *flip* : *transport-Mono-Dep-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2* ⟨*proof*⟩

lemma *galois-connection-left-rightI*:
assumes $(\mathit{tdfr.L} \Rrightarrow_m \mathit{tdfr.R})\ l$ **and** $(\mathit{tdfr.R} \Rrightarrow_m \mathit{tdfr.L})\ r$
and $((\le_{L1}) \dashv (\le_{R1}))\ l1\ r1$
and $\mathit{reflexive\text{-}on}\ (\mathit{in\text{-}codom}\ (\le_{L1}))\ (\le_{L1})$
and $\mathit{reflexive\text{-}on}\ (\mathit{in\text{-}dom}\ (\le_{R1}))\ (\le_{R1})$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \trianglelefteq (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x2\ x2}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow (\le_{L2\ x\ (\eta_1\ x)}) \le (\le_{L2\ x\ x})$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x')\ x'}) \le (\le_{R2\ x'\ x'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x1'\ x1'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow$
$([\mathit{in\text{-}dom}\ (\le_{L2\ x\ (\eta_1\ x)})] \Rrightarrow (\le_{R2\ (l1\ x)\ (l1\ x)}))\ (l2_{(l1\ x)\ x})\ (l2_{(l1\ x)\ (\eta_1\ x)})$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow$
$([\mathit{in\text{-}codom}\ (\le_{R2\ (\varepsilon_1\ x')\ x'})] \Rrightarrow (\le_{L2\ (r1\ x')\ (r1\ x')}))\ (r2_{(r1\ x')\ (\varepsilon_1\ x')})\ (r2_{(r1\ x')\ x'})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow \mathit{transitive}\ (\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow \mathit{transitive}\ (\le_{R2\ x1'\ x2'})$
shows $((\le_L) \dashv (\le_R))\ l\ r$
⟨*proof*⟩

lemma *galois-connection-left-rightI'*:
assumes $((\le_{L1}) \dashv (\le_{R1}))\ l1\ r1$
and $\mathit{reflexive\text{-}on}\ (\mathit{in\text{-}codom}\ (\le_{L1}))\ (\le_{L1})$
and $\mathit{reflexive\text{-}on}\ (\mathit{in\text{-}dom}\ (\le_{R1}))\ (\le_{R1})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$
$((\le_{L2\ (r1\ x1')\ (r1\ x2')}) \Rrightarrow_m (\le_{R2\ (\varepsilon_1\ x1')\ x2'}))\ (l2_{x2'\ (r1\ x1')})$

and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow ((\le_{R2\ (l1\ x1)\ (l1\ x2)}) \Rrightarrow_m (\le_{L2\ x1\ (\eta_1\ x2)}))\ (r2_{x1\ (l1\ x2)})$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \trianglelefteq (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x2\ x2}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x1')\ x2'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x1'\ x1'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$
$([\textit{in-dom}\ (\le_{L2\ (r1\ x1')\ (r1\ x2')})] \Rrightarrow (\le_{R2\ x1'\ x2'}))\ (l2_{x1'\ (r1\ x1')})\ (l2_{x2'\ (r1\ x1')})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$
$([\textit{in-codom}\ (\le_{L2\ (r1\ x1')\ (r1\ x2')})] \Rrightarrow (\le_{R2\ x1'\ x2'}))\ (l2_{x2'\ (r1\ x1')})\ (l2_{x2'\ (r1\ x2')})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow$
$([\textit{in-dom}\ (\le_{L2\ x\ (\eta_1\ x)})] \Rrightarrow (\le_{R2\ (l1\ x)\ (l1\ x)}))\ (l2_{(l1\ x)\ x})\ (l2_{(l1\ x)\ (\eta_1\ x)})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$
$([\textit{in-codom}\ (\le_{R2\ (l1\ x1)\ (l1\ x2)})] \Rrightarrow (\le_{L2\ x1\ x2}))\ (r2_{x1\ (l1\ x2)})\ (r2_{x2\ (l1\ x2)})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$
$([\textit{in-dom}\ (\le_{R2\ (l1\ x1)\ (l1\ x2)})] \Rrightarrow (\le_{L2\ x1\ x2}))\ (r2_{x1\ (l1\ x1)})\ (r2_{x1\ (l1\ x2)})$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow$
$([\textit{in-codom}\ (\le_{R2\ (\varepsilon_1\ x')\ x'})] \Rrightarrow (\le_{L2\ (r1\ x')\ (r1\ x')}))\ (r2_{(r1\ x')\ (\varepsilon_1\ x')})\ (r2_{(r1\ x')\ x'})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow \textit{transitive}\ (\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow \textit{transitive}\ (\le_{R2\ x1'\ x2'})$
shows $((\le_L) \dashv (\le_R))\ l\ r$
$\langle \textit{proof} \rangle$

lemma *galois-connection-left-right-if-galois-connectionI*:
assumes $((\le_{L1}) \dashv (\le_{R1}))\ l1\ r1$
and *reflexive-on* (*in-codom* $(\le_{L1})$) $(\le_{L1})$
and *reflexive-on* (*in-dom* $(\le_{R1})$) $(\le_{R1})$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \dashv (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x2\ x2}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x1')\ x2'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x1'\ x1'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$
$([\textit{in-dom}\ (\le_{L2\ (r1\ x1')\ (r1\ x2')})] \Rrightarrow (\le_{R2\ x1'\ x2'}))\ (l2_{x1'\ (r1\ x1')})\ (l2_{x2'\ (r1\ x1')})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$
$([\textit{in-codom}\ (\le_{L2\ (r1\ x1')\ (r1\ x2')})] \Rrightarrow (\le_{R2\ x1'\ x2'}))\ (l2_{x2'\ (r1\ x1')})\ (l2_{x2'\ (r1\ x2')})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow$
$([\textit{in-dom}\ (\le_{L2\ x\ (\eta_1\ x)})] \Rrightarrow (\le_{R2\ (l1\ x)\ (l1\ x)}))\ (l2_{(l1\ x)\ x})\ (l2_{(l1\ x)\ (\eta_1\ x)})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$
$([\textit{in-codom}\ (\le_{R2\ (l1\ x1)\ (l1\ x2)})] \Rrightarrow (\le_{L2\ x1\ x2}))\ (r2_{x1\ (l1\ x2)})\ (r2_{x2\ (l1\ x2)})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$
$([\textit{in-dom}\ (\le_{R2\ (l1\ x1)\ (l1\ x2)})] \Rrightarrow (\le_{L2\ x1\ x2}))\ (r2_{x1\ (l1\ x1)})\ (r2_{x1\ (l1\ x2)})$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow$
$([\textit{in-codom}\ (\le_{R2\ (\varepsilon_1\ x')\ x'})] \Rrightarrow (\le_{L2\ (r1\ x')\ (r1\ x')}))\ (r2_{(r1\ x')\ (\varepsilon_1\ x')})\ (r2_{(r1\ x')\ x'})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow \textit{transitive}\ (\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow \textit{transitive}\ (\le_{R2\ x1'\ x2'})$

shows $((\le_{L}) \dashv (\le_{R}))\ l\ r$
$\langle proof \rangle$

corollary *galois-connection-left-right-if-galois-connectionI'*:
assumes $((\le_{L1}) \dashv (\le_{R1}))\ l1\ r1$
and *reflexive-on* $($*in-field* $(\le_{L1}))\ (\le_{L1})$
and *reflexive-on* $($*in-field* $(\le_{R1}))\ (\le_{R1})$
and $\bigwedge x\ x'.\ x\ _{L1}\lessapprox\ x' \Longrightarrow$
$((\le_{L2\ x\ (r1\ x')}) \dashv (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $\bigwedge x1\ x2.\ x1\ \le_{L1}\ x2 \Longrightarrow (\le_{L2\ x2\ x2}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1\ \le_{L1}\ x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1}\ x2' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x1')\ x2'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1}\ x2' \Longrightarrow (\le_{R2\ x1'\ x1'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1}\ x2' \Longrightarrow$
$([$*in-dom* $(\le_{L2\ (r1\ x1')\ (r1\ x2')})] \Rrightarrow (\le_{R2\ x1'\ x2'}))\ (l2_{x1'\ (r1\ x1')})\ (l2_{x2'\ (r1\ x1')})$
and $([x' ::$ *in-codom* $(\le_{R1})] \Rrightarrow_m [x1\ x2 :: (\le_{L1})\ |\ x2\ _{L1}\lessapprox\ x'] \Rrightarrow_m$
$[$*in-field* $(\le_{L2\ x1\ (r1\ x')})] \Rrightarrow (\le_{R2\ (l1\ x1)\ x'}))\ l2$
and $\bigwedge x1\ x2.\ x1\ \le_{L1}\ x2 \Longrightarrow$
$([$*in-codom* $(\le_{R2\ (l1\ x1)\ (l1\ x2)})] \Rrightarrow (\le_{L2\ x1\ x2}))\ (r2_{x1\ (l1\ x2)})\ (r2_{x2\ (l1\ x2)})$
and $([x ::$ *in-dom* $(\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1})\ |\ x\ _{L1}\lessapprox\ x1'] \Rrightarrow_m$
$[$*in-field* $(\le_{R2\ (l1\ x)\ x2'})] \Rrightarrow (\le_{L2\ x\ (r1\ x2')}))\ r2$
and $\bigwedge x1\ x2.\ x1\ \le_{L1}\ x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1}\ x2' \Longrightarrow$ *transitive* $(\le_{R2\ x1'\ x2'})$
shows $((\le_{L}) \dashv (\le_{R}))\ l\ r$
$\langle proof \rangle$

corollary *galois-connection-left-right-if-mono-if-galois-connectionI*:
assumes $((\le_{L1}) \dashv (\le_{R1}))\ l1\ r1$
and *reflexive-on* $($*in-field* $(\le_{L1}))\ (\le_{L1})$
and *reflexive-on* $($*in-field* $(\le_{R1}))\ (\le_{R1})$
and $\bigwedge x\ x'.\ x\ _{L1}\lessapprox\ x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \dashv (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $\bigwedge x1\ x2.\ x1\ \le_{L1}\ x2 \Longrightarrow (\le_{L2\ x2\ x2}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1\ \le_{L1}\ x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1}\ x2' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x1')\ x2'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1}\ x2' \Longrightarrow (\le_{R2\ x1'\ x1'}) \le (\le_{R2\ x1'\ x2'})$
and $([x1'\ x2' :: (\le_{R1})] \Rrightarrow_m [x1\ x2 :: (\le_{L1})\ |\ x2\ _{L1}\lessapprox\ x1'] \Rrightarrow$
$[$*in-field* $(\le_{L2\ x1\ (r1\ x2')})] \Rrightarrow (\le_{R2\ (l1\ x1)\ x2'}))\ l2$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1})\ |\ x2\ _{L1}\lessapprox\ x1'] \Rrightarrow$
$[$*in-field* $(\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
and $\bigwedge x1\ x2.\ x1\ \le_{L1}\ x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1}\ x2' \Longrightarrow$ *transitive* $(\le_{R2\ x1'\ x2'})$
shows $((\le_{L}) \dashv (\le_{R}))\ l\ r$
$\langle proof \rangle$

corollary *galois-connection-left-right-if-mono-if-galois-connectionI'*:
assumes $((\le_{L1}) \dashv (\le_{R1}))\ l1\ r1$
and *reflexive-on* $($*in-field* $(\le_{L1}))\ (\le_{L1})$

and *reflexive-on* (*in-field* $(\leq_{R1})$) $(\leq_{R1})$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow ((\leq_{L2\ x\ (r1\ x')}) \dashv (\leq_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $([-\ x2 :: (\leq_{L1})] \Rrightarrow_m [x3\ x4 :: (\leq_{L1}) \mid (x2 \leq_{L1} x3 \wedge x4 \leq_{L1} \eta_1\ x3)] \Rrightarrow (\geq))$ $L2$
and $([x1'\ x2' :: (\leq_{R1}) \mid \varepsilon_1\ x2' \leq_{R1} x1'] \Rrightarrow_m [x3'\ - :: (\leq_{R1}) \mid x2' \leq_{R1} x3'] \Rrightarrow (\leq))\ R2$
and $([x1'\ x2' :: (\leq_{R1})] \Rrightarrow_m [x1\ x2 :: (\leq_{L1}) \mid x2\ {}_{L1}\lessapprox\ x1'] \Rrightarrow$
$[\textit{in-field}\ (\leq_{L2\ x1\ (r1\ x2')})] \Rrightarrow (\leq_{R2\ (l1\ x1)\ x2'}))\ l2$
and $([x1\ x2 :: (\leq_{L1})] \Rrightarrow_m [x1'\ x2' :: (\leq_{R1}) \mid x2\ {}_{L1}\lessapprox\ x1'] \Rrightarrow$
$[\textit{in-field}\ (\leq_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\leq_{L2\ x1\ (r1\ x2')}))\ r2$
and $\bigwedge x1\ x2.\ x1 \leq_{L1} x2 \Longrightarrow$ *transitive* $(\leq_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \leq_{R1} x2' \Longrightarrow$ *transitive* $(\leq_{R2\ x1'\ x2'})$
shows $((\leq_L) \dashv (\leq_R))\ l\ r$
⟨*proof*⟩

end

Monotone Function Relator **context** *transport-Mono-Fun-Rel*
begin

interpretation *flip* : *transport-Mono-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2* ⟨*proof*⟩

lemma *galois-connection-left-rightI*:
assumes $((\leq_{L1}) \dashv (\leq_{R1}))\ l1\ r1$
and *reflexive-on* (*in-codom* $(\leq_{L1})$) $(\leq_{L1})$
and *reflexive-on* (*in-dom* $(\leq_{R1})$) $(\leq_{R1})$
and $((\leq_{L2}) \dashv (\leq_{R2}))\ l2\ r2$
and *transitive* $(\leq_{L2})$
and *transitive* $(\leq_{R2})$
shows $((\leq_L) \dashv (\leq_R))\ l\ r$
⟨*proof*⟩

end

end

2.8.5 Basic Order Properties

theory *Transport-Functions-Order-Base*
imports
Transport-Functions-Base
begin

Dependent Function Relator **context** *hom-Dep-Fun-Rel-orders*
begin

lemma *reflexive-on-in-domI*:
assumes *refl-L*: *reflexive-on* (*in-codom* $(\leq_L)$) $(\leq_L)$

and *R-le-R-if-L*: $\bigwedge x1\ x2.\ x1 \le_L x2 \Longrightarrow (\le_{R\ x2\ x2}) \le (\le_{R\ x1\ x2})$
and *pequiv-R*: $\bigwedge x1\ x2.\ x1 \le_L x2 \Longrightarrow$ *partial-equivalence-rel* $(\le_{R\ x1\ x2})$
shows *reflexive-on* (*in-dom DFR*) *DFR*
⟨*proof*⟩

lemma *reflexive-on-in-codomI*:
assumes *refl-L*: *reflexive-on* (*in-dom* $(\le_L)$) $(\le_L)$
and *R-le-R-if-L*: $\bigwedge x1\ x2.\ x1 \le_L x2 \Longrightarrow (\le_{R\ x1\ x1}) \le (\le_{R\ x1\ x2})$
and *pequiv-R*: $\bigwedge x1\ x2.\ x1 \le_L x2 \Longrightarrow$ *partial-equivalence-rel* $(\le_{R\ x1\ x2})$
shows *reflexive-on* (*in-codom DFR*) *DFR*
⟨*proof*⟩

corollary *reflexive-on-in-fieldI*:
assumes *reflexive-on* (*in-field* $(\le_L)$) $(\le_L)$
and $\bigwedge x1\ x2.\ x1 \le_L x2 \Longrightarrow (\le_{R\ x2\ x2}) \le (\le_{R\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_L x2 \Longrightarrow (\le_{R\ x1\ x1}) \le (\le_{R\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_L x2 \Longrightarrow$ *partial-equivalence-rel* $(\le_{R\ x1\ x2})$
shows *reflexive-on* (*in-field DFR*) *DFR*
⟨*proof*⟩

lemma *transitiveI*:
assumes *refl-L*: *reflexive-on* (*in-dom* $(\le_L)$) $(\le_L)$
and *R-le-R-if-L*: $\bigwedge x1\ x2.\ x1 \le_L x2 \Longrightarrow (\le_{R\ x1\ x1}) \le (\le_{R\ x1\ x2})$
and *trans*: $\bigwedge x1\ x2.\ x1 \le_L x2 \Longrightarrow$ *transitive* $(\le_{R\ x1\ x2})$
shows *transitive DFR*
⟨*proof*⟩

lemma *transitiveI'*:
assumes *refl-L*: *reflexive-on* (*in-codom* $(\le_L)$) $(\le_L)$
and *R-le-R-if-L*: $\bigwedge x1\ x2.\ x1 \le_L x2 \Longrightarrow (\le_{R\ x2\ x2}) \le (\le_{R\ x1\ x2})$
and *trans*: $\bigwedge x1\ x2.\ x1 \le_L x2 \Longrightarrow$ *transitive* $(\le_{R\ x1\ x2})$
shows *transitive DFR*
⟨*proof*⟩

lemma *preorder-on-in-fieldI*:
assumes *reflexive-on* (*in-field* $(\le_L)$) $(\le_L)$
and $\bigwedge x1\ x2.\ x1 \le_L x2 \Longrightarrow (\le_{R\ x2\ x2}) \le (\le_{R\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_L x2 \Longrightarrow (\le_{R\ x1\ x1}) \le (\le_{R\ x1\ x2})$
and *pequiv-R*: $\bigwedge x1\ x2.\ x1 \le_L x2 \Longrightarrow$ *partial-equivalence-rel* $(\le_{R\ x1\ x2})$
shows *preorder-on* (*in-field DFR*) *DFR*
⟨*proof*⟩

lemma *symmetricI*:
assumes *sym-L*: *symmetric* $(\le_L)$
and *R-le-R-if-L*: $\bigwedge x1\ x2.\ x1 \le_L x2 \Longrightarrow (\le_{R\ x1\ x2}) \le (\le_{R\ x2\ x1})$
and *sym-R*: $\bigwedge x1\ x2.\ x1 \le_L x2 \Longrightarrow$ *symmetric* $(\le_{R\ x1\ x2})$
shows *symmetric DFR*
⟨*proof*⟩

corollary *partial-equivalence-relI*:
  assumes *reflexive-on* (*in-field* $(\le_L)$) $(\le_L)$
  and *sym-L*: *symmetric* $(\le_L)$
  and *mono-R*: $([x1\ x2 :: (\le_L)] \Rrightarrow_m [x3\ x4 :: (\le_L) \mid x1 \le_L x3] \Rrightarrow (\le))\ R$
  and $\bigwedge x1\ x2.\ x1 \le_L x2 \Longrightarrow$ *partial-equivalence-rel* $(\le_{R\ x1\ x2})$
  shows *partial-equivalence-rel DFR*
⟨*proof*⟩

end

context *transport-Dep-Fun-Rel*
begin

lemmas *reflexive-on-in-field-leftI* = *dfro1.reflexive-on-in-fieldI*
  [*folded left-rel-eq-Dep-Fun-Rel*]
lemmas *transitive-leftI* = *dfro1.transitiveI*[*folded left-rel-eq-Dep-Fun-Rel*]
lemmas *transitive-leftI'* = *dfro1.transitiveI'*[*folded left-rel-eq-Dep-Fun-Rel*]
lemmas *preorder-on-in-field-leftI* = *dfro1.preorder-on-in-fieldI*
  [*folded left-rel-eq-Dep-Fun-Rel*]
lemmas *symmetric-leftI* = *dfro1.symmetricI*[*folded left-rel-eq-Dep-Fun-Rel*]
lemmas *partial-equivalence-rel-leftI* = *dfro1.partial-equivalence-relI*
  [*folded left-rel-eq-Dep-Fun-Rel*]

Introduction Rules for Assumptions **lemma** *transitive-left2-if-transitive-left2-if-left-GaloisI*:
  assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1$
  and $((\le_{L1}) \trianglelefteq_h (\le_{R1}))\ l1\ r1$
  and *L2-eq*: $(\le_{L2\ x1\ x2}) = (\le_{L2\ x1\ (\eta_1\ x2)})$
  and $\bigwedge x\ x'.\ x\ _{L1}\lessapprox x' \Longrightarrow$ *transitive* $(\le_{L2\ x\ (r1\ x')})$
  and $x1 \le_{L1} x2$
  shows *transitive* $(\le_{L2\ x1\ x2})$
  ⟨*proof*⟩

lemma *symmetric-left2-if-symmetric-left2-if-left-GaloisI*:
  assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1$
  and $((\le_{L1}) \trianglelefteq_h (\le_{R1}))\ l1\ r1$
  and *L2-eq*: $(\le_{L2\ x1\ x2}) = (\le_{L2\ x1\ (\eta_1\ x2)})$
  and $\bigwedge x\ x'.\ x\ _{L1}\lessapprox x' \Longrightarrow$ *symmetric* $(\le_{L2\ x\ (r1\ x')})$
  and $x1 \le_{L1} x2$
  shows *symmetric* $(\le_{L2\ x1\ x2})$
  ⟨*proof*⟩

lemma *partial-equivalence-rel-left2-if-partial-equivalence-rel-left2-if-left-GaloisI*:
  assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1$
  and $((\le_{L1}) \trianglelefteq_h (\le_{R1}))\ l1\ r1$
  and *L2-eq*: $(\le_{L2\ x1\ x2}) = (\le_{L2\ x1\ (\eta_1\ x2)})$
  and $\bigwedge x\ x'.\ x\ _{L1}\lessapprox x' \Longrightarrow$ *partial-equivalence-rel* $(\le_{L2\ x\ (r1\ x')})$
  and $x1 \le_{L1} x2$
  shows *partial-equivalence-rel* $(\le_{L2\ x1\ x2})$

⟨*proof*⟩

context
assumes *galois-prop*: $((\le_{L1}) \trianglelefteq (\le_{R1}))$ *l1 r1*
begin

interpretation *flip-inv* :
transport-Dep-Fun-Rel $(\ge_{R1})$ $(\ge_{L1})$ *r1 l1 flip2 R2 flip2 L2 r2 l2*
rewrites *flip-inv.t1.unit* $\equiv \varepsilon_1$
and $\bigwedge R\ x\ y.$ (*flip2* $R\ x\ y) \equiv (R\ y\ x)^{-1}$
and $\bigwedge R\ S.\ R^{-1} = S^{-1} \equiv R = S$
and $\bigwedge R\ S.\ (R^{-1} \Rrightarrow_m S^{-1}) \equiv (R \Rrightarrow_m S)$
and $\bigwedge x\ x'.\ x'\ {}_{R1}\lessapprox x \equiv x\ {}_{L1}\lessapprox x'$
and $((\ge_{R1}) \trianglelefteq_h (\ge_{L1}))$ *r1 l1* $\equiv$ *True*
and $\bigwedge R.$ *transitive* $R^{-1} \equiv$ *transitive* R
and $\bigwedge R.$ *symmetric* $R^{-1} \equiv$ *symmetric* R
and $\bigwedge R.$ *partial-equivalence-rel* $R^{-1} \equiv$ *partial-equivalence-rel* R
and $\bigwedge P.$ (*True* $\Longrightarrow P) \equiv$ *Trueprop* P
and $\bigwedge P\ Q.$ (*True* $\Longrightarrow$ *PROP* $P \Longrightarrow$ *PROP* $Q) \equiv$ (*PROP* $P \Longrightarrow$ *True* $\Longrightarrow$ *PROP* Q)
⟨*proof*⟩

lemma *transitive-right2-if-transitive-right2-if-left-GaloisI*:
assumes $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*
and $(\le_{R2\ x1\ x2}) = (\le_{R2\ (\varepsilon_1\ x1)\ x2})$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow$ *transitive* $(\le_{R2\ (l1\ x)\ x'})$
and $x1 \le_{R1} x2$
shows *transitive* $(\le_{R2\ x1\ x2})$
⟨*proof*⟩

lemma *symmetric-right2-if-symmetric-right2-if-left-GaloisI*:
assumes $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*
and $(\le_{R2\ x1\ x2}) = (\le_{R2\ (\varepsilon_1\ x1)\ x2})$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow$ *symmetric* $(\le_{R2\ (l1\ x)\ x'})$
and $x1 \le_{R1} x2$
shows *symmetric* $(\le_{R2\ x1\ x2})$
⟨*proof*⟩

lemma *partial-equivalence-rel-right2-if-partial-equivalence-rel-right2-if-left-GaloisI*:
assumes $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*
and $(\le_{R2\ x1\ x2}) = (\le_{R2\ (\varepsilon_1\ x1)\ x2})$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow$ *partial-equivalence-rel* $(\le_{R2\ (l1\ x)\ x'})$
and $x1 \le_{R1} x2$
shows *partial-equivalence-rel* $(\le_{R2\ x1\ x2})$
⟨*proof*⟩

end

lemma *transitive-left2-if-preorder-equivalenceI*:
assumes *pre-equiv1*: $((\le_{L1}) \equiv_{pre} (\le_{R1}))$ *l1 r1*
and $([x1\ x2 :: (\ge_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid x1 \le_{L1} x3] \Rrightarrow (\le))$ *L2*
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \equiv_{pre} (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $x1 \le_{L1} x2$
shows *transitive* $(\le_{L2\ x1\ x2})$
⟨*proof*⟩

lemma *symmetric-left2-if-partial-equivalence-rel-equivalenceI*:
assumes *PER-equiv1*: $((\le_{L1}) \equiv_{PER} (\le_{R1}))$ *l1 r1*
and $([x1\ x2 :: (\ge_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid x1 \le_{L1} x3] \Rrightarrow (\le))$ *L2*
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \equiv_{PER} (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $x1 \le_{L1} x2$
shows *symmetric* $(\le_{L2\ x1\ x2})$
⟨*proof*⟩

lemma *partial-equivalence-rel-left2-if-partial-equivalence-rel-equivalenceI*:
assumes *PER-equiv1*: $((\le_{L1}) \equiv_{PER} (\le_{R1}))$ *l1 r1*
and $([x1\ x2 :: (\ge_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid x1 \le_{L1} x3] \Rrightarrow (\le))$ *L2*
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \equiv_{PER} (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $x1 \le_{L1} x2$
shows *partial-equivalence-rel* $(\le_{L2\ x1\ x2})$
⟨*proof*⟩

interpretation *flip* : *transport-Dep-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2*
rewrites *flip.t1.counit* $\equiv \eta_1$ **and** *flip.t1.unit* $\equiv \varepsilon_1$
⟨*proof*⟩

lemma *transitive-right2-if-preorder-equivalenceI*:
assumes *pre-equiv1*: $((\le_{L1}) \equiv_{pre} (\le_{R1}))$ *l1 r1*
and $([x1'\ x2' :: (\ge_{R1})] \Rrightarrow_m [x3'\ x4' :: (\le_{R1}) \mid x1' \le_{R1} x3'] \Rrightarrow (\le))$ *R2*
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \equiv_{pre} (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $x1' \le_{R1} x2'$
shows *transitive* $(\le_{R2\ x1'\ x2'})$
⟨*proof*⟩

lemma *symmetric-right2-if-partial-equivalence-rel-equivalenceI*:
assumes *PER-equiv1*: $((\le_{L1}) \equiv_{PER} (\le_{R1}))$ *l1 r1*
and $([x1'\ x2' :: (\ge_{R1})] \Rrightarrow_m [x3'\ x4' :: (\le_{R1}) \mid x1' \le_{R1} x3'] \Rrightarrow (\le))$ *R2*
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \equiv_{PER} (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $x1' \le_{R1} x2'$
shows *symmetric* $(\le_{R2\ x1'\ x2'})$
⟨*proof*⟩

lemma *partial-equivalence-rel-right2-if-partial-equivalence-rel-equivalenceI*:
assumes *PER-equiv1*: $((\le_{L1}) \equiv_{PER} (\le_{R1}))$ *l1 r1*
and $([x1'\ x2' :: (\ge_{R1})] \Rrightarrow_m [x3'\ x4' :: (\le_{R1}) \mid x1' \le_{R1} x3'] \Rrightarrow (\le))$ *R2*
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \equiv_{PER} (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$

and $x1' \le_{R1} x2'$
shows *partial-equivalence-rel* $(\le_{R2\ x1'\ x2'})$
⟨*proof*⟩

end

Function Relator **context** *transport-Fun-Rel*
begin

lemma *reflexive-on-in-field-leftI*:
assumes *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and *partial-equivalence-rel* $(\le_{L2})$
shows *reflexive-on* (*in-field* $(\le_{L})$) $(\le_{L})$
⟨*proof*⟩

lemma *transitive-leftI*:
assumes *reflexive-on* (*in-dom* $(\le_{L1})$) $(\le_{L1})$
and *transitive* $(\le_{L2})$
shows *transitive* $(\le_{L})$
⟨*proof*⟩

lemma *transitive-leftI'*:
assumes *reflexive-on* (*in-codom* $(\le_{L1})$) $(\le_{L1})$
and *transitive* $(\le_{L2})$
shows *transitive* $(\le_{L})$
⟨*proof*⟩

lemma *preorder-on-in-field-leftI*:
assumes *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and *partial-equivalence-rel* $(\le_{L2})$
shows *preorder-on* (*in-field* $(\le_{L})$) $(\le_{L})$
⟨*proof*⟩

lemma *symmetric-leftI*:
assumes *symmetric* $(\le_{L1})$
and *symmetric* $(\le_{L2})$
shows *symmetric* $(\le_{L})$
⟨*proof*⟩

corollary *partial-equivalence-rel-leftI*:
assumes *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and *symmetric* $(\le_{L1})$
and *partial-equivalence-rel* $(\le_{L2})$
shows *partial-equivalence-rel* $(\le_{L})$
⟨*proof*⟩

end

Monotone Dependent Function Relator **context** *transport-Mono-Dep-Fun-Rel*

begin

lemmas *reflexive-on-in-field-leftI* = *Refl-Rel-reflexive-on-in-field*[*of tdfr.L*,
folded left-rel-eq-tdfr-left-Refl-Rel]
lemmas *transitive-leftI* = *Refl-Rel-transitiveI*
[*of tdfr.L*, *folded left-rel-eq-tdfr-left-Refl-Rel*]
lemmas *preorder-on-in-field-leftI* = *Refl-Rel-preorder-on-in-fieldI*[*of tdfr.L*,
folded left-rel-eq-tdfr-left-Refl-Rel]
lemmas *symmetric-leftI* = *Refl-Rel-symmetricI*[*of tdfr.L*,
OF tdfr.symmetric-leftI, *folded left-rel-eq-tdfr-left-Refl-Rel*]
lemmas *partial-equivalence-rel-leftI* = *Refl-Rel-partial-equivalence-relI*[*of tdfr.L*,
OF tdfr.partial-equivalence-rel-leftI, *folded left-rel-eq-tdfr-left-Refl-Rel*]

end

Monotone Function Relator **context** *transport-Mono-Fun-Rel*
begin

lemma *symmetric-leftI*:
assumes *symmetric* $(\leq_{L1})$
and *symmetric* $(\leq_{L2})$
shows *symmetric* $(\leq_{L})$
⟨*proof*⟩

lemma *partial-equivalence-rel-leftI*:
assumes *reflexive-on* (*in-field* $(\leq_{L1})$) $(\leq_{L1})$
and *symmetric* $(\leq_{L1})$
and *partial-equivalence-rel* $(\leq_{L2})$
shows *partial-equivalence-rel* $(\leq_{L})$
⟨*proof*⟩

end

end

2.8.6 Galois Equivalence

theory *Transport-Functions-Galois-Equivalence*
imports
Transport-Functions-Galois-Connection
Transport-Functions-Order-Base
begin

Dependent Function Relator **context** *transport-Dep-Fun-Rel*
begin

Lemmas for Monotone Function Relator **lemma** *flip-half-galois-prop-left2-if-half-galois-prop-left2-i*
assumes $((\leq_{L1}) \Rightarrow_m (\leq_{R1}))$ *l1*

and $((\le_{L1}) \trianglelefteq_h (\le_{R1}))$ *l1 r1*
and *half-galois-prop-left2*: $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow$
$((\le_{R2\ (l1\ x)\ x'})\ {}_h\trianglelefteq\ (\le_{L2\ x\ (r1\ x')}))\ (r2_{x\ x'})\ (l2_{x'\ x})$
and $(\le_{L2\ (\eta_1\ x)\ x}) = (\le_{L2\ x\ x})$
and $(\le_{L2\ x\ (\eta_1\ x)}) = (\le_{L2\ x\ x})$
and $x \le_{L1} x$
shows $((\le_{R2\ (l1\ x)\ (l1\ x)})\ {}_h\trianglelefteq\ (\le_{L2\ (\eta_1\ x)\ x}))\ (r2_{x\ (l1\ x)})\ (l2_{(l1\ x)\ x})$
⟨*proof*⟩

lemma *flip-half-galois-prop-right2-if-half-galois-prop-right2-if-GaloisI*:
assumes $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*
and *half-galois-prop-right2*: $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow$
$((\le_{R2\ (l1\ x)\ x'}) \trianglelefteq_h (\le_{L2\ x\ (r1\ x')}))\ (r2_{x\ x'})\ (l2_{x'\ x})$
and $(\le_{R2\ (\varepsilon_1\ x')\ x'}) = (\le_{R2\ x'\ x'})$
and $(\le_{R2\ x'\ (\varepsilon_1\ x')}) = (\le_{R2\ x'\ x'})$
and $x' \le_{R1} x'$
shows $((\le_{R2\ x'\ (\varepsilon_1\ x')}) \trianglelefteq_h (\le_{L2\ (r1\ x')\ (r1\ x')}))\ (r2_{(r1\ x')\ x'})\ (l2_{x'\ (r1\ x')})$
⟨*proof*⟩

interpretation *flip* : *transport-Dep-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2*
rewrites *flip.t1.counit* $\equiv \eta_1$ **and** *flip.t1.unit* $\equiv \varepsilon_1$
⟨*proof*⟩

lemma *galois-equivalence-if-mono-if-galois-equivalence-mono-assms-leftI*:
assumes *galois-equiv1*: $((\le_{L1}) \equiv_G (\le_{R1}))$ *l1 r1*
and *preorder-L1*: *preorder-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and *mono-L2*: $([x1\ x2 :: (\ge_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid x1 \le_{L1} x3] \Rrightarrow (\le))$ *L2*
shows $([x1\ x2 :: (\le_{L1}) \mid \eta_1\ x2 \le_{L1} x1] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid x2 \le_{L1} x3] \Rrightarrow (\le))$ *L2* (**is** *?goal1*)
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid (x2 \le_{L1} x3 \wedge x4 \le_{L1} \eta_1\ x3)] \Rrightarrow (\ge))$ *L2* (**is** *?goal2*)
⟨*proof*⟩

lemma *galois-equivalence-if-mono-if-galois-equivalence-Dep-Fun-Rel-pred-assm-leftI*:
assumes *galois-equiv1*: $((\le_{L1}) \equiv_G (\le_{R1}))$ *l1 r1*
and *refl-L1*: *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and *refl-R1*: *reflexive-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$
and *mono-L2*: $([x1\ x2 :: (\ge_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid x1 \le_{L1} x3] \Rrightarrow (\le))$ *L2*
and *mono-R2*: $([x1'\ x2' :: (\ge_{R1})] \Rrightarrow_m [x3'\ x4' :: (\le_{R1}) \mid x1' \le_{R1} x3'] \Rrightarrow (\le))$ *R2*
and *mono-l2*: $([x1'\ x2' :: (\le_{R1})] \Rrightarrow_m [x1\ x2 :: (\le_{L1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
$[\textit{in-field}\ (\le_{L2\ x1\ (r1\ x2')})] \Rrightarrow (\le_{R2\ (l1\ x1)\ x2'}))$ *l2*
and $x \le_{L1} x$
shows $([\textit{in-codom}\ (\le_{L2\ (\eta_1\ x)\ x})] \Rrightarrow (\le_{R2\ (l1\ x)\ (l1\ x)}))\ (l2_{(l1\ x)\ (\eta_1\ x)})\ (l2_{(l1\ x)\ x})$
⟨*proof*⟩

lemma *galois-equivalence-if-mono-if-galois-equivalence-Dep-Fun-Rel-pred-assm-right*:

assumes *galois-equiv1*: $((\le_{L1}) \equiv_G (\le_{R1}))$ *l1 r1*
and *refl-R1*: *reflexive-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$
and *mono-L2*: $([x1\ x2 :: (\ge_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid x1 \le_{L1} x3] \Rrightarrow (\le))$ *L2*
and *mono-R2*: $([x1'\ x2' :: (\ge_{R1})] \Rrightarrow_m [x3'\ x4' :: (\le_{R1}) \mid x1' \le_{R1} x3'] \Rrightarrow (\le))$
R2
and *mono-r2*: $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
$[\textit{in-field}\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))$ *r2*
and $x' \le_{R1} x'$
shows $([\textit{in-dom}\ (\le_{R2\ x'\ (\varepsilon_1\ x')})] \Rrightarrow (\le_{L2\ (r1\ x')\ (r1\ x')}))\ (r2_{(r1\ x')\ x'})\ (r2_{(r1\ x')\ (\varepsilon_1\ x')})$
⟨*proof*⟩

end

Monotone Dependent Function Relator **context** *transport-Mono-Dep-Fun-Rel*
begin

context
begin

interpretation *flip* : *transport-Mono-Dep-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2*
rewrites *flip.t1.counit* $\equiv \eta_1$ **and** *flip.t1.unit* $\equiv \varepsilon_1$
⟨*proof*⟩

lemma *galois-equivalence-if-galois-equivalenceI*:
assumes *galois-equiv1*: $((\le_{L1}) \equiv_G (\le_{R1}))$ *l1 r1*
and *refl-L1*: *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and *reflexive-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$
and *galois-equiv2*: $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow$
$((\le_{L2\ x\ (r1\ x')}) \equiv_G (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x2\ x2}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow (\le_{L2\ (\eta_1\ x)\ x}) \le (\le_{L2\ x\ x})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ x1}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x2'\ x2'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x1')\ x2'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x1'\ x1'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow (\le_{R2\ x'\ (\varepsilon_1\ x')}) \le (\le_{R2\ x'\ x'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$
$([\textit{in-dom}\ (\le_{L2\ (r1\ x1')\ (r1\ x2')})] \Rrightarrow (\le_{R2\ x1'\ x2'}))\ (l2_{x1'\ (r1\ x1')})\ (l2_{x2'\ (r1\ x1')})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$
$([\textit{in-codom}\ (\le_{L2\ (r1\ x1')\ (r1\ x2')})] \Rrightarrow (\le_{R2\ x1'\ x2'}))\ (l2_{x2'\ (r1\ x1')})\ (l2_{x2'\ (r1\ x2')})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow$
$([\textit{in-dom}\ (\le_{L2\ x\ (\eta_1\ x)})] \Rrightarrow (\le_{R2\ (l1\ x)\ (l1\ x)}))\ (l2_{(l1\ x)\ x})\ (l2_{(l1\ x)\ (\eta_1\ x)})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow$
$([\textit{in-codom}\ (\le_{L2\ (\eta_1\ x)\ x})] \Rrightarrow (\le_{R2\ (l1\ x)\ (l1\ x)}))\ (l2_{(l1\ x)\ (\eta_1\ x)})\ (l2_{(l1\ x)\ x})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$
$([\textit{in-codom}\ (\le_{R2\ (l1\ x1)\ (l1\ x2)})] \Rrightarrow (\le_{L2\ x1\ x2}))\ (r2_{x1\ (l1\ x2)})\ (r2_{x2\ (l1\ x2)})$

and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$
$([in\text{-}dom\ (\le_{R2\ (l1\ x1)\ (l1\ x2)})] \Rrightarrow (\le_{L2\ x1\ x2}))\ (r2_{x1\ (l1\ x1)})\ (r2_{x1\ (l1\ x2)})$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow$
$([in\text{-}codom\ (\le_{R2\ (\varepsilon_1\ x')\ x'})] \Rrightarrow (\le_{L2\ (r1\ x')\ (r1\ x')}))\ (r2_{(r1\ x')\ (\varepsilon_1\ x')})\ (r2_{(r1\ x')\ x'})$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow$
$([in\text{-}dom\ (\le_{R2\ x'\ (\varepsilon_1\ x')})] \Rrightarrow (\le_{L2\ (r1\ x')\ (r1\ x')}))\ (r2_{(r1\ x')\ x'})\ (r2_{(r1\ x')\ (\varepsilon_1\ x')})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow transitive\ (\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow transitive\ (\le_{R2\ x1'\ x2'})$
shows $((\le_L) \equiv_G (\le_R))\ l\ r$
⟨*proof*⟩

corollary *galois-equivalence-if-galois-equivalenceI′*:
assumes $((\le_{L1}) \equiv_G (\le_{R1}))\ l1\ r1$
and *reflexive-on* $(in\text{-}field\ (\le_{L1}))\ (\le_{L1})$
and *reflexive-on* $(in\text{-}field\ (\le_{R1}))\ (\le_{R1})$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \equiv_G (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x2\ x2}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow (\le_{L2\ (\eta_1\ x)\ x}) \le (\le_{L2\ x\ x})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ x1}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x2'\ x2'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x1')\ x2'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x1'\ x1'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow (\le_{R2\ x'\ (\varepsilon_1\ x')}) \le (\le_{R2\ x'\ x'})$
and $([x1'\ x2' :: (\le_{R1})] \Rrightarrow_m [x1\ x2 :: (\le_{L1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
$[in\text{-}field\ (\le_{L2\ x1\ (r1\ x2')})] \Rrightarrow (\le_{R2\ (l1\ x1)\ x2'}))\ l2$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow$
$([in\text{-}codom\ (\le_{L2\ (\eta_1\ x)\ x})] \Rrightarrow (\le_{R2\ (l1\ x)\ (l1\ x)}))\ (l2_{(l1\ x)\ (\eta_1\ x)})\ (l2_{(l1\ x)\ x})$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
$[in\text{-}field\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow$
$([in\text{-}dom\ (\le_{R2\ x'\ (\varepsilon_1\ x')})] \Rrightarrow (\le_{L2\ (r1\ x')\ (r1\ x')}))\ (r2_{(r1\ x')\ x'})\ (r2_{(r1\ x')\ (\varepsilon_1\ x')})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow transitive\ (\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow transitive\ (\le_{R2\ x1'\ x2'})$
shows $((\le_L) \equiv_G (\le_R))\ l\ r$
⟨*proof*⟩

corollary *galois-equivalence-if-mono-if-galois-equivalenceI*:
assumes $((\le_{L1}) \equiv_G (\le_{R1}))\ l1\ r1$
and *reflexive-on* $(in\text{-}field\ (\le_{L1}))\ (\le_{L1})$
and *reflexive-on* $(in\text{-}field\ (\le_{R1}))\ (\le_{R1})$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \equiv_G (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $([x1\ x2 :: (\le_{L1}) \mid \eta_1\ x2 \le_{L1} x1] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid x2 \le_{L1} x3] \Rrightarrow (\le))$
$L2$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid (x2 \le_{L1} x3 \wedge x4 \le_{L1} \eta_1\ x3)] \Rrightarrow$
$(\ge))\ L2$
and $([x1'\ x2' :: (\le_{R1}) \mid \varepsilon_1\ x2' \le_{R1} x1'] \Rrightarrow_m [x3'\ x4' :: (\le_{R1}) \mid x2' \le_{R1} x3']$

$\Rrightarrow (\le))\ R2$
and $([x1'\ x2' :: (\le_{R1})] \Rrightarrow_m [x3'\ x4' :: (\le_{R1}) \mid (x2' \le_{R1} x3' \wedge x4' \le_{R1} \varepsilon_1\ x3')]$
$\Rrightarrow (\ge))\ R2$
and $([x1'\ x2' :: (\le_{R1})] \Rrightarrow_m [x1\ x2 :: (\le_{L1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
$[in\text{-}field\ (\le_{L2\ x1\ (r1\ x2')})] \Rrightarrow (\le_{R2\ (l1\ x1)\ x2'}))\ l2$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow$
$([in\text{-}codom\ (\le_{L2\ (\eta_1\ x)\ x})] \Rrightarrow (\le_{R2\ (l1\ x)\ (l1\ x)}))\ (l2_{(l1\ x)\ (\eta_1\ x)})\ (l2_{(l1\ x)\ x})$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
$[in\text{-}field\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow$
$([in\text{-}dom\ (\le_{R2\ x'\ (\varepsilon_1\ x')})] \Rrightarrow (\le_{L2\ (r1\ x')\ (r1\ x')}))\ (r2_{(r1\ x')\ x'})\ (r2_{(r1\ x')\ (\varepsilon_1\ x')})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow transitive\ (\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow transitive\ (\le_{R2\ x1'\ x2'})$
shows $((\le_L) \equiv_G (\le_R))\ l\ r$
⟨*proof*⟩

end

interpretation *flip* : *transport-Mono-Dep-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2*
rewrites *flip.t1.counit* $\equiv \eta_1$ **and** *flip.t1.unit* $\equiv \varepsilon_1$
⟨*proof*⟩

lemma *galois-equivalence-if-mono-if-preorder-equivalenceI*:
assumes $((\le_{L1}) \equiv_{pre} (\le_{R1}))\ l1\ r1$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \equiv_G (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $([x1\ x2 :: (\ge_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid x1 \le_{L1} x3] \Rrightarrow (\le))\ L2$
and $([x1'\ x2' :: (\ge_{R1})] \Rrightarrow_m [x3'\ x4' :: (\le_{R1}) \mid x1' \le_{R1} x3'] \Rrightarrow (\le))\ R2$
and $([x1'\ x2' :: (\le_{R1})] \Rrightarrow_m [x1\ x2 :: (\le_{L1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
$[in\text{-}field\ (\le_{L2\ x1\ (r1\ x2')})] \Rrightarrow (\le_{R2\ (l1\ x1)\ x2'}))\ l2$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
$[in\text{-}field\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow transitive\ (\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow transitive\ (\le_{R2\ x1'\ x2'})$
shows $((\le_L) \equiv_G (\le_R))\ l\ r$
⟨*proof*⟩

theorem *galois-equivalence-if-mono-if-preorder-equivalenceI'*:
assumes $((\le_{L1}) \equiv_{pre} (\le_{R1}))\ l1\ r1$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \equiv_{pre} (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $([x1\ x2 :: (\ge_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid x1 \le_{L1} x3] \Rrightarrow (\le))\ L2$
and $([x1'\ x2' :: (\ge_{R1})] \Rrightarrow_m [x3'\ x4' :: (\le_{R1}) \mid x1' \le_{R1} x3'] \Rrightarrow (\le))\ R2$
and $([x1'\ x2' :: (\le_{R1})] \Rrightarrow_m [x1\ x2 :: (\le_{L1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
$[in\text{-}field\ (\le_{L2\ x1\ (r1\ x2')})] \Rrightarrow (\le_{R2\ (l1\ x1)\ x2'}))\ l2$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
$[in\text{-}field\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
shows $((\le_L) \equiv_G (\le_R))\ l\ r$
⟨*proof*⟩

end

Monotone Function Relator **context** *transport-Mono-Fun-Rel*
begin

interpretation *flip* : *transport-Mono-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2* ⟨*proof*⟩

lemma *galois-equivalenceI*:
assumes $((\le_{L1}) \equiv_G (\le_{R1}))$ *l1 r1*
and *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and *reflexive-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$
and $((\le_{L2}) \equiv_G (\le_{R2}))$ *l2 r2*
and *transitive* $(\le_{L2})$
and *transitive* $(\le_{R2})$
shows $((\le_L) \equiv_G (\le_R))$ *l r*
⟨*proof*⟩

end

end

2.8.7 Simplification of Left and Right Relations

theory *Transport-Functions-Relation-Simplifications*
imports
Transport-Functions-Order-Base
Transport-Functions-Galois-Equivalence
begin

Dependent Function Relator **context** *transport-Dep-Fun-Rel*
begin

Due to *reflexive-on* (*in-field* (*transport-Dep-Fun-Rel.L ?L1.0 ?L2.0*)) (*transport-Dep-Fun-Rel.L ?L1.0 ?L2.0*) $\Longrightarrow$ *transport-Mono-Dep-Fun-Rel.L ?L1.0 ?L2.0* = *transport-Dep-Fun-Rel.L ?L1.0 ?L2.0*, we can apply all results from *transport-Mono-Dep-Fun-Rel* to *transport-Dep-Fun-Rel* whenever $(\le_L)$ and $(\le_R)$ are reflexive.

lemma *reflexive-on-in-field-left-rel2-le-assmI*:
assumes *refl-L1*: *reflexive-on* (*in-dom* $(\le_{L1})$) $(\le_{L1})$
and *mono-L2*: $([x1 :: \top] \Rrightarrow_m [x2\ x3 :: (\le_{L1}) \mid x1 \le_{L1} x2] \Rrightarrow_m (\le))$ *L2*
and $x1 \le_{L1} x2$
shows $(\le_{L2\ x1\ x1}) \le (\le_{L2\ x1\ x2})$
⟨*proof*⟩

lemma *reflexive-on-in-field-mono-assm-left2I*:
assumes *mono-L2*: $([x1\ x2 :: (\ge_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid x1 \le_{L1} x3] \Rrightarrow (\le))$ *L2*

and *refl-L1*: *reflexive-on* (*in-dom* $(\leq_{L1})$) $(\leq_{L1})$
shows $([x1 :: \top] \Rrightarrow_m [x2\ x3 :: (\leq_{L1}) \mid x1 \leq_{L1} x2] \Rrightarrow_m (\leq))$ *L2*
⟨*proof*⟩

lemma *reflexive-on-in-field-left-if-equivalencesI*:
assumes $((\leq_{L1}) \equiv_G (\leq_{R1}))$ *l1 r1*
and *preorder-on* (*in-field* $(\leq_{L1})$) $(\leq_{L1})$
and $([x1\ x2 :: (\geq_{L1})] \Rrightarrow_m [x3\ x4 :: (\leq_{L1}) \mid x1 \leq_{L1} x3] \Rrightarrow (\leq))$ *L2*
and $\bigwedge x1\ x2.\ x1 \leq_{L1} x2 \Longrightarrow$ *partial-equivalence-rel* $(\leq_{L2\ x1\ x2})$
shows *reflexive-on* (*in-field* $(\leq_L)$) $(\leq_L)$
⟨*proof*⟩

end

Monotone Dependent Function Relator **context** *transport-Mono-Dep-Fun-Rel*
begin

lemma *left-rel-eq-tdfr-leftI*:
assumes *reflexive-on* (*in-field* $(\leq_{L1})$) $(\leq_{L1})$
and $\bigwedge x1\ x2.\ x1 \leq_{L1} x2 \Longrightarrow (\leq_{L2\ x2\ x2}) \leq (\leq_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \leq_{L1} x2 \Longrightarrow (\leq_{L2\ x1\ x1}) \leq (\leq_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \leq_{L1} x2 \Longrightarrow$ *partial-equivalence-rel* $(\leq_{L2\ x1\ x2})$
shows $(\leq_L) =$ *tdfr.L*
⟨*proof*⟩

lemma *left-rel-eq-tdfr-leftI-if-equivalencesI*:
assumes $((\leq_{L1}) \equiv_G (\leq_{R1}))$ *l1 r1*
and *preorder-on* (*in-field* $(\leq_{L1})$) $(\leq_{L1})$
and $([x1\ x2 :: (\geq_{L1})] \Rrightarrow_m [x3\ x4 :: (\leq_{L1}) \mid x1 \leq_{L1} x3] \Rrightarrow (\leq))$ *L2*
and $\bigwedge x1\ x2.\ x1 \leq_{L1} x2 \Longrightarrow$ *partial-equivalence-rel* $(\leq_{L2\ x1\ x2})$
shows $(\leq_L) =$ *tdfr.L*
⟨*proof*⟩

end

Monotone Function Relator **context** *transport-Mono-Fun-Rel*
begin

lemma *left-rel-eq-tfr-leftI*:
assumes *reflexive-on* (*in-field* $(\leq_{L1})$) $(\leq_{L1})$
and *partial-equivalence-rel* $(\leq_{L2})$
shows $(\leq_L) =$ *tfr.tdfr.L*
⟨*proof*⟩

end

end

2.8.8 Galois Relator

theory *Transport-Functions-Galois-Relator*
  imports
    Transport-Functions-Relation-Simplifications
begin

Dependent Function Relator **context** *transport-Dep-Fun-Rel*
begin

interpretation *flip* : *transport-Dep-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2*
  rewrites *flip.t1.counit* $\equiv \eta_1$ ⟨*proof*⟩

lemma *Dep-Fun-Rel-left-Galois-if-left-GaloisI*:
  assumes $((\le_{L1})\ {}_h\trianglelefteq\ (\le_{R1}))\ l1\ r1$
  and *refl-L1*: *reflexive-on* (*in-dom* $(\le_{L1})$) $(\le_{L1})$
  and *mono-r2*: $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow ((\le_{R2\ (l1\ x)\ x'}) \Rrightarrow_m (\le_{L2\ x\ (r1\ x')}))\ (r2_{x\ x'})$
  and *L2-le2*: $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ x1}) \le (\le_{L2\ x1\ x2})$
  and *ge-L2-r2-le2*: $\bigwedge x\ x'\ y'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow$ *in-dom* $(\le_{R2\ (l1\ x)\ x'})\ y' \Longrightarrow$
    $(\ge_{L2\ x\ (r1\ x')})\ (r2_{x\ (l1\ x)}\ y') \le (\ge_{L2\ x\ (r1\ x')})\ (r2_{x\ x'}\ y')$
  and *trans-L2*: $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
  and $g \le_R g$
  and $f\ {}_L\lessapprox\ g$
  shows $([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\ f\ g$
⟨*proof*⟩

lemma *left-rel-right-if-Dep-Fun-Rel-left-GaloisI*:
  assumes *mono-l1*: $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1$
  and *half-galois-prop-right1*: $((\le_{L1}) \trianglelefteq_h (\le_{R1}))\ l1\ r1$
  and *L2-unit-le2*: $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
  and *ge-L2-r2-le1*: $\bigwedge x1\ x2\ y'.\ x1 \le_{L1} x2 \Longrightarrow$ *in-codom* $(\le_{R2\ (l1\ x1)\ (l1\ x2)})\ y'$
$\Longrightarrow$
    $(\ge_{L2\ x1\ x2})\ (r2_{x1\ (l1\ x2)}\ y') \le (\ge_{L2\ x1\ x2})\ (r2_{x2\ (l1\ x2)}\ y')$
  and *rel-f-g*: $([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\ f\ g$
  shows $f \le_L r\ g$
⟨*proof*⟩

lemma *left-Galois-if-Dep-Fun-Rel-left-GaloisI*:
  assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1$
  and $((\le_{L1}) \trianglelefteq_h (\le_{R1}))\ l1\ r1$
  and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
  and $\bigwedge x1\ x2\ y'.\ x1 \le_{L1} x2 \Longrightarrow$ *in-codom* $(\le_{R2\ (l1\ x1)\ (l1\ x2)})\ y' \Longrightarrow$
    $(\ge_{L2\ x1\ x2})\ (r2_{x1\ (l1\ x2)}\ y') \le (\ge_{L2\ x1\ x2})\ (r2_{x2\ (l1\ x2)}\ y')$
  and *in-codom* $(\le_R)\ g$
  and $([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\ f\ g$
  shows $f\ {}_L\lessapprox\ g$
  ⟨*proof*⟩

lemma *left-right-rel-if-Dep-Fun-Rel-left-GaloisI*:
assumes *mono-r1*: $((\le_{R1}) \Rrightarrow_m (\le_{L1}))\ r1$
and *half-galois-prop-left2*: $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$
$((\le_{L2\ (r1\ x1')\ (r1\ x2')})\ {}_h\trianglelefteq\ (\le_{R2\ (\varepsilon_1\ x1')\ x2'}))\ (l2_{x2'\ (r1\ x1')})\ (r2_{(r1\ x1')\ x2'})$
and *R2-le1*: $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x1')\ x2'}) \le (\le_{R2\ x1'\ x2'})$
and *R2-l2-le1*: $\bigwedge x1'\ x2'\ y.\ x1' \le_{R1} x2' \Longrightarrow in\text{-}dom\ (\le_{L2\ (r1\ x1')\ (r1\ x2')})\ y \Longrightarrow$
$(\le_{R2\ x1'\ x2'})\ (l2_{x2'\ (r1\ x1')}\ y) \le (\le_{R2\ x1'\ x2'})\ (l2_{x1'\ (r1\ x1')}\ y)$
and *rel-f-g*: $([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\ f\ g$
shows $l\ f \le_R g$
⟨*proof*⟩

lemma *left-Galois-if-Dep-Fun-Rel-left-GaloisI'*:
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1$ **and** $((\le_{R1}) \Rrightarrow_m (\le_{L1}))\ r1$
and $((\le_{L1}) \trianglelefteq_h (\le_{R1}))\ l1\ r1$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$
$((\le_{L2\ (r1\ x1')\ (r1\ x2')})\ {}_h\trianglelefteq\ (\le_{R2\ (\varepsilon_1\ x1')\ x2'}))\ (l2_{x2'\ (r1\ x1')})\ (r2_{(r1\ x1')\ x2'})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x1')\ x2'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x1\ x2\ y'.\ x1 \le_{L1} x2 \Longrightarrow in\text{-}codom\ (\le_{R2\ (l1\ x1)\ (l1\ x2)})\ y' \Longrightarrow$
$(\ge_{L2\ x1\ x2})\ (r2_{x1\ (l1\ x2)}\ y') \le (\ge_{L2\ x1\ x2})\ (r2_{x2\ (l1\ x2)}\ y')$
and $\bigwedge x1'\ x2'\ y.\ x1' \le_{R1} x2' \Longrightarrow in\text{-}dom\ (\le_{L2\ (r1\ x1')\ (r1\ x2')})\ y \Longrightarrow$
$(\le_{R2\ x1'\ x2'})\ (l2_{x2'\ (r1\ x1')}\ y) \le (\le_{R2\ x1'\ x2'})\ (l2_{x1'\ (r1\ x1')}\ y)$
and $([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\ f\ g$
shows $f\ {}_{L}\lessapprox\ g$
⟨*proof*⟩

lemma *left-Galois-iff-Dep-Fun-Rel-left-GaloisI*:
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1$
and $((\le_{L1}) \trianglelefteq (\le_{R1}))\ l1\ r1$
and *reflexive-on* $(in\text{-}dom\ (\le_{L1}))\ (\le_{L1})$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow ((\le_{R2\ (l1\ x)\ x'}) \Rrightarrow_m (\le_{L2\ x\ (r1\ x')}))\ (r2_{x\ x'})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ x1}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2\ y'.\ x1 \le_{L1} x2 \Longrightarrow in\text{-}codom\ (\le_{R2\ (l1\ x1)\ (l1\ x2)})\ y' \Longrightarrow$
$(\ge_{L2\ x1\ x2})\ (r2_{x1\ (l1\ x2)}\ y') \le (\ge_{L2\ x1\ x2})\ (r2_{x2\ (l1\ x2)}\ y')$
and $\bigwedge x\ x'\ y'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow in\text{-}dom\ (\le_{R2\ (l1\ x)\ x'})\ y' \Longrightarrow$
$(\ge_{L2\ x\ (r1\ x')})\ (r2_{x\ (l1\ x)}\ y') \le (\ge_{L2\ x\ (r1\ x')})\ (r2_{x\ x'}\ y')$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow transitive\ (\le_{L2\ x1\ x2})$
and $g \le_R g$
shows $f\ {}_{L}\lessapprox\ g \longleftrightarrow ([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\ f\ g$
⟨*proof*⟩

lemma *left-Galois-iff-Dep-Fun-Rel-left-Galois-ge-left-rel2-assmI*:
assumes $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$
$([in\text{-}codom\ (\le_{R2\ (l1\ x1)\ (l1\ x2)})] \Rrightarrow (\le_{L2\ x1\ x2}))\ (r2_{x1\ (l1\ x2)})\ (r2_{x2\ (l1\ x2)})$

and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
shows $\bigwedge x1\ x2\ y'.\ x1 \le_{L1} x2 \Longrightarrow$ *in-codom* $(\le_{R2\ (l1\ x1)\ (l1\ x2)})\ y' \Longrightarrow$
$(\ge_{L2\ x1\ x2})\ (r2_{x1\ (l1\ x2)}\ y') \le (\ge_{L2\ x1\ x2})\ (r2_{x2\ (l1\ x2)}\ y')$
⟨*proof*⟩

lemma *left-Galois-iff-Dep-Fun-Rel-left-Galois-ge-left-rel2-assmI'*:
assumes $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow$
$([$*in-dom* $(\le_{R2\ (l1\ x)\ x'})] \Rrightarrow (\le_{L2\ x\ (r1\ x')}))\ (r2_{x\ (l1\ x)})\ (r2_{x\ x'})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
shows $\bigwedge x\ x'\ y'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow$ *in-dom* $(\le_{R2\ (l1\ x)\ x'})\ y' \Longrightarrow$
$(\ge_{L2\ x\ (r1\ x')})\ (r2_{x\ (l1\ x)}\ y') \le (\ge_{L2\ x\ (r1\ x')})\ (r2_{x\ x'}\ y')$
⟨*proof*⟩

lemma *left-Galois-iff-Dep-Fun-Rel-left-Galois-mono-assm-in-codom-rightI*:
assumes *mono-l1*: $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1$
and *half-galois-prop-right1*: $((\le_{L1}) \trianglelefteq_h (\le_{R1}))\ l1\ r1$
and *refl-L1*: *reflexive-on* (*in-codom* $(\le_{L1})$) $(\le_{L1})$
and *L2-le-unit2*: $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
and *mono-r2*: $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
$[$*in-field* $(\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
and $x1 \le_{L1} x2$
shows $([$*in-codom* $(\le_{R2\ (l1\ x1)\ (l1\ x2)})] \Rrightarrow (\le_{L2\ x1\ x2}))\ (r2_{x1\ (l1\ x2)})\ (r2_{x2\ (l1\ x2)})$
⟨*proof*⟩

lemma *left-Galois-iff-Dep-Fun-Rel-left-Galois-mono-assm-in-dom-rightI*:
assumes *mono-l1*: $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1$
and *half-galois-prop-right1*: $((\le_{L1}) \trianglelefteq (\le_{R1}))\ l1\ r1$
and *refl-L1*: *reflexive-on* (*in-dom* $(\le_{L1})$) $(\le_{L1})$
and *mono-r2*: $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
$[$*in-field* $(\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
and $x\ {}_{L1}\lessapprox x'$
shows $([$*in-dom* $(\le_{R2\ (l1\ x)\ x'})] \Rrightarrow (\le_{L2\ x\ (r1\ x')}))\ (r2_{x\ (l1\ x)})\ (r2_{x\ x'})$
⟨*proof*⟩

lemma *left-Galois-iff-Dep-Fun-Rel-left-Galois-if-monoI*:
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1$
and $((\le_{L1}) \trianglelefteq (\le_{R1}))\ l1\ r1$
and *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow ((\le_{R2\ (l1\ x)\ x'}) \Rrightarrow_m (\le_{L2\ x\ (r1\ x')}))\ (r2_{x\ x'})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ x1}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
$[$*in-field* $(\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
and $g \le_R g$
shows $f\ {}_{L}\lessapprox g \longleftrightarrow ([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\ f\ g$
⟨*proof*⟩

lemma *left-Galois-iff-Dep-Fun-Rel-left-Galois-left-rel2-le-assmI*:
  assumes *refl-L1*: *reflexive-on* (*in-dom* $(\le_{L1})$) $(\le_{L1})$
  and *mono-L2*: $([x1 :: \top] \Rrightarrow_m [x2 \text{ - } :: (\le_{L1}) \mid x1 \le_{L1} x2] \Rrightarrow_m (\le))$ *L2*
  and $x1 \le_{L1} x2$
  shows $(\le_{L2\ x1\ x1}) \le (\le_{L2\ x1\ x2})$
⟨*proof*⟩

lemma *left-Galois-iff-Dep-Fun-Rel-left-Galois-left-rel2-unit1-le-assmI*:
  assumes *mono-l1*: $((\le_{L1}) \Rrightarrow_m (\le_{R1}))$ *l1*
  and *half-galois-prop-right1*: $((\le_{L1}) \trianglelefteq_h (\le_{R1}))$ *l1 r1*
  and *refl-L1*: *reflexive-on* (*in-codom* $(\le_{L1})$) $(\le_{L1})$
  and *antimono-L2*:
   $([x1 :: \top] \Rrightarrow_m [x2\ x3 :: (\le_{L1}) \mid (x1 \le_{L1} x2 \wedge x3 \le_{L1} \eta_1\ x2)] \Rrightarrow_m (\ge))$ *L2*
  and $x1 \le_{L1} x2$
  shows $(\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
⟨*proof*⟩

lemma *left-Galois-iff-Dep-Fun-Rel-left-Galois-if-monoI'*:
  assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))$ *l1*
  and $((\le_{L1}) \trianglelefteq (\le_{R1}))$ *l1 r1*
  and *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
  and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow ((\le_{R2\ (l1\ x)\ x'}) \Rrightarrow_m (\le_{L2\ x\ (r1\ x')}))\ (r2_{x\ x'})$
  and $([x1 :: \top] \Rrightarrow_m [x2 \text{ - } :: (\le_{L1}) \mid x1 \le_{L1} x2] \Rrightarrow_m (\le))$ *L2*
  and $([x1 :: \top] \Rrightarrow_m [x2\ x3 :: (\le_{L1}) \mid (x1 \le_{L1} x2 \wedge x3 \le_{L1} \eta_1\ x2)] \Rrightarrow_m (\ge))$ *L2*
  and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox\ x1'] \Rrightarrow$
   $[\textit{in-field}\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
  and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
  and $g \le_R g$
  shows $f\ {}_{L}\lessapprox\ g \longleftrightarrow ([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\ f\ g$
  ⟨*proof*⟩

corollary *left-Galois-iff-Dep-Fun-Rel-left-Galois-if-mono-if-galois-connectionI*:
  assumes $((\le_{L1}) \dashv (\le_{R1}))$ *l1 r1*
  and *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
  and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow ((\le_{R2\ (l1\ x)\ x'}) \Rrightarrow_m (\le_{L2\ x\ (r1\ x')}))\ (r2_{x\ x'})$
  and $([x1 :: \top] \Rrightarrow_m [x2 \text{ - } :: (\le_{L1}) \mid x1 \le_{L1} x2] \Rrightarrow_m (\le))$ *L2*
  and $([x1 :: \top] \Rrightarrow_m [x2\ x3 :: (\le_{L1}) \mid (x1 \le_{L1} x2 \wedge x3 \le_{L1} \eta_1\ x2)] \Rrightarrow_m (\ge))$ *L2*
  and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox\ x1'] \Rrightarrow$
   $[\textit{in-field}\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
  and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
  and $g \le_R g$
  shows $f\ {}_{L}\lessapprox\ g \longleftrightarrow ([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\ f\ g$
  ⟨*proof*⟩

interpretation *flip-inv* : *galois* $(\ge_{R1})$ $(\ge_{L1})$ *r1 l1* ⟨*proof*⟩

lemma *left-Galois-iff-Dep-Fun-Rel-left-Galois-left-rel2-unit1-le-assm-if-galois-equivI*:

assumes *galois-equiv1*: $((\le_{L1}) \equiv_{G} (\le_{R1}))$ *l1 r1*
and *refl-L1*: *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and *mono-L2*: $([x1 :: \top] \Rrightarrow_m [x2 \text{ - } :: (\le_{L1}) \mid x1 \le_{L1} x2] \Rrightarrow_m (\le))$ *L2*
and $x1 \le_{L1} x2$
shows $(\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
⟨*proof*⟩

lemma *left-Galois-iff-Dep-Fun-Rel-left-Galois-if-galois-equivalenceI*:
assumes $((\le_{L1}) \equiv_{G} (\le_{R1}))$ *l1 r1*
and *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow ((\le_{R2\ (l1\ x)\ x'}) \Rrightarrow_m (\le_{L2\ x\ (r1\ x')}))\ (r2_{x\ x'})$
and $([x1 :: \top] \Rrightarrow_m [x2 \text{ - } :: (\le_{L1}) \mid x1 \le_{L1} x2] \Rrightarrow_m (\le))$ *L2*
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox\ x1'] \Rrightarrow$
$[\textit{in-field}\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))$ *r2*
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
and $g \le_{R} g$
shows $f\ {}_{L}\lessapprox\ g \longleftrightarrow ([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\ f\ g$
⟨*proof*⟩

corollary *left-Galois-iff-Dep-Fun-Rel-left-Galois-if-galois-equivalenceI'*:
assumes $((\le_{L1}) \equiv_{G} (\le_{R1}))$ *l1 r1*
and *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow ((\le_{R2\ (l1\ x)\ x'}) \Rrightarrow_m (\le_{L2\ x\ (r1\ x')}))\ (r2_{x\ x'})$
and $([x1\ x2 :: (\ge_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid x1 \le_{L1} x3] \Rrightarrow (\le))$ *L2*
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox\ x1'] \Rrightarrow$
$[\textit{in-field}\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))$ *r2*
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
and $g \le_{R} g$
shows $f\ {}_{L}\lessapprox\ g \longleftrightarrow ([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\ f\ g$
⟨*proof*⟩

corollary *left-Galois-iff-Dep-Fun-Rel-left-Galois-if-preorder-equivalenceI*:
assumes $((\le_{L1}) \equiv_{pre} (\le_{R1}))$ *l1 r1*
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow ((\le_{R2\ (l1\ x)\ x'}) \Rrightarrow_m (\le_{L2\ x\ (r1\ x')}))\ (r2_{x\ x'})$
and $([x1\ x2 :: (\ge_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid x1 \le_{L1} x3] \Rrightarrow (\le))$ *L2*
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox\ x1'] \Rrightarrow$
$[\textit{in-field}\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))$ *r2*
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
and $g \le_{R} g$
shows $f\ {}_{L}\lessapprox\ g \longleftrightarrow ([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\ f\ g$
⟨*proof*⟩

corollary *left-Galois-iff-Dep-Fun-Rel-left-Galois-if-preorder-equivalenceI'*:
assumes $((\le_{L1}) \equiv_{pre} (\le_{R1}))$ *l1 r1*
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \equiv_{pre} (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $([x1\ x2 :: (\ge_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid x1 \le_{L1} x3] \Rrightarrow (\le))$ *L2*
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox\ x1'] \Rrightarrow$

$[\textit{in-field}\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
and $g \le_R g$
shows $f\ {}_{L}\lessapprox\ g \longleftrightarrow ([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\ f\ g$
⟨*proof*⟩

Simplification of Restricted Function Relator **lemma** *Dep-Fun-Rel-left-Galois-restrict-left-right-eq*
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1$ **and** $((\le_{R1}) \Rrightarrow_m (\le_{L1}))\ r1$
and $((\le_{L1}) \trianglelefteq_h (\le_{R1}))\ l1\ r1$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$
$((\le_{L2\ (r1\ x1')\ (r1\ x2')})\ {}_h\trianglelefteq\ (\le_{R2\ (\varepsilon_1\ x1')\ x2'}))\ (l2_{x2'\ (r1\ x1')})\ (r2_{(r1\ x1')\ x2'})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x1')\ x2'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x1'\ x2'\ y.\ x1' \le_{R1} x2' \Longrightarrow \textit{in-dom}\ (\le_{L2\ (r1\ x1')\ (r1\ x2')})\ y \Longrightarrow$
$(\le_{R2\ x1'\ x2'})\ (l2_{x2'\ (r1\ x1')}\ y) \le (\le_{R2\ x1'\ x2'})\ (l2_{x1'\ (r1\ x1')}\ y)$
and $\bigwedge x1\ x2\ y'.\ x1 \le_{L1} x2 \Longrightarrow \textit{in-codom}\ (\le_{R2\ (l1\ x1)\ (l1\ x2)})\ y' \Longrightarrow$
$(\ge_{L2\ x1\ x2})\ (r2_{x1\ (l1\ x2)}\ y') \le (\ge_{L2\ x1\ x2})\ (r2_{x2\ (l1\ x2)}\ y')$
shows $([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\restriction_{\textit{in-dom}\ (\le_L)}\upharpoonright_{\textit{in-codom}\ (\le_R)}$
$= ([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))$
⟨*proof*⟩

lemma *Dep-Fun-Rel-left-Galois-restrict-left-right-eq-Dep-Fun-Rel-left-GaloisI'*:
assumes $((\le_{L1}) \dashv (\le_{R1}))\ l1\ r1$
and *reflexive-on* $(\textit{in-field}\ (\le_{L1}))\ (\le_{L1})$
and *reflexive-on* $(\textit{in-field}\ (\le_{R1}))\ (\le_{R1})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$
$((\le_{L2\ (r1\ x1')\ (r1\ x2')})\ {}_h\trianglelefteq\ (\le_{R2\ (\varepsilon_1\ x1')\ x2'}))\ (l2_{x2'\ (r1\ x1')})\ (r2_{(r1\ x1')\ x2'})$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid (x2 \le_{L1} x3 \wedge x4 \le_{L1} \eta_1\ x3)] \Rrightarrow (\ge))\ L2$
and $([x1'\ x2' :: (\le_{R1}) \mid \varepsilon_1\ x2' \le_{R1} x1'] \Rrightarrow_m [x3'\ x4' :: (\le_{R1}) \mid x2' \le_{R1} x3'] \Rrightarrow (\le))\ R2$
and $([x1'\ x2' :: (\le_{R1})] \Rrightarrow_m [x1\ x2 :: (\le_{L1}) \mid x2\ {}_{L1}\lessapprox\ x1'] \Rrightarrow$
$[\textit{in-field}\ (\le_{L2\ x1\ (r1\ x2')})] \Rrightarrow (\le_{R2\ (l1\ x1)\ x2'}))\ l2$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox\ x1'] \Rrightarrow$
$[\textit{in-field}\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$ *transitive* $(\le_{R2\ x1'\ x2'})$
shows $([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\restriction_{\textit{in-dom}\ (\le_L)}\upharpoonright_{\textit{in-codom}\ (\le_R)}$
$= ([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))$
⟨*proof*⟩

Simplification of Restricted Function Relator for Nested Transports

lemma *Dep-Fun-Rel-left-Galois-restrict-left-right-restrict-left-right-eq*:
fixes $S :: 'a1 \Rightarrow 'a2 \Rightarrow 'b1 \Rightarrow 'b2 \Rightarrow bool$
assumes $((\le_{L1})\ {}_h\trianglelefteq\ (\le_{R1}))\ l1\ r1$
shows $([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow (S\ x\ x')\restriction_{\textit{in-dom}\ (\le_{L2\ x\ (r1\ x')})}\upharpoonright_{\textit{in-codom}\ (\le_{R2\ (l1\ x)\ x')})}$
$\restriction_{\textit{in-dom}\ (\le_L)}\upharpoonright_{\textit{in-codom}\ (\le_R)} =$

$([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow S\ x\ x')\restriction_{in\text{-}dom\ (\le_L)}\upharpoonright_{in\text{-}codom\ (\le_R)}$ (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

end

Function Relator **context** *transport-Fun-Rel*
begin

corollary *Fun-Rel-left-Galois-if-left-GaloisI*:
assumes $((\le_{L1})\ {}_h\trianglelefteq\ (\le_{R1}))\ l1\ r1$
and *reflexive-on* $(in\text{-}dom\ (\le_{L1}))\ (\le_{L1})$
and $((\le_{R2}) \Rrightarrow_m (\le_{L2}))\ r2$
and *transitive* $(\le_{L2})$
and $g \le_R g$
and $f\ {}_L\lessapprox\ g$
shows $(({}_{L1}\lessapprox) \Rrightarrow ({}_{L2}\lessapprox))\ f\ g$
⟨*proof*⟩

corollary *left-Galois-if-Fun-Rel-left-GaloisI*:
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1$
and $((\le_{L1}) \trianglelefteq_h (\le_{R1}))\ l1\ r1$
and *in-codom* $(\le_R)\ g$
and $(({}_{L1}\lessapprox) \Rrightarrow ({}_{L2}\lessapprox))\ f\ g$
shows $f\ {}_L\lessapprox\ g$
⟨*proof*⟩

lemma *left-Galois-if-Fun-Rel-left-GaloisI'*:
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1$ **and** $((\le_{R1}) \Rrightarrow_m (\le_{L1}))\ r1$
and $((\le_{L1}) \trianglelefteq_h (\le_{R1}))\ l1\ r1$
and $((\le_{L2})\ {}_h\trianglelefteq\ (\le_{R2}))\ l2\ r2$
and $(({}_{L1}\lessapprox) \Rrightarrow ({}_{L2}\lessapprox))\ f\ g$
shows $f\ {}_L\lessapprox\ g$
⟨*proof*⟩

corollary *left-Galois-iff-Fun-Rel-left-GaloisI*:
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1$
and $((\le_{L1}) \trianglelefteq (\le_{R1}))\ l1\ r1$
and *reflexive-on* $(in\text{-}dom\ (\le_{L1}))\ (\le_{L1})$
and $((\le_{R2}) \Rrightarrow_m (\le_{L2}))\ r2$
and *transitive* $(\le_{L2})$
and $g \le_R g$
shows $f\ {}_L\lessapprox\ g \longleftrightarrow (({}_{L1}\lessapprox) \Rrightarrow ({}_{L2}\lessapprox))\ f\ g$
⟨*proof*⟩

Simplification of Restricted Function Relator **lemma** *Fun-Rel-left-Galois-restrict-left-right-eq-Fun*
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1$ **and** $((\le_{R1}) \Rrightarrow_m (\le_{L1}))\ r1$
and $((\le_{L1}) \trianglelefteq_h (\le_{R1}))\ l1\ r1$
and $((\le_{L2})\ {}_h\trianglelefteq\ (\le_{R2}))\ l2\ r2$
shows $(({}_{L1}\lessapprox) \Rrightarrow ({}_{L2}\lessapprox))\restriction_{in\text{-}dom\ (\le_L)}\upharpoonright_{in\text{-}codom\ (\le_R)} = (({}_{L1}\lessapprox) \Rrightarrow ({}_{L2}\lessapprox))$

$\langle proof \rangle$

Simplification of Restricted Function Relator for Nested Transports

lemma *Fun-Rel-left-Galois-restrict-left-right-restrict-left-right-eq*:
fixes $S :: 'b1 \Rightarrow 'b2 \Rightarrow bool$
assumes $((\leq_{L1})\ {}_h\trianglelefteq\ (\leq_{R1}))\ l1\ r1$
shows $(({}_{L1}\lessapprox) \Rrightarrow S\restriction_{in\text{-}dom\ (\leq_{L2})}\upharpoonright_{in\text{-}codom\ (\leq_{R2})})\restriction_{in\text{-}dom\ (\leq_{L})}\upharpoonright_{in\text{-}codom\ (\leq_{R})}$
$=$
$(({}_{L1}\lessapprox) \Rrightarrow S)\restriction_{in\text{-}dom\ (\leq_{L})}\upharpoonright_{in\text{-}codom\ (\leq_{R})}$
$\langle proof \rangle$

end

Monotone Dependent Function Relator **context** *transport-Mono-Dep-Fun-Rel*
begin

lemma *Dep-Fun-Rel-left-Galois-if-left-GaloisI*:
assumes $((\leq_{L1})\ {}_h\trianglelefteq\ (\leq_{R1}))\ l1\ r1$
and *reflexive-on* $(in\text{-}dom\ (\leq_{L1}))\ (\leq_{L1})$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow ((\leq_{R2\ (l1\ x)\ x'}) \Rrightarrow_m (\leq_{L2\ x\ (r1\ x')}))\ (r2_{x\ x'})$
and $\bigwedge x1\ x2.\ x1 \leq_{L1} x2 \Longrightarrow (\leq_{L2\ x1\ x1}) \leq (\leq_{L2\ x1\ x2})$
and $\bigwedge x\ x'\ y'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow in\text{-}dom\ (\leq_{R2\ (l1\ x)\ x'})\ y' \Longrightarrow$
$(\geq_{L2\ x\ (r1\ x')})\ (r2_{x\ (l1\ x)}\ y') \leq (\geq_{L2\ x\ (r1\ x')})\ (r2_{x\ x'}\ y')$
and $\bigwedge x1\ x2.\ x1 \leq_{L1} x2 \Longrightarrow transitive\ (\leq_{L2\ x1\ x2})$
and $f\ {}_{L}\lessapprox\ g$
shows $([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\ f\ g$
$\langle proof \rangle$

lemma *left-Galois-if-Dep-Fun-Rel-left-GaloisI*:
assumes $(tdfr.R \Rrightarrow_m tdfr.L)\ r$
and $((\leq_{L1}) \Rrightarrow_m (\leq_{R1}))\ l1$
and $((\leq_{L1}) \trianglelefteq_h (\leq_{R1}))\ l1\ r1$
and $\bigwedge x1\ x2.\ x1 \leq_{L1} x2 \Longrightarrow (\leq_{L2\ x1\ (\eta_1\ x2)}) \leq (\leq_{L2\ x1\ x2})$
and $\bigwedge x1\ x2\ y'.\ x1 \leq_{L1} x2 \Longrightarrow in\text{-}codom\ (\leq_{R2\ (l1\ x1)\ (l1\ x2)})\ y' \Longrightarrow$
$(\geq_{L2\ x1\ x2})\ (r2_{x1\ (l1\ x2)}\ y') \leq (\geq_{L2\ x1\ x2})\ (r2_{x2\ (l1\ x2)}\ y')$
and $in\text{-}dom\ (\leq_{L})\ f$
and $in\text{-}codom\ (\leq_{R})\ g$
and $([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\ f\ g$
shows $f\ {}_{L}\lessapprox\ g$
$\langle proof \rangle$

lemma *left-Galois-iff-Dep-Fun-Rel-left-GaloisI*:
assumes $(tdfr.R \Rrightarrow_m tdfr.L)\ r$
and $((\leq_{L1}) \Rrightarrow_m (\leq_{R1}))\ l1$
and $((\leq_{L1}) \trianglelefteq (\leq_{R1}))\ l1\ r1$
and *reflexive-on* $(in\text{-}field\ (\leq_{L1}))\ (\leq_{L1})$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow ((\leq_{R2\ (l1\ x)\ x'}) \Rrightarrow_m (\leq_{L2\ x\ (r1\ x')}))\ (r2_{x\ x'})$
and $\bigwedge x1\ x2.\ x1 \leq_{L1} x2 \Longrightarrow (\leq_{L2\ x1\ x1}) \leq (\leq_{L2\ x1\ x2})$

and $\bigwedge$*x1 x2*. *x1* $\le_{L1}$ *x2* $\Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox\ x1'] \Rrightarrow$
$[\textit{in-field}\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
and $\bigwedge$*x1 x2*. *x1* $\le_{L1}$ *x2* $\Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
and *in-dom* $(\le_L)$ *f*
and *in-codom* $(\le_R)$ *g*
shows $f\ {}_{L}\lessapprox\ g \longleftrightarrow ([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\ f\ g$
⟨*proof*⟩

lemma *left-Galois-iff-Dep-Fun-Rel-left-Galois-if-mono-if-galois-connectionI*:
assumes *galois-conn1*: $((\le_{L1}) \dashv (\le_{R1}))\ l1\ r1$
and *refl-L1*: *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow ((\le_{R2\ (l1\ x)\ x'}) \Rrightarrow_m (\le_{L2\ x\ (r1\ x')}))\ (r2_{x\ x'})$
and $\bigwedge$*x1 x2*. *x1* $\le_{L1}$ *x2* $\Longrightarrow (\le_{L2\ x1\ x1}) \le (\le_{L2\ x1\ x2})$
and *L2-le-unit2*: $\bigwedge$*x1 x2*. *x1* $\le_{L1}$ *x2* $\Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
and *mono-r2*: $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox\ x1'] \Rrightarrow$
$[\textit{in-field}\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
and *trans-L2*: $\bigwedge$*x1 x2*. *x1* $\le_{L1}$ *x2* $\Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
and *in-dom* $(\le_L)$ *f*
and *in-codom* $(\le_R)$ *g*
shows $f\ {}_{L}\lessapprox\ g \longleftrightarrow ([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\ f\ g$ (**is** *?lhs* $\longleftrightarrow$ *?rhs*)
⟨*proof*⟩

corollary *left-Galois-iff-Dep-Fun-Rel-left-Galois-if-mono-if-galois-connectionI'*:
assumes $((\le_{L1}) \dashv (\le_{R1}))\ l1\ r1$
and *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow ((\le_{R2\ (l1\ x)\ x'}) \Rrightarrow_m (\le_{L2\ x\ (r1\ x')}))\ (r2_{x\ x'})$
and $([x1 :: \top] \Rrightarrow_m [x2\ \text{-} :: (\le_{L1}) \mid x1 \le_{L1} x2] \Rrightarrow_m (\le))\ L2$
and $([x1 :: \top] \Rrightarrow_m [x2\ x3 :: (\le_{L1}) \mid (x1 \le_{L1} x2 \wedge x3 \le_{L1} \eta_1\ x2)] \Rrightarrow_m (\ge))\ L2$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox\ x1'] \Rrightarrow$
$[\textit{in-field}\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
and $\bigwedge$*x1 x2*. *x1* $\le_{L1}$ *x2* $\Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
and *in-dom* $(\le_L)$ *f*
and *in-codom* $(\le_R)$ *g*
shows $f\ {}_{L}\lessapprox\ g \longleftrightarrow ([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\ f\ g$ (**is** *?lhs* $\longleftrightarrow$ *?rhs*)
⟨*proof*⟩

corollary *left-Galois-eq-Dep-Fun-Rel-left-Galois-restrict-if-mono-if-galois-connectionI*:
assumes $((\le_{L1}) \dashv (\le_{R1}))\ l1\ r1$
and *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow ((\le_{R2\ (l1\ x)\ x'}) \Rrightarrow_m (\le_{L2\ x\ (r1\ x')}))\ (r2_{x\ x'})$
and $([x1 :: \top] \Rrightarrow_m [x2\ \text{-} :: (\le_{L1}) \mid x1 \le_{L1} x2] \Rrightarrow_m (\le))\ L2$
and $([x1 :: \top] \Rrightarrow_m [x2\ x3 :: (\le_{L1}) \mid (x1 \le_{L1} x2 \wedge x3 \le_{L1} \eta_1\ x2)] \Rrightarrow_m (\ge))\ L2$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox\ x1'] \Rrightarrow$
$[\textit{in-field}\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
and $\bigwedge$*x1 x2*. *x1* $\le_{L1}$ *x2* $\Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
shows $({}_{L}\lessapprox) = ([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\restriction_{\textit{in-dom}\ (\le_L)}\upharpoonright_{\textit{in-codom}\ (\le_R)}$

⟨*proof*⟩

lemma *left-Galois-iff-Dep-Fun-Rel-left-Galois-if-galois-equivalenceI*:
assumes $((\le_{L1}) \equiv_G (\le_{R1}))$ *l1 r1*
and *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow ((\le_{R2\ (l1\ x)\ x'}) \Rrightarrow_m (\le_{L2\ x\ (r1\ x')}))\ (r2_{x\ x'})$
and $([x1\ x2 :: (\ge_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid x1 \le_{L1} x3] \Rrightarrow (\le))\ L2$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox\ x1'] \Rrightarrow$
$[in\text{-}field\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
and *in-dom* $(\le_L)$ *f*
and *in-codom* $(\le_R)$ *g*
shows $f\ {}_{L}\lessapprox\ g \longleftrightarrow ([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\ f\ g$
⟨*proof*⟩

theorem *left-Galois-eq-Dep-Fun-Rel-left-Galois-restrict-if-galois-equivalenceI*:
assumes $((\le_{L1}) \equiv_G (\le_{R1}))$ *l1 r1*
and *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow ((\le_{R2\ (l1\ x)\ x'}) \Rrightarrow_m (\le_{L2\ x\ (r1\ x')}))\ (r2_{x\ x'})$
and $([x1\ x2 :: (\ge_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid x1 \le_{L1} x3] \Rrightarrow (\le))\ L2$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox\ x1'] \Rrightarrow$
$[in\text{-}field\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
shows $({}_{L}\lessapprox) = ([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\restriction_{in\text{-}dom\ (\le_L)}\upharpoonleft_{in\text{-}codom\ (\le_R)}$
⟨*proof*⟩

corollary *left-Galois-iff-Dep-Fun-Rel-left-Galois-if-preorder-equivalenceI*:
assumes $((\le_{L1}) \equiv_{pre} (\le_{R1}))$ *l1 r1*
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow ((\le_{R2\ (l1\ x)\ x'}) \Rrightarrow_m (\le_{L2\ x\ (r1\ x')}))\ (r2_{x\ x'})$
and $([x1\ x2 :: (\ge_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid x1 \le_{L1} x3] \Rrightarrow (\le))\ L2$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox\ x1'] \Rrightarrow$
$[in\text{-}field\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
and *in-dom* $(\le_L)$ *f*
and *in-codom* $(\le_R)$ *g*
shows $f\ {}_{L}\lessapprox\ g \longleftrightarrow ([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\ f\ g$
⟨*proof*⟩

corollary *left-Galois-eq-Dep-Fun-Rel-left-Galois-restrict-if-preorder-equivalenceI*:
assumes $((\le_{L1}) \equiv_{pre} (\le_{R1}))$ *l1 r1*
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow ((\le_{R2\ (l1\ x)\ x'}) \Rrightarrow_m (\le_{L2\ x\ (r1\ x')}))\ (r2_{x\ x'})$
and $([x1\ x2 :: (\ge_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid x1 \le_{L1} x3] \Rrightarrow (\le))\ L2$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox\ x1'] \Rrightarrow$
$[in\text{-}field\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
shows $({}_{L}\lessapprox) = ([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\restriction_{in\text{-}dom\ (\le_L)}\upharpoonleft_{in\text{-}codom\ (\le_R)}$
⟨*proof*⟩

corollary *left-Galois-iff-Dep-Fun-Rel-left-Galois-if-preorder-equivalenceI'*:
assumes $((\le_{L1}) \equiv_{pre} (\le_{R1}))$ *l1 r1*
and $\bigwedge x\ x'.\ x\ _{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \equiv_{pre} (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $([x1\ x2 :: (\ge_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid x1 \le_{L1} x3] \Rrightarrow (\le))\ L2$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ _{L1}\lessapprox x1'] \Rrightarrow$
$[in\text{-}field\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
and *in-dom* $(\le_L)$ *f*
and *in-codom* $(\le_R)$ *g*
shows $f\ _L\lessapprox g \longleftrightarrow ([x\ x' :: (_{L1}\lessapprox)] \Rrightarrow (_{L2\ x\ x'}\lessapprox))\ f\ g$
⟨*proof*⟩

corollary *left-Galois-eq-Dep-Fun-Rel-left-Galois-restrict-if-preorder-equivalenceI'*:
assumes $((\le_{L1}) \equiv_{pre} (\le_{R1}))$ *l1 r1*
and $\bigwedge x\ x'.\ x\ _{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \equiv_{pre} (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $([x1\ x2 :: (\ge_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid x1 \le_{L1} x3] \Rrightarrow (\le))\ L2$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ _{L1}\lessapprox x1'] \Rrightarrow$
$[in\text{-}field\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
shows $(_L\lessapprox) = ([x\ x' :: (_{L1}\lessapprox)] \Rrightarrow (_{L2\ x\ x'}\lessapprox))\restriction_{in\text{-}dom\ (\le_L)}\upharpoonleft_{in\text{-}codom\ (\le_R)}$
⟨*proof*⟩

Simplification of Restricted Function Relator **lemma** *Dep-Fun-Rel-left-Galois-restrict-left-right-eq*
assumes *reflexive-on* (*in-field tdfr.L*) *tdfr.L*
and *reflexive-on* (*in-field tdfr.R*) *tdfr.R*
and $((\le_{L1}) \dashv (\le_{R1}))$ *l1 r1*
and *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and *reflexive-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$
$((\le_{L2\ (r1\ x1')\ (r1\ x2')})\ _h\trianglelefteq (\le_{R2\ (\varepsilon_1\ x1')\ x2'}))\ (l2_{x2'\ (r1\ x1')})\ (r2_{(r1\ x1')\ x2'})$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid (x2 \le_{L1} x3 \wedge x4 \le_{L1} \eta_1\ x3)] \Rrightarrow$
$(\ge))\ L2$
and $([x1'\ x2' :: (\le_{R1}) \mid \varepsilon_1\ x2' \le_{R1} x1'] \Rrightarrow_m [x3'\ x4' :: (\le_{R1}) \mid x2' \le_{R1} x3']$
$\Rrightarrow (\le))\ R2$
and $([x1'\ x2' :: (\le_{R1})] \Rrightarrow_m [x1\ x2 :: (\le_{L1}) \mid x2\ _{L1}\lessapprox x1'] \Rrightarrow$
$[in\text{-}field\ (\le_{L2\ x1\ (r1\ x2')})] \Rrightarrow (\le_{R2\ (l1\ x1)\ x2'}))\ l2$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ _{L1}\lessapprox x1'] \Rrightarrow$
$[in\text{-}field\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$ *transitive* $(\le_{R2\ x1'\ x2'})$
shows $([x\ x' :: (_{L1}\lessapprox)] \Rrightarrow (_{L2\ x\ x'}\lessapprox))\restriction_{in\text{-}dom\ (\le_L)}\upharpoonleft_{in\text{-}codom\ (\le_R)}$
$= ([x\ x' :: (_{L1}\lessapprox)] \Rrightarrow (_{L2\ x\ x'}\lessapprox))$
⟨*proof*⟩

interpretation *flip* : *transport-Dep-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2*
rewrites *flip.t1.unit* $\equiv \varepsilon_1$ ⟨*proof*⟩

lemma *Dep-Fun-Rel-left-Galois-restrict-left-right-eq-Dep-Fun-Rel-left-GaloisI*:

assumes $((\le_{L1}) \equiv_{pre} (\le_{R1}))$ *l1 r1*
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$
$((\le_{L2\ (r1\ x1')\ (r1\ x2')})\ {}_h\trianglelefteq\ (\le_{R2\ (\varepsilon_1\ x1')\ x2'}))\ (l2_{x2'\ (r1\ x1')})\ (r2_{(r1\ x1')\ x2'})$
and $([x1\ x2 :: (\ge_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid x1 \le_{L1} x3] \Rrightarrow (\le))\ L2$
and $([x1'\ x2' :: (\ge_{R1})] \Rrightarrow_m [x3'\ x4' :: (\le_{R1}) \mid x1' \le_{R1} x3'] \Rrightarrow (\le))\ R2$
and $([x1'\ x2' :: (\le_{R1})] \Rrightarrow_m [x1\ x2 :: (\le_{L1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
$[\textit{in-field}\ (\le_{L2\ x1\ (r1\ x2')})] \Rrightarrow (\le_{R2\ (l1\ x1)\ x2'}))\ l2$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
$[\textit{in-field}\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
and *PERS*: $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow \textit{partial-equivalence-rel}\ (\le_{L2\ x1\ x2})$
$\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow \textit{partial-equivalence-rel}\ (\le_{R2\ x1'\ x2'})$
shows $([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))\restriction_{\textit{in-dom}\ (\le_L)}\upharpoonright_{\textit{in-codom}\ (\le_R)}$
$= ([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow ({}_{L2\ x\ x'}\lessapprox))$
⟨*proof*⟩

Simplification of Restricted Function Relator for Nested Transports

lemma *Dep-Fun-Rel-left-Galois-restrict-left-right-restrict-left-right-eq*:
fixes $S :: 'a1 \Rightarrow 'a2 \Rightarrow 'b1 \Rightarrow 'b2 \Rightarrow bool$
assumes $((\le_{L1})\ {}_h\trianglelefteq\ (\le_{R1}))$ *l1 r1*
shows $([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow (S\ x\ x')\restriction_{\textit{in-dom}\ (\le_{L2\ x\ (r1\ x')})}\upharpoonright_{\textit{in-codom}\ (\le_{R2\ (l1\ x)\ x'})})$
$\restriction_{\textit{in-dom}\ (\le_L)}\upharpoonright_{\textit{in-codom}\ (\le_R)} =$
$([x\ x' :: ({}_{L1}\lessapprox)] \Rrightarrow S\ x\ x')\restriction_{\textit{in-dom}\ (\le_L)}\upharpoonright_{\textit{in-codom}\ (\le_R)}$
(**is** $?lhs\restriction_{?DL}\upharpoonright_{?CR} = ?rhs\restriction_{?DL}\upharpoonright_{?CR}$)
⟨*proof*⟩

end

Monotone Function Relator **context** *transport-Mono-Fun-Rel*
begin

corollary *Fun-Rel-left-Galois-if-left-GaloisI*:
assumes $((\le_{L1})\ {}_h\trianglelefteq\ (\le_{R1}))$ *l1 r1*
and *reflexive-on* $(\textit{in-dom}\ (\le_{L1}))\ (\le_{L1})$
and $((\le_{R2}) \Rrightarrow_m (\le_{L2}))\ (r2)$
and *transitive* $(\le_{L2})$
and $f\ {}_L\lessapprox\ g$
shows $(({}_{L1}\lessapprox) \Rrightarrow ({}_{L2}\lessapprox))\ f\ g$
⟨*proof*⟩

interpretation *flip* : *transport-Mono-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2* ⟨*proof*⟩

lemma *left-Galois-if-Fun-Rel-left-GaloisI*:
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))$ *l1*
and $((\le_{L1}) \trianglelefteq_h (\le_{R1}))$ *l1 r1*
and $((\le_{R2}) \Rrightarrow_m (\le_{L2}))$ *r2*
and *in-dom* $(\le_L)\ f$
and *in-codom* $(\le_R)\ g$

and $(({}_{L1}\lessapprox) \Rrightarrow ({}_{L2}\lessapprox))$ *f g*
shows *f* ${}_{L}\lessapprox$ *g*
⟨*proof*⟩

corollary *left-Galois-iff-Fun-Rel-left-GaloisI*:
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))$ *l1*
and $((\le_{L1}) \trianglelefteq (\le_{R1}))$ *l1 r1*
and *reflexive-on* (*in-dom* $(\le_{L1})$) $(\le_{L1})$
and $((\le_{R2}) \Rrightarrow_m (\le_{L2}))$ (*r2*)
and *transitive* $(\le_{L2})$
and *in-dom* $(\le_L)$ *f*
and *in-codom* $(\le_R)$ *g*
shows *f* ${}_{L}\lessapprox$ *g* $\longleftrightarrow$ $(({}_{L1}\lessapprox) \Rrightarrow ({}_{L2}\lessapprox))$ *f g*
⟨*proof*⟩

theorem *left-Galois-eq-Fun-Rel-left-Galois-restrictI*:
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))$ *l1*
and $((\le_{L1}) \trianglelefteq (\le_{R1}))$ *l1 r1*
and *reflexive-on* (*in-dom* $(\le_{L1})$) $(\le_{L1})$
and $((\le_{R2}) \Rrightarrow_m (\le_{L2}))$ *r2*
and *transitive* $(\le_{L2})$
shows $({}_{L}\lessapprox) = (({}_{L1}\lessapprox) \Rrightarrow ({}_{L2}\lessapprox))\restriction_{\textit{in-dom } (\le_L)}\upharpoonleft_{\textit{in-codom } (\le_R)}$
⟨*proof*⟩

Simplification of Restricted Function Relator **lemma** *Fun-Rel-left-Galois-restrict-left-right-eq-Fun*
assumes *reflexive-on* (*in-field tfr.tdfr.L*) *tfr.tdfr.L*
and *reflexive-on* (*in-field tfr.tdfr.R*) *tfr.tdfr.R*
and $((\le_{L1}) \Rrightarrow_m (\le_{R1}))$ *l1* **and** $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*
and $((\le_{L1}) \trianglelefteq_h (\le_{R1}))$ *l1 r1*
and $((\le_{L2})\ {}_h\trianglelefteq (\le_{R2}))$ *l2 r2*
shows $(({}_{L1}\lessapprox) \Rrightarrow ({}_{L2}\lessapprox))\restriction_{\textit{in-dom } (\le_L)}\upharpoonleft_{\textit{in-codom } (\le_R)} = (({}_{L1}\lessapprox) \Rrightarrow ({}_{L2}\lessapprox))$
⟨*proof*⟩

lemma *Fun-Rel-left-Galois-restrict-left-right-eq-Fun-Rel-left-GaloisI*:
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))$ *l1* **and** $((\le_{R1}) \Rrightarrow_m (\le_{L1}))$ *r1*
and $((\le_{L1}) \trianglelefteq_h (\le_{R1}))$ *l1 r1*
and *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and *reflexive-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$
and $((\le_{L2})\ {}_h\trianglelefteq (\le_{R2}))$ *l2 r2*
and *partial-equivalence-rel* $(\le_{L2})$
and *partial-equivalence-rel* $(\le_{R2})$
shows $(({}_{L1}\lessapprox) \Rrightarrow ({}_{L2}\lessapprox))\restriction_{\textit{in-dom } (\le_L)}\upharpoonleft_{\textit{in-codom } (\le_R)} = (({}_{L1}\lessapprox) \Rrightarrow ({}_{L2}\lessapprox))$
⟨*proof*⟩

Simplification of Restricted Function Relator for Nested Transports

lemma *Fun-Rel-left-Galois-restrict-left-right-restrict-left-right-eq*:
fixes *S* :: *'b1* ⇒ *'b2* ⇒ *bool*
assumes $((\le_{L1})\ {}_h\trianglelefteq (\le_{R1}))$ *l1 r1*

shows $((_{L1}\lessapprox) \Rrightarrow S\restriction_{in\text{-}dom\ (\le_{L2})}\upharpoonleft_{in\text{-}codom\ (\le_{R2})})\restriction_{in\text{-}dom\ (\le_{L})}\upharpoonleft_{in\text{-}codom\ (\le_{R})}$
=
$((_{L1}\lessapprox) \Rrightarrow S)\restriction_{in\text{-}dom\ (\le_{L})}\upharpoonleft_{in\text{-}codom\ (\le_{R})}$
⟨*proof*⟩

end

end

2.8.9 Order Equivalence

theory *Transport-Functions-Order-Equivalence*
imports
Transport-Functions-Monotone
Transport-Functions-Galois-Equivalence
begin

Dependent Function Relator **context** *transport-Dep-Fun-Rel*
begin

Inflationary **lemma** *rel-unit-self-if-rel-selfI*:
assumes *inflationary-unit1*: *inflationary-on* $(in\text{-}codom\ (\le_{L1}))\ (\le_{L1})\ \eta_1$
and *refl-L1*: *reflexive-on* $(in\text{-}codom\ (\le_{L1}))\ (\le_{L1})$
and *trans-L1*: *transitive* $(\le_{L1})$
and *mono-l2*: $\bigwedge x.\ x \le_{L1} x \Longrightarrow ((\le_{L2\ x\ x}) \Rrightarrow_m (\le_{R2\ (l1\ x)\ (l1\ x)}))\ (l2_{(l1\ x)\ x})$
and *mono-r2*: $\bigwedge x.\ x \le_{L1} x \Longrightarrow ((\le_{R2\ (l1\ x)\ (l1\ x)}) \Rrightarrow_m (\le_{L2\ x\ (\eta_1\ x)}))\ (r2_{x\ (l1\ x)})$
and *inflationary-unit2*: $\bigwedge x.\ x \le_{L1} x \Longrightarrow$
inflationary-on $(in\text{-}codom\ (\le_{L2\ x\ x}))\ (\le_{L2\ x\ x})\ (\eta_{2\ x\ (l1\ x)})$
and *L2-le1*: $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x2\ x2}) \le (\le_{L2\ x1\ x2})$
and *L2-unit-le2*: $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
and *ge-R2-l2-le2*: $\bigwedge x\ y.\ x \le_{L1} x \Longrightarrow in\text{-}codom\ (\le_{L2\ x\ (\eta_1\ x)})\ y \Longrightarrow$
$(\ge_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ x}\ y) \le (\ge_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ (\eta_1\ x)}\ y)$
and *trans-L2*: $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow transitive\ (\le_{L2\ x1\ x2})$
and $f \le_L f$
shows $f \le_L \eta\ f$
⟨*proof*⟩

Deflationary **interpretation** *flip-inv* :
transport-Dep-Fun-Rel $(\ge_{R1})\ (\ge_{L1})$ *r1 l1 flip2 R2 flip2 L2 r2 l2*
rewrites *flip-inv.L* $\equiv (\ge_R)$ **and** *flip-inv.R* $\equiv (\ge_L)$
and *flip-inv.unit* $\equiv \varepsilon$
and *flip-inv.t1.unit* $\equiv \varepsilon_1$
and $\bigwedge x\ y.$ *flip-inv.t2-unit* $x\ y \equiv \varepsilon_{2\ y\ x}$
and $\bigwedge R\ x\ y.\ (flip2\ R\ x\ y)^{-1} \equiv R\ y\ x$
and $\bigwedge R.\ in\text{-}codom\ R^{-1} \equiv in\text{-}dom\ R$
and $\bigwedge R\ x1\ x2.\ in\text{-}codom\ (flip2\ R\ x1\ x2) \equiv in\text{-}dom\ (R\ x2\ x1)$

and $\bigwedge x1\ x2\ x1'\ x2'.\ (flip2\ R2\ x1'\ x2' \Rrightarrow_m flip2\ L2\ x1\ x2) \equiv ((\le_{R2\ x2'\ x1'}) \Rrightarrow_m (\le_{L2\ x2\ x1}))$
and $\bigwedge x1\ x2\ x1'\ x2'.\ (flip2\ L2\ x1\ x2 \Rrightarrow_m flip2\ R2\ x1'\ x2') \equiv ((\le_{L2\ x2\ x1}) \Rrightarrow_m (\le_{R2\ x2'\ x1'}))$
and $\bigwedge P.\ inflationary\text{-}on\ P\ (\ge_{R1}) \equiv deflationary\text{-}on\ P\ (\le_{R1})$
and $\bigwedge P\ x.\ inflationary\text{-}on\ P\ (flip2\ R2\ x\ x) \equiv deflationary\text{-}on\ P\ (\le_{R2\ x\ x})$
and $\bigwedge x1\ x2\ x3\ x4.\ flip2\ R2\ x1\ x2 \le flip2\ R2\ x3\ x4 \equiv (\le_{R2\ x2\ x1}) \le (\le_{R2\ x4\ x3})$
and $\bigwedge(R :: 'z \Rightarrow \text{-})\ (P :: 'z \Rightarrow bool).\ reflexive\text{-}on\ P\ R^{-1} \equiv reflexive\text{-}on\ P\ R$
and $\bigwedge R.\ transitive\ R^{-1} \equiv transitive\ R$
and $\bigwedge x1'\ x2'.\ transitive\ (flip2\ R2\ x1'\ x2') \equiv transitive\ (\le_{R2\ x2'\ x1'})$
⟨*proof*⟩

lemma *counit-rel-self-if-rel-selfI*:
assumes $deflationary\text{-}on\ (in\text{-}dom\ (\le_{R1}))\ (\le_{R1})\ \varepsilon_1$
and $reflexive\text{-}on\ (in\text{-}dom\ (\le_{R1}))\ (\le_{R1})$
and $transitive\ (\le_{R1})$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow ((\le_{L2\ (r1\ x')\ (r1\ x')}) \Rrightarrow_m (\le_{R2\ (\varepsilon_1\ x')\ x'}))\ (l2_{\ x'\ (r1\ x')})$
and $\bigwedge x'\ x'.\ x' \le_{R1} x' \Longrightarrow ((\le_{R2\ x'\ x'}) \Rrightarrow_m (\le_{L2\ (r1\ x')\ (r1\ x')}))\ (r2_{(r1\ x')\ x'})$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow deflationary\text{-}on\ (in\text{-}dom\ (\le_{R2\ x'\ x'}))\ (\le_{R2\ x'\ x'})\ (\varepsilon_{2\ (r1\ x')\ x'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x1')\ x2'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x1'\ x1'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x'\ y'.\ x' \le_{R1} x' \Longrightarrow in\text{-}dom\ (\le_{R2\ (\varepsilon_1\ x')\ x'})\ y' \Longrightarrow$
$(\le_{L2\ (r1\ x')\ (r1\ x')})\ (r2_{(r1\ x')\ x'}\ y') \le (\le_{L2\ (r1\ x')\ (r1\ x')})\ (r2_{(r1\ x')\ (\varepsilon_1\ x')}\ y')$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow transitive\ (\le_{R2\ x1'\ x2'})$
and $g \le_R g$
shows $\varepsilon\ g \le_R g$
⟨*proof*⟩

Relational Equivalence **lemma** *bi-related-unit-self-if-rel-self-aux*:
assumes *rel-equiv-unit1*: $rel\text{-}equivalence\text{-}on\ (in\text{-}field\ (\le_{L1}))\ (\le_{L1})\ \eta_1$
and *mono-r2*: $\bigwedge x.\ x \le_{L1} x \Longrightarrow ((\le_{R2\ (l1\ x)\ (l1\ x)}) \Rrightarrow_m (\le_{L2\ x\ x}))\ (r2_{x\ (l1\ x)})$
and *rel-equiv-unit2*: $\bigwedge x.\ x \le_{L1} x \Longrightarrow$
$rel\text{-}equivalence\text{-}on\ (in\text{-}field\ (\le_{L2\ x\ x}))\ (\le_{L2\ x\ x})\ (\eta_{2\ x\ (l1\ x)})$
and *L2-le1*: $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x2\ x2}) \le (\le_{L2\ x1\ x2})$
and *L2-le2*: $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ x1}) \le (\le_{L2\ x1\ x2})$
and [*iff*]: $x \le_{L1} x$
shows $((\le_{R2\ (l1\ x)\ (l1\ x)}) \Rrightarrow_m (\le_{L2\ x\ (\eta_1\ x)}))\ (r2_{x\ (l1\ x)})$
and $((\le_{R2\ (l1\ x)\ (l1\ x)}) \Rrightarrow_m (\le_{L2\ (\eta_1\ x)\ x}))\ (r2_{x\ (l1\ x)})$
and $deflationary\text{-}on\ (in\text{-}dom\ (\le_{L2\ x\ x}))\ (\le_{L2\ x\ x})\ \eta_{2\ x\ (l1\ x)}$
and $inflationary\text{-}on\ (in\text{-}codom\ (\le_{L2\ x\ x}))\ (\le_{L2\ x\ x})\ \eta_{2\ x\ (l1\ x)}$
⟨*proof*⟩

interpretation *flip* : *transport-Dep-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2*
rewrites $flip.counit \equiv \eta$ **and** $flip.t1.counit \equiv \eta_1$
and $\bigwedge x\ y.\ flip.t2\text{-}counit\ x\ y \equiv \eta_{2\ y\ x}$
⟨*proof*⟩

lemma *bi-related-unit-self-if-rel-selfI*:
assumes *rel-equiv-unit1*: *rel-equivalence-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$ η_1
and *trans-L1*: *transitive* $(\le_{L1})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow ((\le_{L2\ x\ x}) \Rrightarrow_m (\le_{R2\ (l1\ x)\ (l1\ x)}))\ (l2_{(l1\ x)\ x})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow ((\le_{R2\ (l1\ x)\ (l1\ x)}) \Rrightarrow_m (\le_{L2\ x\ x}))\ (r2_{x\ (l1\ x)})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow$
rel-equivalence-on (*in-field* $(\le_{L2\ x\ x})$) $(\le_{L2\ x\ x})$ $(\eta_{2\ x\ (l1\ x)})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x2\ x2}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ (\eta_1\ x1)\ x2}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ x1}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x\ y.\ x \le_{L1} x \Longrightarrow$ *in-dom* $(\le_{L2\ (\eta_1\ x)\ x})\ y \Longrightarrow$
$(\le_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ x}\ y) \le (\le_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ (\eta_1\ x)}\ y)$
and $\bigwedge x\ y.\ x \le_{L1} x \Longrightarrow$ *in-codom* $(\le_{L2\ x\ (\eta_1\ x)})\ y \Longrightarrow$
$(\ge_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ x}\ y) \le (\ge_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ (\eta_1\ x)}\ y)$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
and $f \le_L f$
shows $f \equiv_L \eta\ f$
⟨*proof*⟩

Lemmas for Monotone Function Relator **lemma** *order-equivalence-if-order-equivalence-mono-assms*
assumes *order-equiv1*: $((\le_{L1}) \equiv_o (\le_{R1}))\ l1\ r1$
and *refl-R1*: *reflexive-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$
and *R2-counit-le1*: $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x1')\ x2'}) \le (\le_{R2\ x1'\ x2'})$
and *mono-l2*: $([x1'\ x2' :: (\le_{R1})] \Rrightarrow_m [x1\ x2 :: (\le_{L1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
$[$*in-field* $(\le_{L2\ x1\ (r1\ x2')})] \Rrightarrow (\le_{R2\ (l1\ x1)\ x2'}))\ l2$
and [*iff*]: $x1' \le_{R1} x2'$
shows $([$*in-dom* $(\le_{L2\ (r1\ x1')\ (r1\ x2')})] \Rrightarrow (\le_{R2\ x1'\ x2'}))\ (l2_{x1'\ (r1\ x1')})\ (l2_{x2'\ (r1\ x1')})$
and $([$*in-codom* $(\le_{L2\ (r1\ x1')\ (r1\ x2')})] \Rrightarrow (\le_{R2\ x1'\ x2'}))\ (l2_{x2'\ (r1\ x1')})\ (l2_{x2'\ (r1\ x2')})$
⟨*proof*⟩

lemma *order-equivalence-if-order-equivalence-mono-assms-rightI*:
assumes *order-equiv1*: $((\le_{L1}) \equiv_o (\le_{R1}))\ l1\ r1$
and *refl-L1*: *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and *L2-unit-le2*: $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
and *mono-r2*: $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
$[$*in-field* $(\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
and [*iff*]: $x1 \le_{L1} x2$
shows $([$*in-codom* $(\le_{R2\ (l1\ x1)\ (l1\ x2)})] \Rrightarrow (\le_{L2\ x1\ x2}))\ (r2_{x1\ (l1\ x2)})\ (r2_{x2\ (l1\ x2)})$
and $([$*in-dom* $(\le_{R2\ (l1\ x1)\ (l1\ x2)})] \Rrightarrow (\le_{L2\ x1\ x2}))\ (r2_{x1\ (l1\ x1)})\ (r2_{x1\ (l1\ x2)})$
⟨*proof*⟩

lemma *l2-unit-bi-rel-selfI*:
assumes *pre-equiv1*: $((\le_{L1}) \equiv_{pre} (\le_{R1}))\ l1\ r1$
and *mono-L2*:

$([x1\ x2 :: (\leq_{L1})] \Rrightarrow_m [x3\ x4 :: (\leq_{L1}) \mid (x2 \leq_{L1} x3 \wedge x4 \leq_{L1} \eta_1\ x3)] \Rrightarrow (\geq))$ $L2$
and *mono-R2*:
$([x1'\ x2' :: (\leq_{R1})] \Rrightarrow_m [x3'\ x4' :: (\leq_{R1}) \mid (x2' \leq_{R1} x3' \wedge x4' \leq_{R1} \varepsilon_1\ x3')] \Rrightarrow (\geq))$ $R2$
and *mono-l2*: $([x1'\ x2' :: (\leq_{R1})] \Rrightarrow_m [x1\ x2 :: (\leq_{L1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
$[\textit{in-field}\ (\leq_{L2\ x1\ (r1\ x2')})] \Rrightarrow (\leq_{R2\ (l1\ x1)\ x2'}))\ l2$
and $x \leq_{L1} x$
and *in-field* $(\leq_{L2\ x\ x})\ y$
shows $l2_{(l1\ x)\ (\eta_1\ x)}\ y \equiv_{R2\ (l1\ x)\ (l1\ x)} l2_{(l1\ x)\ x}\ y$
⟨*proof*⟩

lemma *r2-counit-bi-rel-selfI*:
assumes *pre-equiv1*: $((\leq_{L1}) \equiv_{pre} (\leq_{R1}))\ l1\ r1$
and *mono-L2*:
$([x1\ x2 :: (\leq_{L1})] \Rrightarrow_m [x3\ x4 :: (\leq_{L1}) \mid (x2 \leq_{L1} x3 \wedge x4 \leq_{L1} \eta_1\ x3)] \Rrightarrow (\geq))$ $L2$
and *mono-R2*:
$([x1'\ x2' :: (\leq_{R1})] \Rrightarrow_m [x3'\ x4' :: (\leq_{R1}) \mid (x2' \leq_{R1} x3' \wedge x4' \leq_{R1} \varepsilon_1\ x3')] \Rrightarrow (\geq))$ $R2$
and *mono-r2*: $([x1\ x2 :: (\leq_{L1})] \Rrightarrow_m [x1'\ x2' :: (\leq_{R1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
$[\textit{in-field}\ (\leq_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\leq_{L2\ x1\ (r1\ x2')}))\ r2$
and $x' \leq_{R1} x'$
and *in-field* $(\leq_{R2\ x'\ x'})\ y'$
shows $r2_{(r1\ x')\ (\varepsilon_1\ x')}\ y' \equiv_{L2\ (r1\ x')\ (r1\ x')} r2_{(r1\ x')\ x'}\ y'$
⟨*proof*⟩

end

Function Relator **context** *transport-Fun-Rel*
begin

corollary *rel-unit-self-if-rel-selfI*:
assumes *inflationary-on* (*in-codom* $(\leq_{L1})$) $(\leq_{L1})\ \eta_1$
and *reflexive-on* (*in-codom* $(\leq_{L1})$) $(\leq_{L1})$
and *transitive* $(\leq_{L1})$
and $((\leq_{L2}) \Rrightarrow_m (\leq_{R2}))\ l2$
and $((\leq_{R2}) \Rrightarrow_m (\leq_{L2}))\ r2$
and *inflationary-on* (*in-codom* $(\leq_{L2})$) $(\leq_{L2})\ \eta_2$
and *transitive* $(\leq_{L2})$
and $f \leq_L f$
shows $f \leq_L \eta\ f$
⟨*proof*⟩

corollary *counit-rel-self-if-rel-selfI*:
assumes *deflationary-on* (*in-dom* $(\leq_{R1})$) $(\leq_{R1})\ \varepsilon_1$
and *reflexive-on* (*in-dom* $(\leq_{R1})$) $(\leq_{R1})$
and *transitive* $(\leq_{R1})$

and $((\le_{L2}) \Rrightarrow_m (\le_{R2}))$ *l2*
and $((\le_{R2}) \Rrightarrow_m (\le_{L2}))$ *r2*
and *deflationary-on* (*in-dom* $(\le_{R2})$) $(\le_{R2})$ ε_2
and *transitive* $(\le_{R2})$
and $g \le_R g$
shows $\varepsilon\ g \le_R g$
⟨*proof*⟩

lemma *bi-related-unit-self-if-rel-selfI*:
assumes *rel-equivalence-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$ η_1
and *transitive* $(\le_{L1})$
and $((\le_{L2}) \Rrightarrow_m (\le_{R2}))$ *l2*
and $((\le_{R2}) \Rrightarrow_m (\le_{L2}))$ *r2*
and *rel-equivalence-on* (*in-field* $(\le_{L2})$) $(\le_{L2})$ η_2
and *transitive* $(\le_{L2})$
and $f \le_L f$
shows $f \equiv_L \eta\ f$
⟨*proof*⟩

end

Monotone Dependent Function Relator **context** *transport-Mono-Dep-Fun-Rel*
begin

Inflationary **lemma** *inflationary-on-unitI*:
assumes $(tdfr.L \Rrightarrow_m tdfr.R)$ *l* **and** $(tdfr.R \Rrightarrow_m tdfr.L)$ *r*
and *inflationary-on* (*in-codom* $(\le_{L1})$) $(\le_{L1})$ η_1
and *reflexive-on* (*in-codom* $(\le_{L1})$) $(\le_{L1})$
and *transitive* $(\le_{L1})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow ((\le_{L2\ x\ x}) \Rrightarrow_m (\le_{R2\ (l1\ x)\ (l1\ x)}))\ (l2_{(l1\ x)\ x})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow ((\le_{R2\ (l1\ x)\ (l1\ x)}) \Rrightarrow_m (\le_{L2\ x\ (\eta_1\ x)}))\ (r2_{x\ (l1\ x)})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow$ *inflationary-on* (*in-codom* $(\le_{L2\ x\ x})$) $(\le_{L2\ x\ x})$ $(\eta_{2\ x\ (l1\ x)})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x2\ x2}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x\ y.\ x \le_{L1} x \Longrightarrow$ *in-codom* $(\le_{L2\ x\ (\eta_1\ x)})\ y \Longrightarrow$
$(\ge_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ x}\ y) \le (\ge_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ (\eta_1\ x)}\ y)$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
shows *inflationary-on* (*in-field* $(\le_L)$) $(\le_L)$ η
⟨*proof*⟩

Deflationary **lemma** *deflationary-on-counitI*:
assumes $(tdfr.L \Rrightarrow_m tdfr.R)$ *l* **and** $(tdfr.R \Rrightarrow_m tdfr.L)$ *r*
and *deflationary-on* (*in-dom* $(\le_{R1})$) $(\le_{R1})$ ε_1
and *reflexive-on* (*in-dom* $(\le_{R1})$) $(\le_{R1})$
and *transitive* $(\le_{R1})$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow ((\le_{L2\ (r1\ x')\ (r1\ x')}) \Rrightarrow_m (\le_{R2\ (\varepsilon_1\ x')\ x'}))\ (l2_{\ x'\ (r1\ x')})$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow$
$((\le_{R2\ x'\ x'}) \Rrightarrow_m (\le_{L2\ (r1\ x')\ (r1\ x')}))\ (r2_{(r1\ x')\ x'})$

and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow$ *deflationary-on* (*in-dom* $(\le_{R2\ x'\ x'})$) $(\le_{R2\ x'\ x'})$ $(\varepsilon_{2\ (r1\ x')\ x'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x1')\ x2'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x1'\ x1'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x'\ y'.\ x' \le_{R1} x' \Longrightarrow$ *in-dom* $(\le_{R2\ (\varepsilon_1\ x')\ x'})\ y' \Longrightarrow$
$(\le_{L2\ (r1\ x')\ (r1\ x')})\ (r2_{(r1\ x')\ x'}\ y') \le (\le_{L2\ (r1\ x')\ (r1\ x')})\ (r2_{(r1\ x')\ (\varepsilon_1\ x')}$
$y')$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$ *transitive* $(\le_{R2\ x1'\ x2'})$
shows *deflationary-on* (*in-field* $(\le_R)$) $(\le_R)\ \varepsilon$
⟨*proof*⟩

Relational Equivalence **context**
begin

interpretation *flip* : *transport-Mono-Dep-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2*
rewrites *flip.counit* $\equiv \eta$ **and** *flip.t1.counit* $\equiv \eta_1$
and $\bigwedge x\ y.$ *flip.t2-counit* $x\ y \equiv \eta_{2\ y\ x}$
⟨*proof*⟩

lemma *rel-equivalence-on-unitI*:
assumes $(tdfr.L \Rrightarrow_m tdfr.R)\ l$ **and** $(tdfr.R \Rrightarrow_m tdfr.L)\ r$
and *rel-equiv-unit1*: *rel-equivalence-on* (*in-field* $(\le_{L1})$) $(\le_{L1})\ \eta_1$
and *trans-L1*: *transitive* $(\le_{L1})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow ((\le_{L2\ x\ x}) \Rrightarrow_m (\le_{R2\ (l1\ x)\ (l1\ x)}))\ (l2_{(l1\ x)\ x})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow ((\le_{R2\ (l1\ x)\ (l1\ x)}) \Rrightarrow_m (\le_{L2\ x\ x}))\ (r2_{x\ (l1\ x)})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow$ *rel-equivalence-on* (*in-field* $(\le_{L2\ x\ x})$) $(\le_{L2\ x\ x})\ (\eta_{2\ x\ (l1\ x)})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x2\ x2}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ (\eta_1\ x1)\ x2}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ x1}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x\ y.\ x \le_{L1} x \Longrightarrow$ *in-dom* $(\le_{L2\ (\eta_1\ x)\ x})\ y \Longrightarrow$
$(\le_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ x}\ y) \le (\le_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ (\eta_1\ x)}\ y)$
and $\bigwedge x\ y.\ x \le_{L1} x \Longrightarrow$ *in-codom* $(\le_{L2\ x\ (\eta_1\ x)})\ y \Longrightarrow$
$(\ge_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ x}\ y) \le (\ge_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ (\eta_1\ x)}\ y)$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
shows *rel-equivalence-on* (*in-field* $(\le_L)$) $(\le_L)\ \eta$
⟨*proof*⟩

end

Order Equivalence **interpretation** *flip* : *transport-Mono-Dep-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2*
rewrites *flip.unit* $\equiv \varepsilon$ **and** *flip.t1.unit* $\equiv \varepsilon_1$
and *flip.counit* $\equiv \eta$ **and** *flip.t1.counit* $\equiv \eta_1$
and $\bigwedge x\ y.$ *flip.t2-unit* $x\ y \equiv \varepsilon_{2\ y\ x}$
⟨*proof*⟩

lemma *order-equivalenceI*:

assumes $(tdfr.L \Rrightarrow_m tdfr.R)\ l$ **and** $(tdfr.R \Rrightarrow_m tdfr.L)\ r$
and *rel-equivalence-on* $(\textit{in-field}\ (\le_{L1}))\ (\le_{L1})\ \eta_1$
and *rel-equivalence-on* $(\textit{in-field}\ (\le_{R1}))\ (\le_{R1})\ \varepsilon_1$
and *transitive* $(\le_{L1})$ **and** *transitive* $(\le_{R1})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow ((\le_{L2\ x\ x}) \Rrightarrow_m (\le_{R2\ (l1\ x)\ (l1\ x)}))\ (l2_{(l1\ x)\ x})$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow ((\le_{L2\ (r1\ x')\ (r1\ x')}) \Rrightarrow_m (\le_{R2\ x'\ x'}))\ (l2_{x'\ (r1\ x')})$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow ((\le_{R2\ x'\ x'}) \Rrightarrow_m (\le_{L2\ (r1\ x')\ (r1\ x')}))\ (r2_{(r1\ x')\ x'})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow ((\le_{R2\ (l1\ x)\ (l1\ x)}) \Rrightarrow_m (\le_{L2\ x\ x}))\ (r2_{x\ (l1\ x)})$
and $\bigwedge x.\ x \le_{L1} x \Longrightarrow$ *rel-equivalence-on* $(\textit{in-field}\ (\le_{L2\ x\ x}))\ (\le_{L2\ x\ x})\ (\eta_{2\ x\ (l1\ x)})$
and $\bigwedge x'.\ x' \le_{R1} x' \Longrightarrow$
rel-equivalence-on $(\textit{in-field}\ (\le_{R2\ x'\ x'}))\ (\le_{R2\ x'\ x'})\ (\varepsilon_{2\ (r1\ x')\ x'})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x2\ x2}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ (\eta_1\ x1)\ x2}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ x1}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x2'\ x2'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x1')\ x2'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x1'\ x1'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x1'\ (\varepsilon_1\ x2')}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x\ y.\ x \le_{L1} x \Longrightarrow \textit{in-dom}\ (\le_{L2\ (\eta_1\ x)\ x})\ y \Longrightarrow$
$(\le_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ x}\ y) \le (\le_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ (\eta_1\ x)}\ y)$
and $\bigwedge x\ y.\ x \le_{L1} x \Longrightarrow \textit{in-codom}\ (\le_{L2\ x\ (\eta_1\ x)})\ y \Longrightarrow$
$(\ge_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ x}\ y) \le (\ge_{R2\ (l1\ x)\ (l1\ x)})\ (l2_{(l1\ x)\ (\eta_1\ x)}\ y)$
and $\bigwedge x'\ y'.\ x' \le_{R1} x' \Longrightarrow \textit{in-dom}\ (\le_{R2\ (\varepsilon_1\ x')\ x'})\ y' \Longrightarrow$
$(\le_{L2\ (r1\ x')\ (r1\ x')})\ (r2_{(r1\ x')\ x'}\ y') \le (\le_{L2\ (r1\ x')\ (r1\ x')})\ (r2_{(r1\ x')\ (\varepsilon_1\ x')}$
$y')$
and $\bigwedge x'\ y'.\ x' \le_{R1} x' \Longrightarrow \textit{in-codom}\ (\le_{R2\ x'\ (\varepsilon_1\ x')})\ y' \Longrightarrow$
$(\ge_{L2\ (r1\ x')\ (r1\ x')})\ (r2_{(r1\ x')\ x'}\ y') \le (\ge_{L2\ (r1\ x')\ (r1\ x')})\ (r2_{(r1\ x')\ (\varepsilon_1\ x')}$
$y')$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_{R1} x2 \Longrightarrow$ *transitive* $(\le_{R2\ x1\ x2})$
shows $((\le_L) \equiv_o (\le_R))\ l\ r$
$\langle proof \rangle$

lemma *order-equivalence-if-preorder-equivalenceI*:
assumes *pre-equiv1*: $((\le_{L1}) \equiv_{pre} (\le_{R1}))\ l1\ r1$
and *order-equiv2*: $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow$
$((\le_{L2\ x\ (r1\ x')}) \equiv_o (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and *L2-les*: $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x2\ x2}) \le (\le_{L2\ x1\ x2})$
$\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ (\eta_1\ x1)\ x2}) \le (\le_{L2\ x1\ x2})$
$\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ x1}) \le (\le_{L2\ x1\ x2})$
$\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
and *R2-les*: $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x2'\ x2'}) \le (\le_{R2\ x1'\ x2'})$
$\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x1')\ x2'}) \le (\le_{R2\ x1'\ x2'})$
$\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x1'\ x1'}) \le (\le_{R2\ x1'\ x2'})$

$\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x1'\ (\varepsilon_1\ x2')}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$
$([in\text{-}dom\ (\le_{L2\ (r1\ x1')\ (r1\ x2')})] \Rrightarrow (\le_{R2\ x1'\ x2'}))\ (l2_{x1'\ (r1\ x1')})\ (l2_{x2'\ (r1\ x1')})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$
$([in\text{-}codom\ (\le_{L2\ (r1\ x1')\ (r1\ x2')})] \Rrightarrow (\le_{R2\ x1'\ x2'}))\ (l2_{x2'\ (r1\ x1')})\ (l2_{x2'\ (r1\ x2')})$
and *l2-bi-rel*: $\bigwedge x\ y.\ x \le_{L1} x \Longrightarrow in\text{-}field\ (\le_{L2\ x\ x})\ y \Longrightarrow$
$l2_{(l1\ x)\ (\eta_1\ x)}\ y \equiv_{R2\ (l1\ x)\ (l1\ x)} l2_{(l1\ x)\ x}\ y$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$
$([in\text{-}codom\ (\le_{R2\ (l1\ x1)\ (l1\ x2)})] \Rrightarrow (\le_{L2\ x1\ x2}))\ (r2_{x1\ (l1\ x2)})\ (r2_{x2\ (l1\ x2)})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$
$([in\text{-}dom\ (\le_{R2\ (l1\ x1)\ (l1\ x2)})] \Rrightarrow (\le_{L2\ x1\ x2}))\ (r2_{x1\ (l1\ x1)})\ (r2_{x1\ (l1\ x2)})$
and *r2-bi-rel*: $\bigwedge x'\ y'.\ x' \le_{R1} x' \Longrightarrow in\text{-}field\ (\le_{R2\ x'\ x'})\ y' \Longrightarrow$
$r2_{(r1\ x')\ (\varepsilon_1\ x')}\ y' \equiv_{L2\ (r1\ x')\ (r1\ x')} r2_{(r1\ x')\ x'}\ y'$
and *trans-L2*: $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow transitive\ (\le_{L2\ x1\ x2})$
and *trans-R2*: $\bigwedge x1\ x2.\ x1 \le_{R1} x2 \Longrightarrow transitive\ (\le_{R2\ x1\ x2})$
shows $((\le_L) \equiv_o (\le_R))\ l\ r$
⟨*proof*⟩

lemma *order-equivalence-if-preorder-equivalenceI′*:
assumes $((\le_{L1}) \equiv_{pre} (\le_{R1}))\ l1\ r1$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \equiv_o (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x2\ x2}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ (\eta_1\ x1)\ x2}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ x1}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow (\le_{L2\ x1\ (\eta_1\ x2)}) \le (\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x2'\ x2'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ (\varepsilon_1\ x1')\ x2'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x1'\ x1'}) \le (\le_{R2\ x1'\ x2'})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow (\le_{R2\ x1'\ (\varepsilon_1\ x2')}) \le (\le_{R2\ x1'\ x2'})$
and $([x1'\ x2' :: (\le_{R1})] \Rrightarrow_m [x1\ x2 :: (\le_{L1})\ |\ x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
$[in\text{-}field\ (\le_{L2\ x1\ (r1\ x2')})] \Rrightarrow (\le_{R2\ (l1\ x1)\ x2'}))\ l2$
and $\bigwedge x\ y.\ x \le_{L1} x \Longrightarrow in\text{-}field\ (\le_{L2\ x\ x})\ y \Longrightarrow$
$l2_{(l1\ x)\ (\eta_1\ x)}\ y \equiv_{R2\ (l1\ x)\ (l1\ x)} l2_{(l1\ x)\ x}\ y$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1})\ |\ x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
$[in\text{-}field\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
and $\bigwedge x'\ y'.\ x' \le_{R1} x' \Longrightarrow in\text{-}field\ (\le_{R2\ x'\ x'})\ y' \Longrightarrow$
$r2_{(r1\ x')\ (\varepsilon_1\ x')}\ y' \equiv_{L2\ (r1\ x')\ (r1\ x')} r2_{(r1\ x')\ x'}\ y'$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow transitive\ (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_{R1} x2 \Longrightarrow transitive\ (\le_{R2\ x1\ x2})$
shows $((\le_L) \equiv_o (\le_R))\ l\ r$
⟨*proof*⟩

lemma *order-equivalence-if-mono-if-preorder-equivalenceI*:
assumes $((\le_{L1}) \equiv_{pre} (\le_{R1}))\ l1\ r1$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \equiv_o (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $([x1\ x2 :: (\le_{L1})\ |\ \eta_1\ x2 \le_{L1} x1] \Rrightarrow_m [x3\ x4 :: (\le_{L1})\ |\ x2 \le_{L1} x3] \Rrightarrow (\le))$

$L2$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid (x2 \le_{L1} x3 \wedge x4 \le_{L1} \eta_1\ x3)] \Rrightarrow (\ge))\ L2$
and $([x1'\ x2' :: (\le_{R1}) \mid \varepsilon_1\ x2' \le_{R1} x1'] \Rrightarrow_m [x3'\ x4' :: (\le_{R1}) \mid x2' \le_{R1} x3'] \Rrightarrow (\le))\ R2$
and $([x1'\ x2' :: (\le_{R1})] \Rrightarrow_m [x3'\ x4' :: (\le_{R1}) \mid (x2' \le_{R1} x3' \wedge x4' \le_{R1} \varepsilon_1\ x3')] \Rrightarrow (\ge))\ R2$
and $([x1'\ x2' :: (\le_{R1})] \Rrightarrow_m [x1\ x2 :: (\le_{L1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow [\textit{in-field}\ (\le_{L2\ x1\ (r1\ x2')})] \Rrightarrow (\le_{R2\ (l1\ x1)\ x2'}))\ l2$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow [\textit{in-field}\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow \textit{transitive}\ (\le_{L2\ x1\ x2})$
and $\bigwedge x1\ x2.\ x1 \le_{R1} x2 \Longrightarrow \textit{transitive}\ (\le_{R2\ x1\ x2})$
shows $((\le_L) \equiv_o (\le_R))\ l\ r$
⟨*proof*⟩

theorem *order-equivalence-if-mono-if-preorder-equivalenceI'*:
assumes $((\le_{L1}) \equiv_{pre} (\le_{R1}))\ l1\ r1$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \equiv_{pre} (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $([x1\ x2 :: (\ge_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid x1 \le_{L1} x3] \Rrightarrow (\le))\ L2$
and $([x1'\ x2' :: (\ge_{R1})] \Rrightarrow_m [x3'\ x4' :: (\le_{R1}) \mid x1' \le_{R1} x3'] \Rrightarrow (\le))\ R2$
and $([x1'\ x2' :: (\le_{R1})] \Rrightarrow_m [x1\ x2 :: (\le_{L1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow [\textit{in-field}\ (\le_{L2\ x1\ (r1\ x2')})] \Rrightarrow (\le_{R2\ (l1\ x1)\ x2'}))\ l2$
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow [\textit{in-field}\ (\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))\ r2$
shows $((\le_L) \equiv_o (\le_R))\ l\ r$
⟨*proof*⟩

end

Monotone Function Relator **context** *transport-Mono-Fun-Rel*
begin

interpretation *flip* : *transport-Mono-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2* ⟨*proof*⟩

lemma *inflationary-on-unitI*:
assumes $((\le_{L1}) \Rrightarrow_m (\le_{R1}))\ l1$
and $((\le_{R1}) \Rrightarrow_m (\le_{L1}))\ r1$
and $\textit{inflationary-on}\ (\textit{in-codom}\ (\le_{L1}))\ (\le_{L1})\ \eta_1$
and $\textit{reflexive-on}\ (\textit{in-codom}\ (\le_{L1}))\ (\le_{L1})$
and $\textit{transitive}\ (\le_{L1})$
and $((\le_{L2}) \Rrightarrow_m (\le_{R2}))\ l2$
and $((\le_{R2}) \Rrightarrow_m (\le_{L2}))\ r2$
and $\textit{inflationary-on}\ (\textit{in-codom}\ (\le_{L2}))\ (\le_{L2})\ \eta_2$
and $\textit{transitive}\ (\le_{L2})$
shows $\textit{inflationary-on}\ (\textit{in-field}\ (\le_L))\ (\le_L)\ \eta$
⟨*proof*⟩

lemma *deflationary-on-counitI*:
assumes $((\leq_{L1}) \Rrightarrow_m (\leq_{R1}))$ *l1*
and $((\leq_{R1}) \Rrightarrow_m (\leq_{L1}))$ *r1*
and *deflationary-on* (*in-dom* $(\leq_{R1})$) $(\leq_{R1})$ ε_1
and *reflexive-on* (*in-dom* $(\leq_{R1})$) $(\leq_{R1})$
and *transitive* $(\leq_{R1})$
and $((\leq_{L2}) \Rrightarrow_m (\leq_{R2}))$ *l2*
and $((\leq_{R2}) \Rrightarrow_m (\leq_{L2}))$ *r2*
and *deflationary-on* (*in-dom* $(\leq_{R2})$) $(\leq_{R2})$ ε_2
and *transitive* $(\leq_{R2})$
shows *deflationary-on* (*in-field* $(\leq_R)$) $(\leq_R)$ ε
⟨*proof*⟩

lemma *rel-equivalence-on-unitI*:
assumes $((\leq_{L1}) \Rrightarrow_m (\leq_{R1}))$ *l1*
and $((\leq_{R1}) \Rrightarrow_m (\leq_{L1}))$ *r1*
and *rel-equivalence-on* (*in-field* $(\leq_{L1})$) $(\leq_{L1})$ η_1
and *transitive* $(\leq_{L1})$
and $((\leq_{L2}) \Rrightarrow_m (\leq_{R2}))$ *l2*
and $((\leq_{R2}) \Rrightarrow_m (\leq_{L2}))$ *r2*
and *rel-equivalence-on* (*in-field* $(\leq_{L2})$) $(\leq_{L2})$ η_2
and *transitive* $(\leq_{L2})$
shows *rel-equivalence-on* (*in-field* $(\leq_L)$) $(\leq_L)$ η
⟨*proof*⟩

lemma *order-equivalenceI*:
assumes $((\leq_{L1}) \equiv_{pre} (\leq_{R1}))$ *l1 r1*
and $((\leq_{L2}) \equiv_{pre} (\leq_{R2}))$ *l2 r2*
shows $((\leq_L) \equiv_o (\leq_R))$ *l r*
⟨*proof*⟩

end

end

theory *Transport-Functions*
imports
Transport-Functions-Galois-Equivalence
Transport-Functions-Galois-Relator
Transport-Functions-Order-Base
Transport-Functions-Order-Equivalence
Transport-Functions-Relation-Simplifications
begin

Summary Composition under (dependent) (monotone) function relators. Refer to [2] for more details.

2.8.10 Summary of Main Results

More precise results can be found in the corresponding subtheories.

Monotone Dependent Function Relator **context** *transport-Mono-Dep-Fun-Rel*
begin

interpretation *flip* : *transport-Mono-Dep-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2*
rewrites *flip.t1.counit* $\equiv \eta_1$ **and** *flip.t1.unit* $\equiv \varepsilon_1$
⟨*proof*⟩

Closure of Order and Galois Concepts **theorem** *preorder-galois-connection-if-galois-connectionI*:
assumes $((\le_{L1}) \dashv (\le_{R1}))$ *l1 r1*
and *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$
and *reflexive-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \dashv (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $([\text{-}\ x2 :: (\le_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid (x2 \le_{L1} x3 \wedge x4 \le_{L1} \eta_1\ x3)] \Rrightarrow (\ge))$
L2
and $([x1'\ x2' :: (\le_{R1}) \mid \varepsilon_1\ x2' \le_{R1} x1'] \Rrightarrow_m [x3'\ \text{-} :: (\le_{R1}) \mid x2' \le_{R1} x3'] \Rrightarrow$
$(\le))$ *R2*
and $([x1'\ x2' :: (\le_{R1})] \Rrightarrow_m [x1\ x2 :: (\le_{L1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
[*in-field* $(\le_{L2\ x1\ (r1\ x2')})] \Rrightarrow (\le_{R2\ (l1\ x1)\ x2'}))$ *l2*
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
[*in-field* $(\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))$ *r2*
and $\bigwedge x1\ x2.\ x1 \le_{L1} x2 \Longrightarrow$ *transitive* $(\le_{L2\ x1\ x2})$
and $\bigwedge x1'\ x2'.\ x1' \le_{R1} x2' \Longrightarrow$ *transitive* $(\le_{R2\ x1'\ x2'})$
shows $((\le_L) \dashv_{pre} (\le_R))$ *l r*
⟨*proof*⟩

theorem *preorder-equivalenceI*:
assumes $((\le_{L1}) \equiv_{pre} (\le_{R1}))$ *l1 r1*
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \equiv_{pre} (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $([x1\ \text{-} :: (\ge_{L1})] \Rrightarrow_m [x3\ \text{-} :: (\le_{L1}) \mid x1 \le_{L1} x3] \Rrightarrow (\le))$ *L2*
and $([x1'\ \text{-} :: (\ge_{R1})] \Rrightarrow_m [x3'\ \text{-} :: (\le_{R1}) \mid x1' \le_{R1} x3'] \Rrightarrow (\le))$ *R2*
and $([x1'\ x2' :: (\le_{R1})] \Rrightarrow_m [x1\ x2 :: (\le_{L1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
[*in-field* $(\le_{L2\ x1\ (r1\ x2')})] \Rrightarrow (\le_{R2\ (l1\ x1)\ x2'}))$ *l2*
and $([x1\ x2 :: (\le_{L1})] \Rrightarrow_m [x1'\ x2' :: (\le_{R1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
[*in-field* $(\le_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\le_{L2\ x1\ (r1\ x2')}))$ *r2*
shows $((\le_L) \equiv_{pre} (\le_R))$ *l r*
⟨*proof*⟩

theorem *partial-equivalence-rel-equivalenceI*:
assumes $((\le_{L1}) \equiv_{PER} (\le_{R1}))$ *l1 r1*
and $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \equiv_{PER} (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
and $([x1\ \text{-} :: (\ge_{L1})] \Rrightarrow_m [x3\ \text{-} :: (\le_{L1}) \mid x1 \le_{L1} x3] \Rrightarrow (\le))$ *L2*
and $([x1'\ \text{-} :: (\ge_{R1})] \Rrightarrow_m [x3'\ \text{-} :: (\le_{R1}) \mid x1' \le_{R1} x3'] \Rrightarrow (\le))$ *R2*
and $([x1'\ x2' :: (\le_{R1})] \Rrightarrow_m [x1\ x2 :: (\le_{L1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow$
[*in-field* $(\le_{L2\ x1\ (r1\ x2')})] \Rrightarrow (\le_{R2\ (l1\ x1)\ x2'}))$ *l2*

and $([x1\ x2 :: (\leq_{L1})] \Rrightarrow_m [x1'\ x2' :: (\leq_{R1}) \mid x2\ {}_{L1}\lessapprox x1'] \Rrightarrow [\textit{in-field}\ (\leq_{R2\ (l1\ x1)\ x2'})] \Rrightarrow (\leq_{L2\ x1\ (r1\ x2')}))\ r2$
shows $((\leq_L) \equiv_{PER} (\leq_R))\ l\ r$
⟨*proof*⟩

Simplification of Left and Right Relations See $[\![t1.\textit{galois-equivalence}; \textit{preorder-on}\ (\textit{in-field}\ (\leq_{L1}))\ (\leq_{L1}); ([x1\ x2 :: (\leq_{L1})^{-1}] \Rrightarrow_m [x3\ x4 :: (\leq_{L1})] \Rrightarrow x1 \leq_{L1} x3 \longrightarrow (\leq))\ L2; \bigwedge x1\ x2.\ x1 \leq_{L1} x2 \Longrightarrow \textit{partial-equivalence-rel}\ (\leq_{L2\ x1\ x2})]\!] \Longrightarrow \textit{flip}.R = \textit{flip.tdfr}.R.$

Simplification of Galois relator See $[\![t1.\textit{galois-connection}; \textit{reflexive-on}\ (\textit{in-field}\ (\leq_{L1}))\ (\leq_{L1}); \bigwedge x\ x'.\ \textit{flip.t1.right-Galois}\ x\ x' \Longrightarrow (\leq_{R2\ l1\ x\ x'} \Rrightarrow_m \leq_{L2\ x\ r1\ x'})\ r2_{x\ x'}; ([x1 :: \top] \Rrightarrow_m [x2\ \text{-} :: (\leq_{L1})] \Rrightarrow_m x1 \leq_{L1} x2 \longrightarrow (\leq))\ L2; ([x1 :: \top] \Rrightarrow_m [x2\ x3 :: (\leq_{L1})] \Rrightarrow_m (x1 \leq_{L1} x2 \wedge x3 \leq_{L1} \eta_1\ x2) \longrightarrow (\lambda x\ y.\ y \leq x))\ L2; ([x1\ x2 :: (\leq_{L1})] \Rrightarrow_m [x1'\ x2' :: (\leq_{R1})] \Rrightarrow \textit{flip.t1.right-Galois}\ x2\ x1' \longrightarrow ([\textit{in-field}\ (\leq_{R2\ l1\ x1\ x2'})] \Rrightarrow \leq_{L2\ x1\ r1\ x2'}))\ r2; \bigwedge x1\ x2.\ x1 \leq_{L1} x2 \Longrightarrow \textit{transitive}\ (\leq_{L2\ x1\ x2})]\!] \Longrightarrow \textit{flip.right-Galois} = (\textit{Dep-Fun-Rel-rel flip.t1.right-Galois t2.left-Galois})\restriction_{\textit{in-dom flip}.R}\upharpoonright_{\textit{in-codom flip}.L}$

$[\![t1.\textit{preorder-equivalence}; \bigwedge x\ x'.\ \textit{flip.t1.right-Galois}\ x\ x' \Longrightarrow (\leq_{R2\ l1\ x\ x'} \Rrightarrow_m \leq_{L2\ x\ r1\ x'})\ r2_{x\ x'}; ([x1\ x2 :: (\leq_{L1})^{-1}] \Rrightarrow_m [x3\ x4 :: (\leq_{L1})] \Rrightarrow x1 \leq_{L1} x3 \longrightarrow (\leq))\ L2; ([x1\ x2 :: (\leq_{L1})] \Rrightarrow_m [x1'\ x2' :: (\leq_{R1})] \Rrightarrow \textit{flip.t1.right-Galois}\ x2\ x1' \longrightarrow ([\textit{in-field}\ (\leq_{R2\ l1\ x1\ x2'})] \Rrightarrow \leq_{L2\ x1\ r1\ x2'}))\ r2; \bigwedge x1\ x2.\ x1 \leq_{L1} x2 \Longrightarrow \textit{transitive}\ (\leq_{L2\ x1\ x2})]\!] \Longrightarrow \textit{flip.right-Galois} = (\textit{Dep-Fun-Rel-rel flip.t1.right-Galois t2.left-Galois})\restriction_{\textit{in-dom flip}.R}\upharpoonright_{\textit{in-codom flip}.L}$

$[\![t1.\textit{preorder-equivalence}; \bigwedge x\ x'.\ \textit{flip.t1.right-Galois}\ x\ x' \Longrightarrow t2.\textit{preorder-equivalence}\ x\ x'; ([x1\ x2 :: (\leq_{L1})^{-1}] \Rrightarrow_m [x3\ x4 :: (\leq_{L1})] \Rrightarrow x1 \leq_{L1} x3 \longrightarrow (\leq))\ L2; ([x1\ x2 :: (\leq_{L1})] \Rrightarrow_m [x1'\ x2' :: (\leq_{R1})] \Rrightarrow \textit{flip.t1.right-Galois}\ x2\ x1' \longrightarrow ([\textit{in-field}\ (\leq_{R2\ l1\ x1\ x2'})] \Rrightarrow \leq_{L2\ x1\ r1\ x2'}))\ r2]\!] \Longrightarrow \textit{flip.right-Galois} = (\textit{Dep-Fun-Rel-rel flip.t1.right-Galois t2.left-Galois})\restriction_{\textit{in-dom flip}.R}\upharpoonright_{\textit{in-codom flip}.L}$

$[\![t1.\textit{preorder-equivalence}; \bigwedge x1'\ x2'.\ x1' \leq_{R1} x2' \Longrightarrow ((\leq_{L2\ r1\ x1'\ r1\ x2'})\ {}_h\trianglelefteq\ (\leq_{R2\ \varepsilon_1\ x1'\ x2'}))\ l2_{x2'\ r1\ x1'}\ r2_{r1\ x1'\ x2'}; ([x1\ x2 :: (\leq_{L1})^{-1}] \Rrightarrow_m [x3\ x4 :: (\leq_{L1})] \Rrightarrow x1 \leq_{L1} x3 \longrightarrow (\leq))\ L2; ([x1'\ x2' :: (\leq_{R1})^{-1}] \Rrightarrow_m [x3'\ x4' :: (\leq_{R1})] \Rrightarrow x1' \leq_{R1} x3' \longrightarrow (\leq))\ R2; ([x1'\ x2' :: (\leq_{R1})] \Rrightarrow_m [x1\ x2 :: (\leq_{L1})] \Rrightarrow \textit{flip.t1.right-Galois}\ x2\ x1' \longrightarrow ([\textit{in-field}\ (\leq_{L2\ x1\ r1\ x2'})] \Rrightarrow \leq_{R2\ l1\ x1\ x2'}))\ l2; ([x1\ x2 :: (\leq_{L1})] \Rrightarrow_m [x1'\ x2' :: (\leq_{R1})] \Rrightarrow \textit{flip.t1.right-Galois}\ x2\ x1' \longrightarrow ([\textit{in-field}\ (\leq_{R2\ l1\ x1\ x2'})] \Rrightarrow \leq_{L2\ x1\ r1\ x2'}))\ r2; \bigwedge x1\ x2.\ x1 \leq_{L1} x2 \Longrightarrow \textit{partial-equivalence-rel}\ (\leq_{L2\ x1\ x2}); \bigwedge x1'\ x2'.\ x1' \leq_{R1} x2' \Longrightarrow \textit{partial-equivalence-rel}\ (\leq_{R2\ x1'\ x2'})]\!] \Longrightarrow (\textit{Dep-Fun-Rel-rel flip.t1.right-Galois t2.left-Galois})\restriction_{\textit{in-dom flip}.R}\upharpoonright_{\textit{in-codom flip}.L} = \textit{Dep-Fun-Rel-rel flip.t1.right-Galois t2.left-Galois}$

$t1.\textit{half-galois-prop-left} \Longrightarrow ([x\ x' :: \textit{flip.t1.right-Galois}] \Rrightarrow (?S\ x\ x')\restriction_{\textit{in-dom}\ (\leq_{L2\ x\ r1\ x'})}\upharpoonright_{\textit{in-codom}\ (\leq}$ $= (\textit{Dep-Fun-Rel-rel flip.t1.right-Galois}\ ?S)\restriction_{\textit{in-dom flip}.R}\upharpoonright_{\textit{in-codom flip}.L}$

end

Monotone Function Relator **context** *transport-Mono-Fun-Rel*
begin

interpretation *flip* : *transport-Mono-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2* ⟨*proof*⟩

Closure of Order and Galois Concepts **lemma** *preorder-galois-connection-if-galois-connectionI*:
assumes $((\le_{L1}) \dashv (\le_{R1}))$ *l1 r1*
and *reflexive-on* (*in-codom* $(\le_{L1})$) $(\le_{L1})$ *reflexive-on* (*in-dom* $(\le_{R1})$) $(\le_{R1})$
and $((\le_{L2}) \dashv (\le_{R2}))$ *l2 r2*
and *transitive* $(\le_{L2})$ *transitive* $(\le_{R2})$
shows $((\le_{L}) \dashv_{pre} (\le_{R}))$ *l r*
⟨*proof*⟩

theorem *preorder-galois-connectionI*:
assumes $((\le_{L1}) \dashv_{pre} (\le_{R1}))$ *l1 r1*
and $((\le_{L2}) \dashv_{pre} (\le_{R2}))$ *l2 r2*
shows $((\le_{L}) \dashv_{pre} (\le_{R}))$ *l r*
⟨*proof*⟩

theorem *preorder-equivalence-if-galois-equivalenceI*:
assumes $((\le_{L1}) \equiv_{G} (\le_{R1}))$ *l1 r1*
and *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$ *reflexive-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$
and $((\le_{L2}) \equiv_{G} (\le_{R2}))$ *l2 r2*
and *transitive* $(\le_{L2})$ *transitive* $(\le_{R2})$
shows $((\le_{L}) \equiv_{pre} (\le_{R}))$ *l r*
⟨*proof*⟩

theorem *preorder-equivalenceI*:
assumes $((\le_{L1}) \equiv_{pre} (\le_{R1}))$ *l1 r1*
and $((\le_{L2}) \equiv_{pre} (\le_{R2}))$ *l2 r2*
shows $((\le_{L}) \equiv_{pre} (\le_{R}))$ *l r*
⟨*proof*⟩

theorem *partial-equivalence-rel-equivalenceI*:
assumes $((\le_{L1}) \equiv_{PER} (\le_{R1}))$ *l1 r1*
and $((\le_{L2}) \equiv_{PER} (\le_{R2}))$ *l2 r2*
shows $((\le_{L}) \equiv_{PER} (\le_{R}))$ *l r*
⟨*proof*⟩

Simplification of Left and Right Relations See ⟦*reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$; *partial-equivalence-rel* $(\le_{L2})$⟧ ⟹ *flip.tpdfr.R* = *flip.tfr.tdfr.R.*

Simplification of Galois relator See ⟦$((\le_{L1}) \Rrightarrow_{m} (\le_{R1}))$ *l1*; *tdfrs.t1.galois-prop l1 r1*; *reflexive-on* (*in-dom* $(\le_{L1})$) $(\le_{L1})$; $((\le_{R2}) \Rrightarrow_{m} (\le_{L2}))$ *r2*; *transitive* $(\le_{L2})$⟧ ⟹ *flip.tpdfr.right-Galois* = (*flip.tdfrs.t1.right-Galois* ⇛ *flip.tdfrs.t2.right-Galois*)$\restriction_{\textit{in-dom flip.tpdfr.R}}\upharpoonleft_{\textit{in-codom flip.tpdfr.L}}$
⟦$((\le_{L1}) \Rrightarrow_{m} (\le_{R1}))$ *l1*; $((\le_{R1}) \Rrightarrow_{m} (\le_{L1}))$ *r1*; *tdfrs.t1.half-galois-prop-right*; *reflexive-on* (*in-field* $(\le_{L1})$) $(\le_{L1})$; *reflexive-on* (*in-field* $(\le_{R1})$) $(\le_{R1})$; *td-*

frs.t2.half-galois-prop-left; *partial-equivalence-rel* $(\le_{L2})$; *partial-equivalence-rel* $(\le_{R2})]\!] \Longrightarrow$ (*flip.tdfrs.t1.right-Galois* $\Rrightarrow$ *flip.tdfrs.t2.right-Galois*)$\restriction_{in\text{-}dom\ flip.tpdfr.R}\upharpoonright_{in\text{-}codom\ flip.tpdfr.L}$ = (*flip.tdfrs.t1.right-Galois* $\Rrightarrow$ *flip.tdfrs.t2.right-Galois*)
tdfrs.t1.half-galois-prop-left $\Longrightarrow$ (*flip.tdfrs.t1.right-Galois* $\Rrightarrow$ *?S*$\restriction_{in\text{-}dom\ (\le_{L2})}\upharpoonright_{in\text{-}codom\ (\le_{R2})}$)$\restriction_{in\text{-}d}$
= (*flip.tdfrs.t1.right-Galois* $\Rrightarrow$ *?S*)$\restriction_{in\text{-}dom\ flip.tpdfr.R}\upharpoonright_{in\text{-}codom\ flip.tpdfr.L}$.

end

Dependent Function Relator While a general transport of functions is only possible for the monotone function relator (see above), the locales *transport-Dep-Fun-Rel* and *transport-Fun-Rel* contain special cases to transport functions that are proven to be monotone using the standard function space.

Moreover, in the special case of equivalences on partial equivalence relations, the standard function space is monotone - see $[\![$*galois.galois-equivalence ?L1.0 ?R1.0 ?l1.0 ?r1.0*; *preorder-on* (*in-field ?L1.0*) *?L1.0*; ([*x1 x2* :: *?L1.0*$^{-1}$] $\Rrightarrow_m$ [*x3 x4* :: *?L1.0*] $\Rrightarrow$ *?L1.0 x1 x3* $\longrightarrow$ $(\le)$) *?L2.0*; $\bigwedge$*x1 x2*. *?L1.0 x1 x2* $\Longrightarrow$ *partial-equivalence-rel* (*?L2.0 x1 x2*)$]\!] \Longrightarrow$ *transport-Mono-Dep-Fun-Rel.L ?L1.0 ?L2.0* = *transport-Dep-Fun-Rel.L ?L1.0 ?L2.0* As such, we can derive general transport theorems from the monotone cases above.

context *transport-Dep-Fun-Rel*
begin

interpretation *tpdfr* : *transport-Mono-Dep-Fun-Rel L1 R1 l1 r1 L2 R2 l2 r2* ⟨*proof*⟩
interpretation *flip* : *transport-Mono-Dep-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2* ⟨*proof*⟩

theorem *partial-equivalence-rel-equivalenceI*:
  assumes $((\le_{L1}) \equiv_{PER} (\le_{R1}))$ *l1 r1*
  and $\bigwedge x\, x'.\ x\ {}_{L1}\lessapprox x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \equiv_{PER} (\le_{R2\ (l1\ x)\ x'}))\ (l2_{x'\ x})\ (r2_{x\ x'})$
  and ([*x1 x2* :: $(\ge_{L1})$] $\Rrightarrow_m$ [*x3 x4* :: $(\le_{L1})$ | *x1* $\le_{L1}$ *x3*] $\Rrightarrow$ $(\le)$) *L2*
  and ([*x1' x2'* :: $(\ge_{R1})$] $\Rrightarrow_m$ [*x3' x4'* :: $(\le_{R1})$ | *x1'* $\le_{R1}$ *x3'*] $\Rrightarrow$ $(\le)$) *R2*
  and ([*x1' x2'* :: $(\le_{R1})$] $\Rrightarrow_m$ [*x1 x2* :: $(\le_{L1})$ | *x2* ${}_{L1}\lessapprox$ *x1'*] $\Rrightarrow$
    [*in-field* $(\le_{L2\ x1\ (r1\ x2')})$] $\Rrightarrow$ $(\le_{R2\ (l1\ x1)\ x2'})$) *l2*
  and ([*x1 x2* :: $(\le_{L1})$] $\Rrightarrow_m$ [*x1' x2'* :: $(\le_{R1})$ | *x2* ${}_{L1}\lessapprox$ *x1'*] $\Rrightarrow$
    [*in-field* $(\le_{R2\ (l1\ x1)\ x2'})$] $\Rrightarrow$ $(\le_{L2\ x1\ (r1\ x2')})$) *r2*
  shows $((\le_L) \equiv_{PER} (\le_R))$ *l r*
⟨*proof*⟩

end

Function Relator **context** *transport-Fun-Rel*
begin

interpretation *tpfr* : *transport-Mono-Fun-Rel L1 R1 l1 r1 L2 R2 l2 r2* ⟨*proof*⟩
interpretation *flip-tpfr* : *transport-Mono-Fun-Rel R1 L1 r1 l1 R2 L2 r2 l2* ⟨*proof*⟩

theorem *partial-equivalence-rel-equivalenceI*:
assumes $((\le_{L1}) \equiv_{PER} (\le_{R1}))$ *l1 r1*
and $((\le_{L2}) \equiv_{PER} (\le_{R2}))$ *l2 r2*
shows $((\le_{L}) \equiv_{PER} (\le_{R}))$ *l r*
⟨*proof*⟩

end

end

2.9 Transport using Identity

theory *Transport-Identity*
imports
Transport-Bijections
begin

Summary Setup for Transport using the identity transport function.

locale *transport-id* =
fixes $L :: 'a \Rightarrow 'a \Rightarrow bool$
begin

sublocale *tbij?* : *transport-bijection L L id id*
⟨*proof*⟩

interpretation *transport L L id id* ⟨*proof*⟩

lemma *left-Galois-eq-left*: $({}_{L}\lessapprox) = (\le_{L})$
⟨*proof*⟩

end

locale *transport-reflexive-on-in-field-id* =
fixes $L :: 'a \Rightarrow 'a \Rightarrow bool$
assumes *reflexive-on-in-field*: *reflexive-on* (*in-field L*) *L*
begin

sublocale *trefl-bij?* : *transport-reflexive-on-in-field-bijection L L id id*
⟨*proof*⟩

end

locale *transport-preorder-on-in-field-id* =
fixes $L :: 'a \Rightarrow 'a \Rightarrow bool$
assumes *preorder-on-in-field*: *preorder-on* (*in-field L*) *L*
begin

sublocale *tpre-bij?* : *transport-preorder-on-in-field-bijection L L id id*
⟨*proof*⟩

end

locale *transport-partial-equivalence-rel-id* =
fixes *L* :: *'a* ⇒ *'a* ⇒ *bool*
assumes *partial-equivalence-rel*: *partial-equivalence-rel L*
begin

sublocale *tper-bij?* : *transport-partial-equivalence-rel-bijection L L id id*
⟨*proof*⟩

end

interpretation *transport-eq-restrict-id* :
transport-eq-restrict-bijection P P id id **for** *P* :: *'a* ⇒ *bool*
⟨*proof*⟩

interpretation *transport-eq-id* : *transport-eq-bijection id id*
⟨*proof*⟩

end

theory *Transport*
imports
Transport-Bijections
Transport-Compositions
Transport-Functions
Transport-Identity
begin

Summary We formalise the theory for the Transport framework. The Transport framework allows us to transport terms along (partial) Galois connections (*galois.galois-connection*) and equivalences (*galois.galois-equivalence*). For details, refer to [2].

end

2.10 Transport for Natural Functors

2.10.1 Basic Setup

theory *Transport-Natural-Functors-Base*
imports
HOL.BNF-Def
HOL-Alignment-Functions

Transport-Base
begin

Summary Basic setup for closure proofs and simple lemmas.

In the following, we willingly use granular apply-style proofs since, in practice, these theorems have to be automatically generated whenever we declare a new natural functor.

Note that "HOL-Library" provides a command *bnf-axiomatization* which allows one to axiomatically declare a bounded natural functor. However, we only need a subset of these axioms - the boundedness of the functor is irrelevant for our purposes. For this reason - and the sake of completeness - we state all the required axioms explicitly below.

lemma *Grp-UNIV-eq-eq-comp*: *BNF-Def.Grp UNIV f* = (=) ∘ *f*
⟨*proof*⟩

lemma *eq-comp-rel-comp-eq-comp*: (=) ∘ *f* ∘∘ *R* = *R* ∘ *f*
⟨*proof*⟩

lemma *Domain-Collect-case-prod-eq-Collect-in-dom*:
Domain {(*x*, *y*). *R x y*} = {*x*. *in-dom R x*}
⟨*proof*⟩

lemma *ball-in-dom-iff-ball-ex*:
(∀ *x* ∈ *S*. *in-dom R x*) ⟷ (∀ *x* ∈ *S*. ∃ *y*. *R x y*)
⟨*proof*⟩

lemma *pair-mem-Collect-case-prod-iff*: (*x*, *y*) ∈ {(*x*, *y*). *R x y*} ⟷ *R x y*
⟨*proof*⟩

Natural Functor Axiomatisation **typedecl** ('*d*, '*a*, '*b*, '*c*) *F*

consts *Fmap* :: ('*a1* ⇒ '*a2*) ⇒ ('*b1* ⇒ '*b2*) ⇒ ('*c1* ⇒ '*c2*) ⇒
('*d*, '*a1*, '*b1*, '*c1*) *F* ⇒ ('*d*, '*a2*, '*b2*, '*c2*) *F*
Fset1 :: ('*d*, '*a*, '*b*, '*c*) *F* ⇒ '*a set*
Fset2 :: ('*d*, '*a*, '*b*, '*c*) *F* ⇒ '*b set*
Fset3 :: ('*d*, '*a*, '*b*, '*c*) *F* ⇒ '*c set*

axiomatization
where *Fmap-id*: *Fmap id id id* = *id*
and *Fmap-comp*: ⋀*f1 f2 f3 g1 g2 g3*.
Fmap (*g1* ∘ *f1*) (*g2* ∘ *f2*) (*g3* ∘ *f3*) = *Fmap g1 g2 g3* ∘ *Fmap f1 f2 f3*
and *Fmap-cong*: ⋀*f1 f2 f3 g1 g2 g3 x*.
(⋀*x1*. *x1* ∈ *Fset1 x* ⟹ *f1 x1* = *g1 x1*) ⟹
(⋀*x2*. *x2* ∈ *Fset2 x* ⟹ *f2 x2* = *g2 x2*) ⟹
(⋀*x3*. *x3* ∈ *Fset3 x* ⟹ *f3 x3* = *g3 x3*) ⟹
Fmap f1 f2 f3 x = *Fmap g1 g2 g3 x*
and *Fset1-natural*: ⋀*f1 f2 f3*. *Fset1* ∘ *Fmap f1 f2 f3* = *image f1* ∘ *Fset1*

and *Fset2-natural*: $\bigwedge f1\ f2\ f3.\ Fset2 \circ Fmap\ f1\ f2\ f3 = image\ f2 \circ Fset2$
and *Fset3-natural*: $\bigwedge f1\ f2\ f3.\ Fset3 \circ Fmap\ f1\ f2\ f3 = image\ f3 \circ Fset3$

lemma *Fmap-id-eq-self*: *Fmap id id id x = x*
⟨*proof*⟩

lemma *Fmap-comp-eq-Fmap-Fmap*:
Fmap (*g1* ∘ *f1*) (*g2* ∘ *f2*) (*g3* ∘ *f3*) *x* = *Fmap g1 g2 g3* (*Fmap f1 f2 f3 x*)
⟨*proof*⟩

lemma *Fset1-Fmap-eq-image-Fset1*: *Fset1* (*Fmap f1 f2 f3 x*) = *f1* ‘ *Fset1 x*
⟨*proof*⟩

lemma *Fset2-Fmap-eq-image-Fset2*: *Fset2* (*Fmap f1 f2 f3 x*) = *f2* ‘ *Fset2 x*
⟨*proof*⟩

lemma *Fset3-Fmap-eq-image-Fset3*: *Fset3* (*Fmap f1 f2 f3 x*) = *f3* ‘ *Fset3 x*
⟨*proof*⟩

lemmas *Fset-Fmap-eqs* = *Fset1-Fmap-eq-image-Fset1 Fset2-Fmap-eq-image-Fset2 Fset3-Fmap-eq-image-Fset3*

Relator **definition** *Frel* :: (*'a1* ⇒ *'a2* ⇒ *bool*) ⇒ (*'b1* ⇒ *'b2* ⇒ *bool*) ⇒ (*'c1* ⇒ *'c2* ⇒ *bool*) ⇒
(*'d*, *'a1*, *'b1*, *'c1*) *F* ⇒ (*'d*, *'a2*, *'b2*, *'c2*) *F* ⇒ *bool*
where *Frel R1 R2 R3 x y* ≡ (∃ *z*.
z ∈ {*x*. *Fset1 x* ⊆ {(*x*, *y*). *R1 x y*} ∧ *Fset2 x* ⊆ {(*x*, *y*). *R2 x y*}
∧ *Fset3 x* ⊆ {(*x*, *y*). *R3 x y*}}
∧ *Fmap fst fst fst z* = *x*
∧ *Fmap snd snd snd z* = *y*)

lemma *FrelI*:
assumes *Fset1 z* ⊆ {(*x*, *y*). *R1 x y*}
and *Fset2 z* ⊆ {(*x*, *y*). *R2 x y*}
and *Fset3 z* ⊆ {(*x*, *y*). *R3 x y*}
and *Fmap fst fst fst z* = *x*
and *Fmap snd snd snd z* = *y*
shows *Frel R1 R2 R3 x y*
⟨*proof*⟩

lemma *FrelE*:
assumes *Frel R1 R2 R3 x y*
obtains *z* **where** *Fset1 z* ⊆ {(*x*, *y*). *R1 x y*} *Fset2 z* ⊆ {(*x*, *y*). *R2 x y*}
Fset3 z ⊆ {(*x*, *y*). *R3 x y*} *Fmap fst fst fst z* = *x Fmap snd snd snd z* = *y*
⟨*proof*⟩

lemma *Grp-UNIV-Fmap-eq-Frel-Grp*: *BNF-Def.Grp UNIV* (*Fmap f1 f2 f3*) =
Frel (*BNF-Def.Grp UNIV f1*) (*BNF-Def.Grp UNIV f2*) (*BNF-Def.Grp UNIV f3*)

⟨*proof*⟩

lemma *Frel-Grp-UNIV-Fmap*:
Frel (*BNF-Def.Grp UNIV f1*) (*BNF-Def.Grp UNIV f2*) (*BNF-Def.Grp UNIV f3*)
x (*Fmap f1 f2 f3 x*)
⟨*proof*⟩

lemma *Frel-Grp-UNIV-iff-eq-Fmap*:
Frel (*BNF-Def.Grp UNIV f1*) (*BNF-Def.Grp UNIV f2*) (*BNF-Def.Grp UNIV f3*) *x y* ⟷
(*y* = *Fmap f1 f2 f3 x*)
⟨*proof*⟩

lemma *Frel-eq*: *Frel* (=) (=) (=) = (=)
⟨*proof*⟩

corollary *Frel-eq-self*: *Frel* (=) (=) (=) *x x*
⟨*proof*⟩

lemma *Frel-mono-strong*:
assumes *Frel R1 R2 R3 x y*
and $\bigwedge$*x1 y1*. *x1* ∈ *Fset1 x* ⟹ *y1* ∈ *Fset1 y* ⟹ *R1 x1 y1* ⟹ *S1 x1 y1*
and $\bigwedge$*x2 y2*. *x2* ∈ *Fset2 x* ⟹ *y2* ∈ *Fset2 y* ⟹ *R2 x2 y2* ⟹ *S2 x2 y2*
and $\bigwedge$*x3 y3*. *x3* ∈ *Fset3 x* ⟹ *y3* ∈ *Fset3 y* ⟹ *R3 x3 y3* ⟹ *S3 x3 y3*
shows *Frel S1 S2 S3 x y*
⟨*proof*⟩

corollary *Frel-mono*:
assumes *R1* ≤ *S1 R2* ≤ *S2 R3* ≤ *S3*
shows *Frel R1 R2 R3* ≤ *Frel S1 S2 S3*
⟨*proof*⟩

lemma *Frel-refl-strong*:
assumes $\bigwedge$*x1*. *x1* ∈ *Fset1 x* ⟹ *R1 x1 x1*
and $\bigwedge$*x2*. *x2* ∈ *Fset2 x* ⟹ *R2 x2 x2*
and $\bigwedge$*x3*. *x3* ∈ *Fset3 x* ⟹ *R3 x3 x3*
shows *Frel R1 R2 R3 x x*
⟨*proof*⟩

lemma *Frel-cong*:
assumes $\bigwedge$*x1 y1*. *x1* ∈ *Fset1 x* ⟹ *y1* ∈ *Fset1 y* ⟹ *R1 x1 y1* ⟷ *R1' x1 y1*
and $\bigwedge$*x2 y2*. *x2* ∈ *Fset2 x* ⟹ *y2* ∈ *Fset2 y* ⟹ *R2 x2 y2* ⟷ *R2' x2 y2*
and $\bigwedge$*x3 y3*. *x3* ∈ *Fset3 x* ⟹ *y3* ∈ *Fset3 y* ⟹ *R3 x3 y3* ⟷ *R3' x3 y3*
shows *Frel R1 R2 R3 x y* = *Frel R1' R2' R3' x y*
⟨*proof*⟩

lemma *Frel-rel-inv-eq-rel-inv-Frel*: *Frel* $R1^{-1}$ $R2^{-1}$ $R3^{-1}$ = (*Frel R1 R2 R3*)$^{-1}$
⟨*proof*⟩

Given the former axioms, the following axiom - subdistributivity of the relator - is equivalent to the (F, Fmap) functor preserving weak pullbacks.

axiomatization
where *Frel-comp-le-Frel-rel-comp*: $\bigwedge R1\ R2\ R3\ S1\ S2\ S3.$
$Frel\ R1\ R2\ R3 \circ\circ Frel\ S1\ S2\ S3 \leq Frel\ (R1 \circ\circ S1)\ (R2 \circ\circ S2)\ (R3 \circ\circ S3)$

lemma *fst-sndOp-eq-snd-fstOp*: $fst \circ BNF\text{-}Def.sndOp\ P\ Q = snd \circ BNF\text{-}Def.fstOp\ P\ Q$
⟨*proof*⟩

lemma *Frel-rel-comp-le-Frel-comp*:
$Frel\ (R1 \circ\circ S1)\ (R2 \circ\circ S2)\ (R3 \circ\circ S3) \leq (Frel\ R1\ R2\ R3 \circ\circ Frel\ S1\ S2\ S3)$
⟨*proof*⟩

corollary *Frel-comp-eq-Frel-rel-comp*:
$Frel\ R1\ R2\ R3 \circ\circ Frel\ S1\ S2\ S3 = Frel\ (R1 \circ\circ S1)\ (R2 \circ\circ S2)\ (R3 \circ\circ S3)$
⟨*proof*⟩

lemma *Frel-Fmap-eq1*: $Frel\ R1\ R2\ R3\ (Fmap\ f1\ f2\ f3\ x)\ y =$
$Frel\ (\lambda x.\ R1\ (f1\ x))\ (\lambda x.\ R2\ (f2\ x))\ (\lambda x.\ R3\ (f3\ x))\ x\ y$
⟨*proof*⟩

lemma *Frel-Fmap-eq2*: $Frel\ R1\ R2\ R3\ x\ (Fmap\ g1\ g2\ g3\ y) =$
$Frel\ (\lambda x\ y.\ R1\ x\ (g1\ y))\ (\lambda x\ y.\ R2\ x\ (g2\ y))\ (\lambda x\ y.\ R3\ x\ (g3\ y))\ x\ y$
⟨*proof*⟩

lemmas *Frel-Fmap-eqs = Frel-Fmap-eq1 Frel-Fmap-eq2*

Predicator

definition $Fpred :: ('a \Rightarrow bool) \Rightarrow ('b \Rightarrow bool) \Rightarrow ('c \Rightarrow bool) \Rightarrow ('d, 'a, 'b, 'c)\ F \Rightarrow bool$
where $Fpred\ P1\ P2\ P3\ x \equiv Frel\ ((=)\restriction_{P1})\ ((=)\restriction_{P2})\ ((=)\restriction_{P3})\ x\ x$

lemma *Fpred-mono-strong*:
assumes $Fpred\ P1\ P2\ P3\ x$
and $\bigwedge x1.\ x1 \in Fset1\ x \Longrightarrow P1\ x1 \Longrightarrow Q1\ x1$
and $\bigwedge x2.\ x2 \in Fset2\ x \Longrightarrow P2\ x2 \Longrightarrow Q2\ x2$
and $\bigwedge x3.\ x3 \in Fset3\ x \Longrightarrow P3\ x3 \Longrightarrow Q3\ x3$
shows $Fpred\ Q1\ Q2\ Q3\ x$
⟨*proof*⟩

lemma *Fpred-top*: $Fpred\ \top\ \top\ \top\ x$
⟨*proof*⟩

lemma *FpredI*:
assumes $\bigwedge x1.\ x1 \in Fset1\ x \Longrightarrow P1\ x1$
and $\bigwedge x2.\ x2 \in Fset2\ x \Longrightarrow P2\ x2$
and $\bigwedge x3.\ x3 \in Fset3\ x \Longrightarrow P3\ x3$
shows $Fpred\ P1\ P2\ P3\ x$
⟨*proof*⟩

lemma *FpredE*:
assumes *Fpred P1 P2 P3 x*
obtains $\bigwedge x1.\ x1 \in Fset1\ x \Longrightarrow P1\ x1$
$\bigwedge x2.\ x2 \in Fset2\ x \Longrightarrow P2\ x2$
$\bigwedge x3.\ x3 \in Fset3\ x \Longrightarrow P3\ x3$
⟨*proof*⟩

lemma *Fpred-eq-ball*: *Fpred P1 P2 P3* =
($\lambda x.$ *Ball* (*Fset1 x*) *P1* ∧ *Ball* (*Fset2 x*) *P2* ∧ *Ball* (*Fset3 x*) *P3*)
⟨*proof*⟩

lemma *Fpred-Fmap-eq*:
Fpred P1 P2 P3 (*Fmap f1 f2 f3 x*) = *Fpred* (*P1* ∘ *f1*) (*P2* ∘ *f2*) (*P3* ∘ *f3*) *x*
⟨*proof*⟩

lemma *Fpred-in-dom-if-in-dom-Frel*:
assumes *in-dom* (*Frel R1 R2 R3*) *x*
shows *Fpred* (*in-dom R1*) (*in-dom R2*) (*in-dom R3*) *x*
⟨*proof*⟩

lemma *in-dom-Frel-if-Fpred-in-dom*:
assumes *Fpred* (*in-dom R1*) (*in-dom R2*) (*in-dom R3*) *x*
shows *in-dom* (*Frel R1 R2 R3*) *x*
⟨*proof*⟩

lemma *in-dom-Frel-eq-Fpred-in-dom*:
in-dom (*Frel R1 R2 R3*) = *Fpred* (*in-dom R1*) (*in-dom R2*) (*in-dom R3*)
⟨*proof*⟩

lemma *in-codom-Frel-eq-Fpred-in-codom*:
in-codom (*Frel R1 R2 R3*) = *Fpred* (*in-codom R1*) (*in-codom R2*) (*in-codom R3*)
⟨*proof*⟩

lemma *in-field-Frel-eq-Fpred-in-in-field*:
in-field (*Frel R1 R2 R3*) =
Fpred (*in-dom R1*) (*in-dom R2*) (*in-dom R3*) ⊔
Fpred (*in-codom R1*) (*in-codom R2*) (*in-codom R3*)
⟨*proof*⟩

lemma *Frel-restrict-left-Fpred-eq-Frel-restrict-left*:
fixes *R1* :: *'a1* ⇒ *'a2* ⇒ *bool*
and *R2* :: *'b1* ⇒ *'b2* ⇒ *bool*
and *R3* :: *'c1* ⇒ *'c2* ⇒ *bool*
and *P1* :: *'a1* ⇒ *bool*
and *P2* :: *'b1* ⇒ *bool*
and *P3* :: *'c1* ⇒ *bool*
shows (*Frel R1 R2 R3* :: (*'d*, *'a1*, *'b1*, *'c1*) *F* ⇒ -)$\restriction_{Fpred\ P1\ P2\ P3\ ::\ ('d,\ 'a1,\ 'b1,\ 'c1)\ F\ \Rightarrow\ -}$
=

Frel $(R1\restriction_{P1})$ $(R2\restriction_{P2})$ $(R3\restriction_{P3})$
⟨*proof*⟩

lemma *Frel-restrict-right-Fpred-eq-Frel-restrict-right*:
fixes *R1* :: *'a1* ⇒ *'a2* ⇒ *bool*
and *R2* :: *'b1* ⇒ *'b2* ⇒ *bool*
and *R3* :: *'c1* ⇒ *'c2* ⇒ *bool*
and *P1* :: *'a2* ⇒ *bool*
and *P2* :: *'b2* ⇒ *bool*
and *P3* :: *'c2* ⇒ *bool*
shows (*Frel R1 R2 R3* :: - ⇒ (*'d*, *'a2*, *'b2*, *'c2*) *F* ⇒ -)$\upharpoonleft_{Fpred\ P1\ P2\ P3\ ::\ ('d,\ 'a2,\ 'b2,\ 'c2)\ F\ \Rightarrow\ -}$ =
Frel $(R1\upharpoonleft_{P1})$ $(R2\upharpoonleft_{P2})$ $(R3\upharpoonleft_{P3})$
⟨*proof*⟩

locale *transport-natural-functor* =
t1 : *transport L1 R1 l1 r1* + *t2* : *transport L2 R2 l2 r2* +
t3 : *transport L3 R3 l3 r3*
for *L1* :: *'a1* ⇒ *'a1* ⇒ *bool*
and *R1* :: *'b1* ⇒ *'b1* ⇒ *bool*
and *l1* :: *'a1* ⇒ *'b1*
and *r1* :: *'b1* ⇒ *'a1*
and *L2* :: *'a2* ⇒ *'a2* ⇒ *bool*
and *R2* :: *'b2* ⇒ *'b2* ⇒ *bool*
and *l2* :: *'a2* ⇒ *'b2*
and *r2* :: *'b2* ⇒ *'a2*
and *L3* :: *'a3* ⇒ *'a3* ⇒ *bool*
and *R3* :: *'b3* ⇒ *'b3* ⇒ *bool*
and *l3* :: *'a3* ⇒ *'b3*
and *r3* :: *'b3* ⇒ *'a3*
begin

notation *L1* (**infix** $\le_{L1}$ *50*)
notation *R1* (**infix** $\le_{R1}$ *50*)
notation *L2* (**infix** $\le_{L2}$ *50*)
notation *R2* (**infix** $\le_{R2}$ *50*)
notation *L3* (**infix** $\le_{L3}$ *50*)
notation *R3* (**infix** $\le_{R3}$ *50*)

notation *t1.ge-left* (**infix** $\ge_{L1}$ *50*)
notation *t1.ge-right* (**infix** $\ge_{R1}$ *50*)
notation *t2.ge-left* (**infix** $\ge_{L2}$ *50*)
notation *t2.ge-right* (**infix** $\ge_{R2}$ *50*)
notation *t3.ge-left* (**infix** $\ge_{L3}$ *50*)
notation *t3.ge-right* (**infix** $\ge_{R3}$ *50*)

notation *t1.left-Galois* (**infix** ${}_{L1}\lessapprox$ *50*)
notation *t1.right-Galois* (**infix** ${}_{R1}\lessapprox$ *50*)
notation *t2.left-Galois* (**infix** ${}_{L2}\lessapprox$ *50*)

notation *t2.right-Galois* (**infix** ${}_{R2}\lessapprox$ *50*)
notation *t3.left-Galois* (**infix** ${}_{L3}\lessapprox$ *50*)
notation *t3.right-Galois* (**infix** ${}_{R3}\lessapprox$ *50*)

notation *t1.ge-Galois-left* (**infix** $\gtrapprox_{L1}$ *50*)
notation *t1.ge-Galois-right* (**infix** $\gtrapprox_{R1}$ *50*)
notation *t2.ge-Galois-left* (**infix** $\gtrapprox_{L2}$ *50*)
notation *t2.ge-Galois-right* (**infix** $\gtrapprox_{R2}$ *50*)
notation *t3.ge-Galois-left* (**infix** $\gtrapprox_{L3}$ *50*)
notation *t3.ge-Galois-right* (**infix** $\gtrapprox_{R3}$ *50*)

notation *t1.right-ge-Galois* (**infix** ${}_{R1}\gtrapprox$ *50*)
notation *t1.Galois-right* (**infix** $\lessapprox_{R1}$ *50*)
notation *t2.right-ge-Galois* (**infix** ${}_{R2}\gtrapprox$ *50*)
notation *t2.Galois-right* (**infix** $\lessapprox_{R2}$ *50*)
notation *t3.right-ge-Galois* (**infix** ${}_{R3}\gtrapprox$ *50*)
notation *t3.Galois-right* (**infix** $\lessapprox_{R3}$ *50*)

notation *t1.left-ge-Galois* (**infix** ${}_{L1}\gtrapprox$ *50*)
notation *t1.Galois-left* (**infix** $\lessapprox_{L1}$ *50*)
notation *t2.left-ge-Galois* (**infix** ${}_{L2}\gtrapprox$ *50*)
notation *t2.Galois-left* (**infix** $\lessapprox_{L2}$ *50*)
notation *t3.left-ge-Galois* (**infix** ${}_{L3}\gtrapprox$ *50*)
notation *t3.Galois-left* (**infix** $\lessapprox_{L3}$ *50*)

notation *t1.unit* (η_1)
notation *t1.counit* (ε_1)
notation *t2.unit* (η_2)
notation *t2.counit* (ε_2)
notation *t3.unit* (η_3)
notation *t3.counit* (ε_3)

definition $L \equiv$ *Frel* ($\le_{L1}$) ($\le_{L2}$) ($\le_{L3}$)

lemma *left-rel-eq-Frel*: $L =$ *Frel* ($\le_{L1}$) ($\le_{L2}$) ($\le_{L3}$)
⟨*proof*⟩

definition $l \equiv$ *Fmap l1 l2 l3*

lemma *left-eq-Fmap*: $l =$ *Fmap l1 l2 l3*
⟨*proof*⟩

context
begin

interpretation *flip* :
transport-natural-functor R1 L1 r1 l1 R2 L2 r2 l2 R3 L3 r3 l3 ⟨*proof*⟩

abbreviation $R \equiv$ *flip.L*

abbreviation $r \equiv$ *flip.l*

lemma *right-rel-eq-Frel*: $R = Frel\ (\leq_{R1})\ (\leq_{R2})\ (\leq_{R3})$
⟨*proof*⟩

lemma *right-eq-Fmap*: $r = Fmap\ r1\ r2\ r3$
⟨*proof*⟩

lemmas *transport-defs* = *left-rel-eq-Frel left-eq-Fmap*
right-rel-eq-Frel right-eq-Fmap

end

sublocale *transport L R l r* ⟨*proof*⟩

notation L (**infix** $\leq_L$ *50*)
notation R (**infix** $\leq_R$ *50*)

lemma *unit-eq-Fmap*: $\eta = Fmap\ \eta_1\ \eta_2\ \eta_3$
⟨*proof*⟩

interpretation *flip-inv* : *transport-natural-functor* $(\geq_{R1})\ (\geq_{L1})\ r1\ l1$
$(\geq_{R2})\ (\geq_{L2})\ r2\ l2\ (\geq_{R3})\ (\geq_{L3})\ r3\ l3$
rewrites *flip-inv.unit* $\equiv \varepsilon$ **and** *flip-inv.t1.unit* $\equiv \varepsilon_1$
and *flip-inv.t2.unit* $\equiv \varepsilon_2$ **and** *flip-inv.t3.unit* $\equiv \varepsilon_3$
⟨*proof*⟩

lemma *counit-eq-Fmap*: $\varepsilon = Fmap\ \varepsilon_1\ \varepsilon_2\ \varepsilon_3$
⟨*proof*⟩

lemma *flip-inv-right-eq-ge-left*: *flip-inv.R* $= (\geq_L)$
⟨*proof*⟩

interpretation *flip* :
transport-natural-functor R1 L1 r1 l1 R2 L2 r2 l2 R3 L3 r3 l3 ⟨*proof*⟩

lemma *flip-inv-left-eq-ge-right*: *flip-inv.L* $\equiv (\geq_R)$
⟨*proof*⟩

lemma *mono-wrt-rel-leftI*:
assumes $((\leq_{L1}) \Rrightarrow_m (\leq_{R1}))\ l1$
and $((\leq_{L2}) \Rrightarrow_m (\leq_{R2}))\ l2$
and $((\leq_{L3}) \Rrightarrow_m (\leq_{R3}))\ l3$
shows $((\leq_L) \Rrightarrow_m (\leq_R))\ l$
⟨*proof*⟩

end

end

2.10.2 Galois Concepts

theory *Transport-Natural-Functors-Galois*
imports
Transport-Natural-Functors-Base
begin

context *transport-natural-functor*
begin

lemma *half-galois-prop-leftI*:
assumes $((\le_{L1})\ {}_h\trianglelefteq\ (\le_{R1}))$ *l1 r1*
and $((\le_{L2})\ {}_h\trianglelefteq\ (\le_{R2}))$ *l2 r2*
and $((\le_{L3})\ {}_h\trianglelefteq\ (\le_{R3}))$ *l3 r3*
shows $((\le_{L})\ {}_h\trianglelefteq\ (\le_{R}))$ *l r*
⟨*proof*⟩

interpretation *flip-inv* : *transport-natural-functor* $(\ge_{R1})$ $(\ge_{L1})$ *r1 l1* $(\ge_{R2})$ $(\ge_{L2})$ *r2 l2* $(\ge_{R3})$ $(\ge_{L3})$ *r3 l3*
rewrites *flip-inv.R* $\equiv (\ge_{L})$
and *flip-inv.L* $\equiv (\ge_{R})$
and $\bigwedge R\ S\ f\ g.\ (R^{-1}\ {}_h\trianglelefteq\ S^{-1})\ f\ g \equiv (S \trianglelefteq_h R)\ g\ f$
⟨*proof*⟩

lemma *half-galois-prop-rightI*:
assumes $((\le_{L1}) \trianglelefteq_h (\le_{R1}))$ *l1 r1*
and $((\le_{L2}) \trianglelefteq_h (\le_{R2}))$ *l2 r2*
and $((\le_{L3}) \trianglelefteq_h (\le_{R3}))$ *l3 r3*
shows $((\le_{L}) \trianglelefteq_h (\le_{R}))$ *l r*
⟨*proof*⟩

corollary *galois-propI*:
assumes $((\le_{L1}) \trianglelefteq (\le_{R1}))$ *l1 r1*
and $((\le_{L2}) \trianglelefteq (\le_{R2}))$ *l2 r2*
and $((\le_{L3}) \trianglelefteq (\le_{R3}))$ *l3 r3*
shows $((\le_{L}) \trianglelefteq (\le_{R}))$ *l r*
⟨*proof*⟩

interpretation *flip* :
transport-natural-functor R1 L1 r1 l1 R2 L2 r2 l2 R3 L3 r3 l3 ⟨*proof*⟩

corollary *galois-connectionI*:
assumes $((\le_{L1}) \dashv (\le_{R1}))$ *l1 r1*
and $((\le_{L2}) \dashv (\le_{R2}))$ *l2 r2*
and $((\le_{L3}) \dashv (\le_{R3}))$ *l3 r3*
shows $((\le_{L}) \dashv (\le_{R}))$ *l r*

⟨*proof*⟩

corollary *galois-equivalenceI*:
assumes $((\le_{L1}) \equiv_G (\le_{R1}))$ *l1 r1*
and $((\le_{L2}) \equiv_G (\le_{R2}))$ *l2 r2*
and $((\le_{L3}) \equiv_G (\le_{R3}))$ *l3 r3*
shows $((\le_L) \equiv_G (\le_R))$ *l r*
⟨*proof*⟩

end

end

2.10.3 Galois Relator

theory *Transport-Natural-Functors-Galois-Relator*
imports
Transport-Natural-Functors-Base
begin

context *transport-natural-functor*
begin

lemma *left-Galois-Frel-left-Galois*: $({}_{L}\lessapprox) \le Frel\ ({}_{L1}\lessapprox)\ ({}_{L2}\lessapprox)\ ({}_{L3}\lessapprox)$
⟨*proof*⟩

lemma *Frel-left-Galois-le-left-Galois*:
$Frel\ ({}_{L1}\lessapprox)\ ({}_{L2}\lessapprox)\ ({}_{L3}\lessapprox) \le ({}_{L}\lessapprox)$
⟨*proof*⟩

corollary *left-Galois-eq-Frel-left-Galois*: $({}_{L}\lessapprox) = Frel\ ({}_{L1}\lessapprox)\ ({}_{L2}\lessapprox)\ ({}_{L3}\lessapprox)$
⟨*proof*⟩

end

end

2.10.4 Basic Order Properties

theory *Transport-Natural-Functors-Order-Base*
imports
Transport-Natural-Functors-Base
begin

lemma *reflexive-on-in-field-FrelI*:
assumes *reflexive-on* (*in-field R1*) *R1*
and *reflexive-on* (*in-field R2*) *R2*
and *reflexive-on* (*in-field R3*) *R3*

defines *R* ≡ *Frel R1 R2 R3*
shows *reflexive-on* (*in-field R*) *R*
⟨*proof*⟩

lemma *transitive-FrelI*:
assumes *transitive R1*
and *transitive R2*
and *transitive R3*
shows *transitive* (*Frel R1 R2 R3*)
⟨*proof*⟩

lemma *preorder-on-in-field-FrelI*:
assumes *preorder-on* (*in-field R1*) *R1*
and *preorder-on* (*in-field R2*) *R2*
and *preorder-on* (*in-field R3*) *R3*
defines *R* ≡ *Frel R1 R2 R3*
shows *preorder-on* (*in-field R*) *R*
⟨*proof*⟩

lemma *symmetric-FrelI*:
assumes *symmetric R1*
and *symmetric R2*
and *symmetric R3*
shows *symmetric* (*Frel R1 R2 R3*)
⟨*proof*⟩

lemma *partial-equivalence-rel-FrelI*:
assumes *partial-equivalence-rel R1*
and *partial-equivalence-rel R2*
and *partial-equivalence-rel R3*
shows *partial-equivalence-rel* (*Frel R1 R2 R3*)
⟨*proof*⟩

context *transport-natural-functor*
begin

lemmas *reflexive-on-in-field-leftI* = *reflexive-on-in-field-FrelI*
[*of L1 L2 L3*, *folded transport-defs*]

lemmas *transitive-leftI* = *transitive-FrelI*[*of L1 L2 L3*, *folded transport-defs*]

lemmas *preorder-on-in-field-leftI* = *preorder-on-in-field-FrelI*
[*of L1 L2 L3*, *folded transport-defs*]

lemmas *symmetricI* = *symmetric-FrelI*[*of L1 L2 L3*, *folded transport-defs*]

lemmas *partial-equivalence-rel-leftI* = *partial-equivalence-rel-FrelI*
[*of L1 L2 L3*, *folded transport-defs*]

end

end

2.10.5 Order Equivalence

theory *Transport-Natural-Functors-Order-Equivalence*
imports
Transport-Natural-Functors-Base
begin

lemma *inflationary-on-in-dom-FrelI*:
assumes *inflationary-on* (*in-dom R1*) *R1 f1*
and *inflationary-on* (*in-dom R2*) *R2 f2*
and *inflationary-on* (*in-dom R3*) *R3 f3*
defines *R* ≡ *Frel R1 R2 R3*
shows *inflationary-on* (*in-dom R*) *R* (*Fmap f1 f2 f3*)
⟨*proof*⟩

lemma *inflationary-on-in-codom-FrelI*:
assumes *inflationary-on* (*in-codom R1*) *R1 f1*
and *inflationary-on* (*in-codom R2*) *R2 f2*
and *inflationary-on* (*in-codom R3*) *R3 f3*
defines *R* ≡ *Frel R1 R2 R3*
shows *inflationary-on* (*in-codom R*) *R* (*Fmap f1 f2 f3*)
⟨*proof*⟩

lemma *inflationary-on-in-field-FrelI*:
assumes *inflationary-on* (*in-field R1*) *R1 f1*
and *inflationary-on* (*in-field R2*) *R2 f2*
and *inflationary-on* (*in-field R3*) *R3 f3*
defines *R* ≡ *Frel R1 R2 R3*
shows *inflationary-on* (*in-field R*) *R* (*Fmap f1 f2 f3*)
⟨*proof*⟩

lemma *deflationary-on-in-dom-FrelI*:
assumes *deflationary-on* (*in-dom R1*) *R1 f1*
and *deflationary-on* (*in-dom R2*) *R2 f2*
and *deflationary-on* (*in-dom R3*) *R3 f3*
defines *R* ≡ *Frel R1 R2 R3*
shows *deflationary-on* (*in-dom R*) *R* (*Fmap f1 f2 f3*)
⟨*proof*⟩

lemma *deflationary-on-in-codom-FrelI*:
assumes *deflationary-on* (*in-codom R1*) *R1 f1*
and *deflationary-on* (*in-codom R2*) *R2 f2*
and *deflationary-on* (*in-codom R3*) *R3 f3*
defines *R* ≡ *Frel R1 R2 R3*

shows *deflationary-on* (*in-codom R*) *R* (*Fmap f1 f2 f3*)
⟨*proof*⟩

lemma *deflationary-on-in-field-FrelI*:
assumes *deflationary-on* (*in-field R1*) *R1 f1*
and *deflationary-on* (*in-field R2*) *R2 f2*
and *deflationary-on* (*in-field R3*) *R3 f3*
defines *R* ≡ *Frel R1 R2 R3*
shows *deflationary-on* (*in-field R*) *R* (*Fmap f1 f2 f3*)
⟨*proof*⟩

lemma *rel-equivalence-on-in-field-FrelI*:
assumes *rel-equivalence-on* (*in-field R1*) *R1 f1*
and *rel-equivalence-on* (*in-field R2*) *R2 f2*
and *rel-equivalence-on* (*in-field R3*) *R3 f3*
defines *R* ≡ *Frel R1 R2 R3*
shows *rel-equivalence-on* (*in-field R*) *R* (*Fmap f1 f2 f3*)
⟨*proof*⟩

context *transport-natural-functor*
begin

lemmas *inflationary-on-in-field-unitI* = *inflationary-on-in-field-FrelI*
[*of L1* η_1 *L2* η_2 *L3* η_3, *folded transport-defs unit-eq-Fmap*]

lemmas *deflationary-on-in-field-unitI* = *deflationary-on-in-field-FrelI*
[*of L1* η_1 *L2* η_2 *L3* η_3, *folded transport-defs unit-eq-Fmap*]

lemmas *rel-equivalence-on-in-field-unitI* = *rel-equivalence-on-in-field-FrelI*
[*of L1* η_1 *L2* η_2 *L3* η_3, *folded transport-defs unit-eq-Fmap*]

interpretation *flip* :
transport-natural-functor R1 L1 r1 l1 R2 L2 r2 l2 R3 L3 r3 l3
rewrites *flip.unit* ≡ ε **and** *flip.t1.unit* ≡ ε_1
and *flip.t2.unit* ≡ ε_2 **and** *flip.t3.unit* ≡ ε_3
⟨*proof*⟩

lemma *order-equivalenceI*:
assumes $((\le_{L1}) \equiv_o (\le_{R1}))$ *l1 r1*
and $((\le_{L2}) \equiv_o (\le_{R2}))$ *l2 r2*
and $((\le_{L3}) \equiv_o (\le_{R3}))$ *l3 r3*
shows $((\le_{L}) \equiv_o (\le_{R}))$ *l r*
⟨*proof*⟩

end

end

theory *Transport-Natural-Functors*
imports
Transport-Natural-Functors-Galois
Transport-Natural-Functors-Galois-Relator
Transport-Natural-Functors-Order-Base
Transport-Natural-Functors-Order-Equivalence
begin

Summary Summary of results for a fixed natural functor with 3 parameters. All apply-style proofs are written such that they also apply to functors with other arities. An automatic derivation of these results for all natural functors needs to be implemented in the BNF package. This is future work.

context *transport-natural-functor*
begin

interpretation *flip* :
transport-natural-functor R1 L1 r1 l1 R2 L2 r2 l2 R3 L3 r3 l3 ⟨*proof*⟩

theorem *preorder-equivalenceI*:
assumes $((\le_{L1}) \equiv_{pre} (\le_{R1}))$ *l1 r1*
and $((\le_{L2}) \equiv_{pre} (\le_{R2}))$ *l2 r2*
and $((\le_{L3}) \equiv_{pre} (\le_{R3}))$ *l3 r3*
shows $((\le_{L}) \equiv_{pre} (\le_{R}))$ *l r*
⟨*proof*⟩

theorem *partial-equivalence-rel-equivalenceI*:
assumes $((\le_{L1}) \equiv_{PER} (\le_{R1}))$ *l1 r1*
and $((\le_{L2}) \equiv_{PER} (\le_{R2}))$ *l2 r2*
and $((\le_{L3}) \equiv_{PER} (\le_{R3}))$ *l3 r3*
shows $((\le_{L}) \equiv_{PER} (\le_{R}))$ *l r*
⟨*proof*⟩

For the simplification of the Galois relator see *flip.right-Galois = Frel flip.t1.right-Galois flip.t2.right-Galois flip.t3.right-Galois*.

end

end

2.11 Transport for Dependent Function Relator with Non-Dependent Functions

theory *Transport-Rel-If*
imports
Transport
begin

Summary We introduce a special case of *transport-Dep-Fun-Rel*. The derived theorem is easier to apply and supported by the current prototype.

context
  fixes $P :: 'a \Rightarrow bool$ **and** $R :: 'a \Rightarrow 'a \Rightarrow bool$
begin

lemma *reflexive-on-rel-if-if-reflexive-onI* [*intro*]:
  assumes $B \Longrightarrow$ *reflexive-on* P R
  shows *reflexive-on* P (*rel-if* B R)
  ⟨*proof*⟩

lemma *transitive-on-rel-if-if-transitive-onI* [*intro*]:
  assumes $B \Longrightarrow$ *transitive-on* P R
  shows *transitive-on* P (*rel-if* B R)
  ⟨*proof*⟩

lemma *preorder-on-rel-if-if-preorder-onI* [*intro*]:
  assumes $B \Longrightarrow$ *preorder-on* P R
  shows *preorder-on* P (*rel-if* B R)
  ⟨*proof*⟩

lemma *symmetric-on-rel-if-if-symmetric-onI* [*intro*]:
  assumes $B \Longrightarrow$ *symmetric-on* P R
  shows *symmetric-on* P (*rel-if* B R)
  ⟨*proof*⟩

lemma *partial-equivalence-rel-on-rel-if-if-partial-equivalence-rel-onI* [*intro*]:
  assumes $B \Longrightarrow$ *partial-equivalence-rel-on* P R
  shows *partial-equivalence-rel-on* P (*rel-if* B R)
  ⟨*proof*⟩

lemma *rel-if-dep-mono-wrt-rel-if-iff-if-dep-mono-wrt-relI*:
  assumes $B \Longrightarrow B' \Longrightarrow ([x\ y :: R] \Rrightarrow_m S\ x\ y)\ f$
  and $B \longleftrightarrow B'$
  shows $([x\ y :: (\textit{rel-if}\ B\ R)] \Rrightarrow_m (\textit{rel-if}\ B'\ (S\ x\ y)))\ f$
  ⟨*proof*⟩

end

corollary *reflexive-rel-if-if-reflexiveI* [*intro*]:
  assumes $B \Longrightarrow$ *reflexive* R
  shows *reflexive* (*rel-if* B R)
  ⟨*proof*⟩

corollary *transitive-rel-if-if-transitiveI* [*intro*]:
  assumes $B \Longrightarrow$ *transitive* R
  shows *transitive* (*rel-if* B R)
  ⟨*proof*⟩

corollary *preorder-rel-if-if-preorderI* [*intro*]:
  assumes $B \Longrightarrow$ *preorder* R
  shows *preorder* (*rel-if* B R)
  ⟨*proof*⟩

corollary *symmetric-rel-if-if-symmetricI* [*intro*]:
  assumes $B \Longrightarrow$ *symmetric* R
  shows *symmetric* (*rel-if* B R)
  ⟨*proof*⟩

corollary *partial-equivalence-rel-rel-if-if-partial-equivalence-relI* [*intro*]:
  assumes $B \Longrightarrow$ *partial-equivalence-rel* R
  shows *partial-equivalence-rel* (*rel-if* B R)
  ⟨*proof*⟩

context *galois-prop*
begin

interpretation *rel-if* : *galois-prop rel-if* B $(\leq_L)$ *rel-if* B' $(\leq_R)$ l r ⟨*proof*⟩
interpretation *flip-inv* : *galois-prop* $(\geq_R)$ $(\geq_L)$ r l ⟨*proof*⟩

lemma *rel-if-half-galois-prop-left-if-iff-if-half-galois-prop-leftI*:
  assumes $B \Longrightarrow B' \Longrightarrow ((\leq_L)\ {}_h\trianglelefteq\ (\leq_R))\ l\ r$
  and $B \longleftrightarrow B'$
  shows $((\textit{rel-if}\ B\ (\leq_L))\ {}_h\trianglelefteq\ (\textit{rel-if}\ B'\ (\leq_R)))\ l\ r$
  ⟨*proof*⟩

lemma *rel-if-half-galois-prop-right-if-iff-if-half-galois-prop-rightI*:
  assumes $B \Longrightarrow B' \Longrightarrow ((\leq_L) \trianglelefteq_h (\leq_R))\ l\ r$
  and $B \longleftrightarrow B'$
  shows $((\textit{rel-if}\ B\ (\leq_L)) \trianglelefteq_h (\textit{rel-if}\ B'\ (\leq_R)))\ l\ r$
  ⟨*proof*⟩

lemma *rel-if-galois-prop-if-iff-if-galois-propI*:
  assumes $B \Longrightarrow B' \Longrightarrow ((\leq_L) \trianglelefteq (\leq_R))\ l\ r$
  and $B \longleftrightarrow B'$
  shows $((\textit{rel-if}\ B\ (\leq_L)) \trianglelefteq (\textit{rel-if}\ B'\ (\leq_R)))\ l\ r$
  ⟨*proof*⟩

end

context *galois*
begin

interpretation *rel-if* : *galois rel-if* B $(\leq_L)$ *rel-if* B' $(\leq_R)$ l r ⟨*proof*⟩

lemma *rel-if-galois-connection-if-iff-if-galois-connectionI*:
  assumes $B \Longrightarrow B' \Longrightarrow ((\leq_L) \dashv (\leq_R))\ l\ r$
  and $B \longleftrightarrow B'$

shows $((\mathit{rel\text{-}if}\ B\ (\le_L)) \dashv (\mathit{rel\text{-}if}\ B'\ (\le_R)))\ l\ r$
⟨*proof*⟩

lemma *rel-if-galois-equivalence-if-iff-if-galois-equivalenceI*:
assumes $B \Longrightarrow B' \Longrightarrow ((\le_L) \equiv_G (\le_R))\ l\ r$
and $B \longleftrightarrow B'$
shows $((\mathit{rel\text{-}if}\ B\ (\le_L)) \equiv_G (\mathit{rel\text{-}if}\ B'\ (\le_R)))\ l\ r$
⟨*proof*⟩

end

context *transport*
begin

interpretation *rel-if* : *transport rel-if* $B\ (\le_L)$ *rel-if* $B'\ (\le_R)\ l\ r$ ⟨*proof*⟩

lemma *rel-if-preorder-equivalence-if-iff-if-preorder-equivalenceI*:
assumes $B \Longrightarrow B' \Longrightarrow ((\le_L) \equiv_{pre} (\le_R))\ l\ r$
and $B \longleftrightarrow B'$
shows $((\mathit{rel\text{-}if}\ B\ (\le_L)) \equiv_{pre} (\mathit{rel\text{-}if}\ B'\ (\le_R)))\ l\ r$
⟨*proof*⟩

lemma *rel-if-partial-equivalence-rel-equivalence-if-iff-if-partial-equivalence-rel-equivalenceI*:
assumes $B \Longrightarrow B' \Longrightarrow ((\le_L) \equiv_{PER} (\le_R))\ l\ r$
and $B \longleftrightarrow B'$
shows $((\mathit{rel\text{-}if}\ B\ (\le_L)) \equiv_{PER} (\mathit{rel\text{-}if}\ B'\ (\le_R)))\ l\ r$
⟨*proof*⟩

end

locale *transport-Dep-Fun-Rel-no-dep-fun* =
transport-Dep-Fun-Rel-syntax L1 R1 l1 r1 L2 R2 λ- -. *l2* λ- -. *r2* +
tdfr : *transport-Dep-Fun-Rel L1 R1 l1 r1 L2 R2* λ- -. *l2* λ- -. *r2*
for *L1* :: *'a1* ⇒ *'a1* ⇒ *bool*
and *R1* :: *'a2* ⇒ *'a2* ⇒ *bool*
and *l1* :: *'a1* ⇒ *'a2*
and *r1* :: *'a2* ⇒ *'a1*
and *L2* :: *'a1* ⇒ *'a1* ⇒ *'b1* ⇒ *'b1* ⇒ *bool*
and *R2* :: *'a2* ⇒ *'a2* ⇒ *'b2* ⇒ *'b2* ⇒ *bool*
and *l2* :: *'b1* ⇒ *'b2*
and *r2* :: *'b2* ⇒ *'b1*
begin

notation *t2.unit* (η_2)
notation *t2.counit* (ε_2)

abbreviation $L \equiv \mathit{tdfr}.L$
abbreviation $R \equiv \mathit{tdfr}.R$

abbreviation $l \equiv$ *tdfr.l*
abbreviation $r \equiv$ *tdfr.r*

notation *tdfr.L* (**infix** $\le_L$ *50*)
notation *tdfr.R* (**infix** $\le_R$ *50*)

notation *tdfr.ge-left* (**infix** $\ge_L$ *50*)
notation *tdfr.ge-right* (**infix** $\ge_R$ *50*)

notation *tdfr.unit* (η)
notation *tdfr.counit* (ε)

theorem *partial-equivalence-rel-equivalenceI*:
  assumes *per-equiv1*: $((\le_{L1}) \equiv_{PER} (\le_{R1}))\ l1\ r1$
  and *per-equiv2*: $\bigwedge x\ x'.\ x\ {}_{L1}\lessapprox\ x' \Longrightarrow ((\le_{L2\ x\ (r1\ x')}) \equiv_{PER} (\le_{R2\ (l1\ x)\ x'}))\ l2$
$r2$
  and $([x1\ x2 :: (\ge_{L1})] \Rrightarrow_m [x3\ x4 :: (\le_{L1}) \mid x1 \le_{L1} x3] \Rrightarrow (\le))\ L2$
  and $([x1'\ x2' :: (\ge_{R1})] \Rrightarrow_m [x3'\ x4' :: (\le_{R1}) \mid x1' \le_{R1} x3'] \Rrightarrow (\le))\ R2$
  shows $((\le_L) \equiv_{PER} (\le_R))\ l\ r$
⟨*proof*⟩

end

end

2.12 Transport via Equivalences on PERs (Prototype)

theory *Transport-Prototype*
  imports
    Transport-Rel-If
    ML-Unification.ML-Unification-HOL-Setup
    ML-Unification.Unify-Resolve-Tactics
  keywords *trp-term* :: *thy-goal-defn*
begin

Summary We implement a simple Transport prototype. The prototype is restricted to work with equivalences on partial equivalence relations. It is also not forming the compositions of equivalences so far. The support for dependent function relators is restricted to the form described in ⟦*transport.partial-equivalence-rel-equivalence ?L1.0 ?R1.0 ?l1.0 ?r1.0*; $\bigwedge x\ x'.$ *galois-rel.Galois ?L1.0 ?R1.0 ?r1.0 x x'* $\Longrightarrow$ *transport.partial-equivalence-rel-equivalence* (*?L2.0 x* (*?r1.0 x'*)) (*?R2.0* (*?l1.0 x*) *x'*) *?l2.0 ?r2.0*; ([*x1 x2* :: *?L1.0*$^{-1}$] $\Rrightarrow_m$ [*x3 x4* :: *?L1.0*] $\Rrightarrow$ *?L1.0 x1 x3* $\longrightarrow$ ($\le$)) *?L2.0*; ([*x1' x2'* :: *?R1.0*$^{-1}$] $\Rrightarrow_m$ [*x3' x4'* :: *?R1.0*] $\Rrightarrow$ *?R1.0 x1' x3'* $\longrightarrow$ ($\le$)) *?R2.0*⟧ $\Longrightarrow$ *transport.partial-equivalence-rel-equivale* (*transport-Dep-Fun-Rel.L ?L1.0 ?L2.0*) (*transport-Dep-Fun-Rel.L ?R1.0 ?R2.0*) (*transport-Dep-Fun-Rel.l ?r1.0* (λ- -. *?l2.0*)) (*transport-Dep-Fun-Rel.l ?l1.0* (λ- -. *?r2.0*)): The relations can be dependent, but the functions must be simple. This is not production ready, but a proof of concept.

The package provides a command **trp-term**, which sets up the required goals to prove a given term. See the examples in this directory for some use cases and refer to [2] for more details.

Theorem Setups **context** *transport*
begin

lemma *left-Galois-left-if-left-rel-if-partial-equivalence-rel-equivalence*:
assumes $((\leq_L) \equiv_{PER} (\leq_R))\ l\ r$
and $x \leq_L x'$
shows $x\ {}_L\lessapprox\ l\ x$
⟨*proof*⟩

definition *transport-per* $x\ y \equiv ((\leq_L) \equiv_{PER} (\leq_R))\ l\ r \wedge x\ {}_L\lessapprox\ y$

The choice of x' is arbitrary. All we need is *in-dom* $(\leq_L)\ x$.

lemma *transport-per-start*:
assumes $((\leq_L) \equiv_{PER} (\leq_R))\ l\ r$
and $x \leq_L x'$
shows *transport-per* $x\ (l\ x)$
⟨*proof*⟩

lemma *left-Galois-if-transport-per*:
assumes *transport-per* $x\ y$
shows $x\ {}_L\lessapprox\ y$
⟨*proof*⟩

end

context *transport-Fun-Rel*
begin

Simplification of Galois relator for simple function relator.

corollary *left-Galois-eq-Fun-Rel-left-Galois*:
assumes $((\leq_{L1}) \equiv_{PER} (\leq_{R1}))\ l1\ r1$
and $((\leq_{L2}) \equiv_{PER} (\leq_{R2}))\ l2\ r2$
shows $({}_L\lessapprox) = (({}_{L1}\lessapprox) \Rrightarrow ({}_{L2}\lessapprox))$
⟨*proof*⟩

end

lemmas *related-Fun-Rel-combI* = *Dep-Fun-Rel-relD*[**where** *?S*=λ- -. *S* **for** *S*, *rotated*]
lemma *related-Fun-Rel-lambdaI*:
assumes $\bigwedge x\ y.\ R\ x\ y \Longrightarrow S\ (f\ x)\ (g\ y)$

and *T* = (*R* ⇛ *S*)
shows *T f g*
⟨*proof*⟩

General ML setups ⟨*ML*⟩

Unification Setup ⟨*ML*⟩
declare [[*trp-uhint* **where** *hint-preprocessor* = ‹*Unification-Hints-Base.obj-logic-hint-preprocessor*
@{*thm atomize-eq*[*symmetric*]} (*Conv.rewr-conv* @{*thm eq-eq-True*})›]]
declare [[*trp-ucombine add* = ‹*Transport-Unification-Combine.eunif-data*
(*Transport-Unification-Hints.try-hints*
|> *Unification-Combinator.norm-unifier*
(#*norm-term Transport-Mixed-Unification.norms-first-higherp-first-comb-higher-unify*)
|> *K*)
(*Transport-Unification-Combine.default-metadata Transport-Unification-Hints.binding*)›]]

Prototype ⟨*ML*⟩

declare
transport-Dep-Fun-Rel.transport-defs[*trp-def*]
transport-Fun-Rel.transport-defs[*trp-def*]

declare

transport-Fun-Rel.partial-equivalence-rel-equivalenceI[*rotated*, *per-intro*]
transport-eq-id.partial-equivalence-rel-equivalenceI[*per-intro*]
transport-eq-restrict-id.partial-equivalence-rel-equivalence[*per-intro*]

declare
transport-id.left-Galois-eq-left[*trp-relator-rewrite*]
transport-Fun-Rel.left-Galois-eq-Fun-Rel-left-Galois[*trp-relator-rewrite*]

end

2.13 Syntax Bundles for Transport

theory *Transport-Syntax*
imports
Transport
begin

abbreviation *Galois-infix x L R r y* ≡ *galois-rel.Galois L R r x y*
abbreviation (*input*) *ge-Galois R r L* ≡ *galois-rel.ge-Galois-left L R r*
abbreviation (*input*) *ge-Galois-infix y R r L x* ≡ *ge-Galois R r L y x*

bundle *galois-rel-syntax*
begin
  notation *galois-rel.Galois* $('((\_)\lessapprox_{(\_)\ (\_)})')$
  notation *Galois-infix* $((\_)\ (\_)\lessapprox_{((\_)\ (\_))}\ (\_)\ [51,51,51,51,51]\ 50)$
  notation *ge-Galois* $('((_{(\_)\ (\_)})\gtrapprox(\_)'))$
  notation *ge-Galois-infix* $((\_)\ (_{(\_)\ (\_)})\gtrapprox(\_)\ (\_)\ [51,51,51,51,51]\ 50)$
end
bundle *no-galois-rel-syntax*
begin
  no-notation *galois-rel.Galois* $('((\_)\lessapprox_{(\_)\ (\_)})')$
  no-notation *Galois-infix* $((\_)\ (\_)\lessapprox_{((\_)\ (\_))}\ (\_)\ [51,51,51,51,51]\ 50)$
  no-notation *ge-Galois* $('((_{(\_)\ (\_)})\gtrapprox(\_)'))$
  no-notation *ge-Galois-infix* $((\_)\ (_{(\_)\ (\_)})\gtrapprox(\_)\ (\_)\ [51,51,51,51,51]\ 50)$
end

bundle *transport-syntax*
begin
  notation *transport.preorder-equivalence* (**infix** $\equiv_{pre}$ *50*)
  notation *transport.partial-equivalence-rel-equivalence* (**infix** $\equiv_{PER}$ *50*)
end
bundle *no-transport-syntax*
begin
  no-notation *transport.preorder-equivalence* (**infix** $\equiv_{pre}$ *50*)
  no-notation *transport.partial-equivalence-rel-equivalence* (**infix** $\equiv_{PER}$ *50*)
end

end

2.14 Example Transports for Dependent Function Relator

theory *Transport-Dep-Fun-Rel-Examples*
  imports
    Transport-Prototype
    Transport-Syntax
    HOL−Library.IArray
begin

Summary Dependent function relator examples from [2]. Refer to the paper for more details.

context
  includes *galois-rel-syntax transport-syntax*
  notes
   transport.rel-if-partial-equivalence-rel-equivalence-if-iff-if-partial-equivalence-rel-equivalenceI
     [*rotated*, *per-intro*]

transport-Dep-Fun-Rel-no-dep-fun.partial-equivalence-rel-equivalenceI
[*ML-Krattr* ‹*Conversion-Util.move-prems-to-front-conv* [*1*] |> *Conversion-Util.thm-conv*›,
ML-Krattr ‹*Conversion-Util.move-prems-to-front-conv* [*2*,*3*] |> *Conversion-Util.thm-conv*›,
per-intro]
begin

interpretation *transport L R l r* **for** *L R l r* ⟨*proof*⟩

abbreviation *Zpos* ≡ $((=_{(\leq)(0\ ::\ int)})\ ::\ int \Rightarrow \text{-})$

lemma *Zpos-per* [*per-intro*]: (*Zpos* $\equiv_{PER}$ (=)) *nat int*
⟨*proof*⟩

lemma *sub-parametric* [*trp-in-dom*]:
([*i* - :: *Zpos*] ⇛ [*j* - :: *Zpos* | *j* ≤ *i*] ⇛ *Zpos*) (−) (−)
⟨*proof*⟩

trp-term *nat-sub* :: *nat* ⇒ *nat* ⇒ *nat* **where** *x* = (−) :: *int* ⇒ -
and *L* = [*i* - :: *Zpos*] ⇛ [*j* - :: *Zpos* | *j* ≤ *i*] ⇛ *Zpos*
and *R* = [*n* - :: (=)] ⇛ [*m* - :: (=)| *m* ≤ *n*] ⇛ (=)

⟨*proof*⟩

thm *nat-sub-app-eq*

Note: as of now, **trp-term** does not rewrite the Galois relator of dependent function relators.

thm *nat-sub-related′*

abbreviation *LRel* ≡ *list-all2*
abbreviation *IARel* ≡ *rel-iarray*

lemma *transp-eq-transitive*: *transp* = *transitive*
⟨*proof*⟩
lemma *symp-eq-symmetric*: *symp* = *symmetric*
⟨*proof*⟩

lemma [*per-intro*]:
assumes *partial-equivalence-rel R*
shows (*LRel R* $\equiv_{PER}$ *IARel R*) *IArray.IArray IArray.list-of*
⟨*proof*⟩

lemma [*trp-in-dom*]:
([*xs* - :: *LRel R*] ⇛ [*i* - :: (=) | *i* < *length xs*] ⇛ *R*) (!) (!)
⟨*proof*⟩

context
fixes *R* :: ′*a* ⇒ - **assumes** [*per-intro*]: *partial-equivalence-rel R*
begin

interpretation *Rper* : *transport-partial-equivalence-rel-id R*
⟨*proof*⟩

declare *Rper.partial-equivalence-rel-equivalence* [*per-intro*]

trp-term *iarray-index* **where** $x = (!) :: 'a\ list \Rightarrow$ -
and $L = ([xs$ - :: $LRel\ R] \Rrightarrow [i$ - :: $(=) \mid i < length\ xs] \Rrightarrow R)$
and $R = ([xs$ - :: $IARel\ R] \Rrightarrow [i$ - :: $(=) \mid i < IArray.length\ xs] \Rrightarrow R)$
⟨*proof*⟩

end
end

end

2.15 Example Transports Between Lists and Sets

theory *Transport-Lists-Sets-Examples*
imports
Transport-Prototype
Transport-Syntax
HOL−Library.FSet
begin

Summary Introductory examples from [2]. Transports between lists and (finite) sets. Refer to the paper for more details.

context
includes *galois-rel-syntax transport-syntax*
begin

Introductory examples from paper Left and right relations.

definition *LFSL xs xs′* ≡ *fset-of-list xs* = *fset-of-list xs′*
abbreviation (*input*) (*LFSR* :: ′*a fset* ⇒ -) ≡ (=)
definition *LSL xs xs′* ≡ *set xs* = *set xs′*
abbreviation (*input*) (*LSR* :: ′*a set* ⇒ -) ≡ $(=_{finite} :: 'a\ set \Rightarrow bool)$

interpretation *t* : *transport LSL R l r* **for** *LSL R l r* ⟨*proof*⟩

Proofs of equivalences.

lemma *list-fset-PER* [*per-intro*]: $(LFSL \equiv_{PER} LFSR)$ *fset-of-list sorted-list-of-fset*
⟨*proof*⟩

lemma *list-set-PER* [*per-intro*]: $(LSL \equiv_{PER} LSR)$ *set sorted-list-of-set*
⟨*proof*⟩

We can rewrite the Galois relators in the following theorems to the relator of the paper.

definition *LFS xs s* ≡ *fset-of-list xs* = *s*
definition *LS xs s* ≡ *set xs* = *s*

lemma *LFSL-Galois-eq-LFS*: $(_{LFSL}\lessapprox_{LFSR\ sorted\text{-}list\text{-}of\text{-}fset}) \equiv LFS$
⟨*proof*⟩
lemma *LFSR-Galois-eq-inv-LFS*: $(_{LFSR}\lessapprox_{LFSL\ fset\text{-}of\text{-}list}) \equiv LFS^{-1}$
⟨*proof*⟩
lemma *LSL-Galois-eq-LS*: $(_{LSL}\lessapprox_{LSR\ sorted\text{-}list\text{-}of\text{-}set}) \equiv LS$
⟨*proof*⟩

declare *LFSL-Galois-eq-LFS*[*trp-relator-rewrite*, *trp-uhint*]
LFSR-Galois-eq-inv-LFS[*trp-relator-rewrite*, *trp-uhint*]
LSL-Galois-eq-LS[*trp-relator-rewrite*, *trp-uhint*]

definition *max-list xs* ≡ *foldr max xs* (*0* :: *nat*)

Proof of parametricity for *max-list*.

lemma *max-max-list-removeAll-eq-maxlist*:
assumes *x* ∈ *set xs*
shows *max x* (*max-list* (*removeAll x xs*)) = *max-list xs*
⟨*proof*⟩

lemma *max-list-parametric* [*trp-in-dom*]: (*LSL* ⇛ (=)) *max-list max-list*
⟨*proof*⟩

lemma *LFSL-eq-LSL*: *LFSL* ≡ *LSL*
⟨*proof*⟩

lemma *max-list-parametricfin* [*trp-in-dom*]: (*LFSL* ⇛ (=)) *max-list max-list*
⟨*proof*⟩

Transport from lists to finite sets.

trp-term *max-fset* :: *nat fset* ⇒ *nat* **where** *x* = *max-list*
and *L* = (*LFSL* ⇛ (=))
⟨*proof*⟩

Use **print-theorems** to show all theorems. Here's the correctness theorem:

lemma (*LFS* ⇛ (=)) *max-list max-fset* ⟨*proof*⟩

lemma [*trp-in-dom*]: (*LFSR* ⇛ (=)) *max-fset max-fset* ⟨*proof*⟩

Transport from lists to sets.

trp-term *max-set* :: *nat set* ⇒ *nat* **where** *x* = *max-list*
⟨*proof*⟩

lemma (*LS* ⇛ (=)) *max-list max-set* ⟨*proof*⟩

The registration of symmetric equivalence rules is not done by default as of now, but that would not be a problem in principle.

lemma *list-fset-PER-sym* [*per-intro*]:
$(LFSR \equiv_{PER} LFSL)$ *sorted-list-of-fset fset-of-list*
⟨*proof*⟩

Transport from finite sets to lists.

trp-term *max-list′* :: *nat list* ⇒ *nat* **where** x = *max-fset*
⟨*proof*⟩

lemma $(LFS^{-1} \Rrightarrow (=))$ *max-fset max-list′* ⟨*proof*⟩

Transporting higher-order functions.

lemma *map-parametric* [*trp-in-dom*]:
$(((=) \Rrightarrow (=)) \Rrightarrow LSL \Rrightarrow LSL)$ *map map*
⟨*proof*⟩

lemma [*trp-uhint*]: $P \equiv (=) \Longrightarrow P \equiv (=) \Rrightarrow (=)$ ⟨*proof*⟩
lemma [*trp-uhint*]: $P \equiv \top \Longrightarrow (=_{P\ ::\ 'a \Rightarrow bool}) \equiv ((=) :: 'a \Rightarrow -)$ ⟨*proof*⟩

trp-term *map-set* :: (*′a* :: *linorder* ⇒ *′b*) ⇒ *′a set* ⇒ (*′b* :: *linorder*) *set*
where x = *map* :: (*′a* :: *linorder* ⇒ *′b*) ⇒ *′a list* ⇒ (*′b* :: *linorder*) *list*
⟨*proof*⟩

lemma $(((=) \Rrightarrow (=)) \Rrightarrow LS \Rrightarrow LS)$ *map map-set* ⟨*proof*⟩

lemma *filter-parametric* [*trp-in-dom*]:
$(((=) \Rrightarrow (\longleftrightarrow)) \Rrightarrow LSL \Rrightarrow LSL)$ *filter filter*
⟨*proof*⟩

trp-term *filter-set* :: (*′a* :: *linorder* ⇒ *bool*) ⇒ *′a set* ⇒ *′a set*
where x = *filter* :: (*′a* :: *linorder* ⇒ *bool*) ⇒ *′a list* ⇒ *′a list*
⟨*proof*⟩

lemma $(((=) \Rrightarrow (=)) \Rrightarrow LS \Rrightarrow LS)$ *filter filter-set* ⟨*proof*⟩

lemma *append-parametric* [*trp-in-dom*]:
$(LSL \Rrightarrow LSL \Rrightarrow LSL)$ *append append*
⟨*proof*⟩

trp-term *append-set* :: (*′a* :: *linorder*) *set* ⇒ *′a set* ⇒ *′a set*
where x = *append* :: (*′a* :: *linorder*) *list* ⇒ *′a list* ⇒ *′a list*
⟨*proof*⟩

lemma $(LS \Rrightarrow LS \Rrightarrow LS)$ *append append-set* ⟨*proof*⟩

The prototype also provides a simplified definition.

lemma *append-set s s′* ≡ *set* (*sorted-list-of-set s*) ∪ *set* (*sorted-list-of-set s′*)
⟨*proof*⟩

lemma *finite* $s \Longrightarrow$ *finite* $s' \Longrightarrow$ *append-set* $s\ s' = s \cup s'$
⟨*proof*⟩

end

end

2.16 Transport for Partial Quotient Types

theory *Transport-Partial-Quotient-Types*
imports
HOL.Lifting
Transport
begin

Summary Every partial quotient type *Quotient*, as used by the Lifting package, is transportable.

context *transport*
begin

interpretation t : *transport* $L\ (=)\ l\ r$ ⟨*proof*⟩

lemma *Quotient-T-eq-Galois*:
assumes *Quotient* $(\leq_L)\ l\ r\ T$
shows $T = t.Galois$
⟨*proof*⟩

lemma *Quotient-if-preorder-equivalence*:
assumes $((\leq_L) \equiv_{pre} (=))\ l\ r$
shows *Quotient* $(\leq_L)\ l\ r\ t.Galois$
⟨*proof*⟩

lemma *partial-equivalence-rel-equivalence-if-Quotient*:
assumes *Quotient* $(\leq_L)\ l\ r\ T$
shows $((\leq_L) \equiv_{PER} (=))\ l\ r$
⟨*proof*⟩

corollary *Quotient-iff-partial-equivalence-rel-equivalence*:
Quotient $(\leq_L)\ l\ r\ t.Galois \longleftrightarrow ((\leq_L) \equiv_{PER} (=))\ l\ r$
⟨*proof*⟩

corollary *Quotient-T-eq-ge-Galois-right*:
assumes *Quotient* $(\leq_L)\ l\ r\ T$
shows $T = t.ge\text{-}Galois\text{-}right$
⟨*proof*⟩

end

end

2.17 Transport for HOL Type Definitions

theory *Transport-Typedef-Base*
  imports
    HOL-Alignment-Binary-Relations
    Transport-Bijections
    HOL.Typedef
begin

context *type-definition*
begin

abbreviation (*input*) $L :: 'a \Rightarrow 'a \Rightarrow bool \equiv (=_A)$
abbreviation (*input*) $R :: 'b \Rightarrow 'b \Rightarrow bool \equiv (=)$

sublocale *transport?* :
  transport-eq-restrict-bijection mem-of A $\top :: 'b \Rightarrow bool$ *Abs Rep*
  rewrites $(=_{mem\text{-}of\ A}) \equiv L$
  and $(=_{\top\ ::\ 'b \Rightarrow bool}) \equiv R$
  and $(galois\text{-}rel.Galois\ (=)\ (=)\ Rep)\restriction_{mem\text{-}of\ A}\upharpoonleft_{\top\ ::\ 'b \Rightarrow bool} \equiv$
    $(galois\text{-}rel.Galois\ (=)\ (=)\ Rep)$
  ⟨*proof*⟩

interpretation *galois L R Abs Rep* ⟨*proof*⟩

lemma *Rep-left-Galois-self*: $Rep\ y\ {}_L\lessapprox\ y$
  ⟨*proof*⟩

definition $AR\ x\ y \equiv x = Rep\ y$

lemma *left-Galois-eq-AR*: $left\text{-}Galois = AR$
  ⟨*proof*⟩

end

end

theory *Transport-Typedef*
  imports
    HOL−Library.FSet
    Transport-Typedef-Base
    Transport-Prototype
    Transport-Syntax

begin

context
includes *galois-rel-syntax transport-syntax*
begin

typedef *pint* = $\{i :: int.\ 0 \le i\}$ ⟨*proof*⟩

interpretation *typedef-pint* : *type-definition Rep-pint Abs-pint* $\{i :: int.\ 0 \le i\}$
⟨*proof*⟩

lemma [*trp-relator-rewrite*, *trp-uhint*]:
$({(=_{Collect\ ((\le)\ (0 :: int))})} \lessapprox_{(=)\ Rep\text{-}pint}) \equiv typedef\text{-}pint.AR$
⟨*proof*⟩

typedef *'a fset* = $\{s :: 'a\ set.\ finite\ s\}$ ⟨*proof*⟩

interpretation *typedef-fset* :
type-definition Rep-fset Abs-fset $\{s :: 'a\ set.\ finite\ s\}$
⟨*proof*⟩

lemma [*trp-relator-rewrite*, *trp-uhint*]:
$({(=_{\{s :: 'a\ set.\ finite\ s\}}) :: 'a\ set \Rightarrow \text{-}} \lessapprox_{(=)\ Rep\text{-}fset}) \equiv typedef\text{-}fset.AR$
⟨*proof*⟩

lemma *eq-restrict-set-eq-eq-uhint* [*trp-uhint*]:
$P \equiv \lambda x.\ x \in A \Longrightarrow ((=_{A :: 'a\ set}) :: 'a \Rightarrow \text{-}) \equiv (=_P)$
⟨*proof*⟩

declare
typedef-pint.partial-equivalence-rel-equivalence[*per-intro*]
typedef-fset.partial-equivalence-rel-equivalence[*per-intro*]

lemma *one-parametric* [*trp-in-dom*]: *typedef-pint.L 1 1* ⟨*proof*⟩

trp-term *pint-one* :: *pint* **where** *x* = *1* :: *int*
⟨*proof*⟩

lemma *add-parametric* [*trp-in-dom*]:
(*typedef-pint.L* ⇛ *typedef-pint.L* ⇛ *typedef-pint.L*) (+) (+)
⟨*proof*⟩

trp-term *pint-add* :: *pint* ⇒ *pint* ⇒ *pint*
where *x* = (+) :: *int* ⇒ -
⟨*proof*⟩

lemma *empty-parametric* [*trp-in-dom*]: *typedef-fset.L* {} {}
⟨*proof*⟩

trp-term *fempty* :: *'a fset* **where** $x = \{\}$ :: *'a set*
⟨*proof*⟩

lemma *insert-parametric* [*trp-in-dom*]:
$((=) \Rrightarrow$ *typedef-fset.L* $\Rrightarrow$ *typedef-fset.L*) *insert insert*
⟨*proof*⟩

trp-term *finsert* :: *'a* ⇒ *'a fset* ⇒ *'a fset* **where** $x =$ *insert*
and $L = ((=) \Rrightarrow$ *typedef-fset.L* $\Rrightarrow$ *typedef-fset.L*)
and $R = ((=) \Rrightarrow$ *typedef-fset.R* $\Rrightarrow$ *typedef-fset.R*)
⟨*proof*⟩

context
notes *refl*[*trp-related-intro*]
begin

trp-term *insert-add-int-fset-whitebox* :: *int fset*
where $x =$ *insert* (*1* + (*1* :: *int*)) {} !
⟨*proof*⟩

lemma *empty-parametric'* [*trp-related-intro*]: (*rel-set R*) {} {}
⟨*proof*⟩

lemma *insert-parametric'* [*trp-related-intro*]:
($R \Rrightarrow$ *rel-set R* $\Rrightarrow$ *rel-set R*) *insert insert*
⟨*proof*⟩

context
assumes [*trp-uhint*]:

$L \equiv$ *rel-set* (*L1* :: *int* ⇒ *int* ⇒ *bool*) $\Longrightarrow R \equiv$ *rel-set* (*R1* :: *pint* ⇒ *pint* ⇒ *bool*)
$\Longrightarrow r \equiv$ *image r1* $\Longrightarrow S \equiv (_{L1}\lessapprox_{R1\ r1}) \Longrightarrow (_{L}\lessapprox_{R\ r}) \equiv$ *rel-set S*
begin

trp-term *insert-add-pint-set-whitebox* :: *pint set*
where $x =$ *insert* (*1* + (*1* :: *int*)) {} !
⟨*proof*⟩

print-statement *insert-add-int-fset-whitebox-def insert-add-pint-set-whitebox-def*

end
end

lemma *image-parametric* [*trp-in-dom*]:
$(((=) \Rrightarrow (=)) \Rrightarrow \textit{typedef-fset.L} \Rrightarrow \textit{typedef-fset.L})$ *image image*
⟨*proof*⟩

trp-term *fimage* :: $('a \Rightarrow 'b) \Rightarrow 'a$ *fset* $\Rightarrow 'b$ *fset* **where** $x = \textit{image}$
⟨*proof*⟩

lemma *rel-fun-eq-Fun-Rel-rel*: *rel-fun* = *Fun-Rel-rel*
⟨*proof*⟩

lemma *image-parametric'* [*trp-related-intro*]:
$((R \Rrightarrow S) \Rrightarrow \textit{rel-set } R \Rrightarrow \textit{rel-set } S)$ *image image*
⟨*proof*⟩

lemma *Galois-id-hint* [*trp-uhint*]:
$(L :: 'a \Rightarrow 'a \Rightarrow bool) \equiv R \Longrightarrow r \equiv id \Longrightarrow E \equiv L \Longrightarrow ({}_{L}\lessapprox_{R\ r}) \equiv E$
⟨*proof*⟩

lemma *Freq* [*trp-uhint*]: $L \equiv (=) \Rrightarrow (=) \Longrightarrow L \equiv (=)$
⟨*proof*⟩

context
fixes *L1 R1 l1 r1 L R l r*
assumes *per1*: $(L1 \equiv_{PER} R1)$ *l1 r1*
defines $L \equiv$ *rel-set L1* **and** $R \equiv$ *rel-set R1*
and $l \equiv$ *image l1* **and** $r \equiv$ *image r1*
begin

interpretation *transport L R l r* ⟨*proof*⟩

context

assumes *transport-per-set*: $((\le_L) \equiv_{PER} (\le_R))$ *l r*
and *compat*: *transport-comp.middle-compatible-codom R typedef-fset.L*
begin

trp-term *fempty-param* :: *'b fset*
where $x = \{\} :: 'a$ *set*
and $L =$ *transport-comp.L ?L1 ?R1* (*?l1* :: *'a set* ⇒ *'b set*) *?r1 typedef-fset.L*
and $R =$ *transport-comp.L typedef-fset.R typedef-fset.L ?r2 ?l2 ?R1*
⟨*proof*⟩

definition *set-succ* ≡ *image* ((+) (*1* :: *int*))

lemma *set-succ-parametric* [*trp-in-dom*]:
(*typedef-fset.L* ⇛ *typedef-fset.L*) *set-succ set-succ*

⟨*proof*⟩

trp-term *fset-succ* :: *int fset* ⇒ *int fset*
where x = *set-succ*
and L = *typedef-fset.L* ⇛ *typedef-fset.L*
and R = *typedef-fset.R* ⇛ *typedef-fset.R*
⟨*proof*⟩

trp-term *fset-succ′* :: *int fset* ⇒ *int fset*
where x = *set-succ*
and L = *typedef-fset.L* ⇛ *typedef-fset.L*
and R = *typedef-fset.R* ⇛ *typedef-fset.R*
unfold set-succ-def !

⟨*proof*⟩

lemma *pint-middle-compat*:
transport-comp.middle-compatible-codom (*rel-set* ((=) :: *pint* ⇒ -))
($=_{Collect}$ (*finite* :: *pint set* ⇒ -))
⟨*proof*⟩

trp-term *pint-fset-succ* :: *pint fset* ⇒ *pint fset*
where x = *set-succ* :: *int set* ⇒ *int set*

⟨*proof*⟩

end
end
end

end

2.18 Transport Paper Guide

theory *Transport-Via-Partial-Galois-Connections-Equivalences-Paper*
imports
Transport
Transport-Natural-Functors
Transport-Partial-Quotient-Types
Transport-Prototype
Transport-Lists-Sets-Examples
Transport-Dep-Fun-Rel-Examples
Transport-Typedef-Base
begin

- Section 3.1: Order basics can be found in *Transport.Binary-Relation-Properties*, *Transport.Preorders*, *Transport.Partial-Equivalence-Relations*, *Transport.Equivalence-Relations*, and *Transport.Order-Functions-Base*. Theorem

- Section 3.2: Function relators and monotonicity can be found in *Transport.Function-Relators* and *Transport.Functions-Monotone*

- Section 3.3: Galois relator can be found in *Transport.Galois-Relator-Base*.

  - Lemma 1: *Transport.Transport-Partial-Quotient-Types* (*results from Appendix*)
  - Lemma 3: *galois-prop.galois-prop ?L ?R ?l ?r* $\Longrightarrow$ (*galois-rel.Galois* $?R^{-1}$ $?L^{-1}$ $?l)^{-1}$ *?x ?y* = *Galois-infix ?x ?L ?R ?r ?y*

- Section 3.4: Partial Galois Connections and Equivalences can be found in *Transport.Half-Galois-Property*, *Transport.Galois-Property*, *Transport.Galois-Connections*, *Transport.Galois-Equivalences*, and *Transport.Order-Equivalences*.

  - Lemma 2: *Transport.Transport-Partial-Quotient-Types* (*results from Appendix*)
  - Lemma 4: ⟦*order-functors.order-equivalence ?L ?R ?l ?r*; *transitive ?L*; *transitive ?R*⟧ $\Longrightarrow$ *galois.galois-equivalence ?L ?R ?l ?r*
  - Lemma 5: ⟦*galois.galois-equivalence ?L ?R ?l ?r*; *reflexive-on* (*in-field ?L*) *?L*; *reflexive-on* (*in-field ?R*) *?R*⟧ $\Longrightarrow$ *order-functors.order-equivalence ?L ?R ?l ?r*

- Section 4.1: Closure (Dependent) Function Relator can be found in `Functions`.

  - Monotone function relator *Transport.Monotone-Function-Relator*.
  - Setup of construction *Transport.Transport-Functions-Base*.
  - Theorem 1: see *Transport.Transport-Functions*
  - Theorem 2: see ⟦*transport.preorder-equivalence ?L1.0 ?R1.0 ?l1.0 ?r1.0*; $\bigwedge$*x x'*. *Galois-infix x ?L1.0 ?R1.0 ?r1.0 x'* $\Longrightarrow$ *transport.preorder-equivalence* (*?L2.0 x* (*?r1.0 x'*)) (*?R2.0* (*?l1.0 x*) *x'*) (*?l2.0 x' x*) (*?r2.0 x x'*); ([*x1 x2* :: $?L1.0^{-1}$] $\Rrightarrow_m$ [*x3 x4* :: *?L1.0*] $\Rrightarrow$ *?L1.0 x1 x3* $\longrightarrow$ ($\le$)) *?L2.0*; ([*x1 x2* :: *?L1.0*] $\Rrightarrow_m$ [*x1' x2'* :: *?R1.0*] $\Rrightarrow$ *Galois-infix x2 ?L1.0 ?R1.0 ?r1.0 x1'* $\longrightarrow$ ([*in-field* (*?R2.0* (*?l1.0 x1*) *x2'*)] $\Rrightarrow$ *?L2.0 x1* (*?r1.0 x2'*))) *?r2.0*; *in-dom* (*transport-Mono-Dep-Fun-Rel.L ?L1.0 ?L2.0*) *?f*; *in-codom* (*transport-Mono-Dep-Fun-Rel.L ?R1.0 ?R2.0*) *?g*⟧ $\Longrightarrow$ *Galois-infix ?f* (*transport-Mono-Dep-Fun-Rel.L ?L1.0 ?L2.0*) (*transport-Mono-Dep-Fun-R* *?R1.0 ?R2.0*) (*transport-Dep-Fun-Rel.l ?l1.0 ?r2.0*) *?g* = ([*x*

x′ :: galois-rel.Galois ?L1.0 ?R1.0 ?r1.0] ⇛ *galois-rel.Galois* (*?L2.0 x* (*?r1.0 x′*)) (*?R2.0* (*?l1.0 x*) *x′*) (*?r2.0 x x′*)) *?f ?g* (*results from Appendix*)

- Lemma 6: ⟦*galois.galois-connection ?L1.0 ?R1.0 ?l1.0 ?r1.0*; *reflexive-on* (*in-codom ?L1.0*) *?L1.0*; *reflexive-on* (*in-dom ?R1.0*) *?R1.0*; *galois.galois-connection ?L2.0 ?R2.0 ?l2.0 ?r2.0*; *transitive ?L2.0*; *transitive ?R2.0*⟧ ⟹ *galois.galois-connection* (*transport-Mono-Dep-Fun-Rel.L ?L1.0* (λ- -. *?L2.0*)) (*transport-Mono-Dep-Fun-Rel.L ?R1.0* (λ- -. *?R2.0*)) (*transport-Dep-Fun-Rel.l ?r1.0* (λ- -. *?l2.0*)) (*transport-Dep-Fun-Rel.l ?l1.0* (λ- -. *?r2.0*))
- Lemma 7: ⟦(*?L1.0* $\Rrightarrow_m$ *?R1.0*) *?l1.0*; *galois-prop.galois-prop ?L1.0 ?R1.0 ?l1.0 ?r1.0*; *reflexive-on* (*in-dom ?L1.0*) *?L1.0*; (*?R2.0* $\Rrightarrow_m$ *?L2.0*) *?r2.0*; *transitive ?L2.0*; *in-dom* (*transport-Mono-Dep-Fun-Rel.L ?L1.0* (λ- -. *?L2.0*)) *?f*; *in-codom* (*transport-Mono-Dep-Fun-Rel.L ?R1.0* (λ- -. *?R2.0*)) *?g*⟧ ⟹ *Galois-infix ?f* (*transport-Mono-Dep-Fun-Rel.L ?L1.0* (λ- -. *?L2.0*)) (*transport-Mono-Dep-Fun-Rel.L ?R1.0* (λ- -. *?R2.0*)) (*transport-Dep-Fun-Rel.l ?l1.0* (λ- -. *?r2.0*)) *?g* = (*galois-rel.Galois ?L1.0 ?R1.0 ?r1.0* ⇛ *galois-rel.Galois ?L2.0 ?R2.0 ?r2.0*) *?f ?g*
- Theorem 7: ⟦*galois.galois-connection ?L1.0 ?R1.0 ?l1.0 ?r1.0*; *reflexive-on* (*in-field ?L1.0*) *?L1.0*; *reflexive-on* (*in-field ?R1.0*) *?R1.0*; ⋀*x x′*. *Galois-infix x ?L1.0 ?R1.0 ?r1.0 x′* ⟹ *galois.galois-connection* (*?L2.0 x* (*?r1.0 x′*)) (*?R2.0* (*?l1.0 x*) *x′*) (*?l2.0 x′ x*) (*?r2.0 x x′*); ([- *x2* :: *?L1.0*] $\Rrightarrow_m$ [*x3 x4* :: *?L1.0*] ⇛ (*?L1.0 x2 x3* ∧ *?L1.0 x4* (*order-functors.unit ?l1.0 ?r1.0 x3*)) ⟶ (λ*x y*. *y* ≤ *x*)) *?L2.0*; ([*x1′ x2′* :: *?R1.0*] $\Rrightarrow_m$ *?R1.0* (*order-functors.counit ?l1.0 ?r1.0 x2′*) *x1′* ⟶ ([*x3′* - :: *?R1.0*] ⇛ *?R1.0 x2′ x3′* ⟶ (≤))) *?R2.0*; ([*x1′ x2′* :: *?R1.0*] $\Rrightarrow_m$ [*x1 x2* :: *?L1.0*] ⇛ *Galois-infix x2 ?L1.0 ?R1.0 ?r1.0 x1′* ⟶ ([*in-field* (*?L2.0 x1* (*?r1.0 x2′*))] ⇛ *?R2.0* (*?l1.0 x1*) *x2′*)) *?l2.0*; ([*x1 x2* :: *?L1.0*] $\Rrightarrow_m$ [*x1′ x2′* :: *?R1.0*] ⇛ *Galois-infix x2 ?L1.0 ?R1.0 ?r1.0 x1′* ⟶ ([*in-field* (*?R2.0* (*?l1.0 x1*) *x2′*)] ⇛ *?L2.0 x1* (*?r1.0 x2′*))) *?r2.0*; ⋀*x1 x2*. *?L1.0 x1 x2* ⟹ *transitive* (*?L2.0 x1 x2*); ⋀*x1′ x2′*. *?R1.0 x1′ x2′* ⟹ *transitive* (*?R2.0 x1′ x2′*)⟧ ⟹ *galois.galois-connection* (*transport-Mono-Dep-Fun-Rel.L ?L1.0 ?L2.0*) (*transport-Mono-Dep-Fun-Rel.L ?R1.0 ?R2.0*) (*transport-Dep-Fun-Rel.l ?r1.0 ?l2.0*) (*transport-Dep-Fun-Rel.l ?l1.0 ?r2.0*)
- Theorem 8: ⟦*galois.galois-connection ?L1.0 ?R1.0 ?l1.0 ?r1.0*; *reflexive-on* (*in-field ?L1.0*) *?L1.0*; ⋀*x x′*. *Galois-infix x ?L1.0 ?R1.0 ?r1.0 x′* ⟹ (*?R2.0* (*?l1.0 x*) *x′* $\Rrightarrow_m$ *?L2.0 x* (*?r1.0 x′*)) (*?r2.0 x x′*); ([*x1* :: ⊤] $\Rrightarrow_m$ [*x2* - :: *?L1.0*] $\Rrightarrow_m$ *?L1.0 x1 x2* ⟶ (≤)) *?L2.0*; ([*x1* :: ⊤] $\Rrightarrow_m$ [*x2 x3* :: *?L1.0*] $\Rrightarrow_m$ (*?L1.0 x1 x2* ∧ *?L1.0 x3* (*order-functors.unit ?l1.0 ?r1.0 x2*)) ⟶ (λ*x y*. *y* ≤ *x*))

?L2.0; ([*x1 x2* :: *?L1.0*] $\Rrightarrow_m$ [*x1′ x2′* :: *?R1.0*] ⇛ *Galois-infix x2 ?L1.0 ?R1.0 ?r1.0 x1′* ⟶ ([*in-field* (*?R2.0* (*?l1.0 x1*) *x2′*)] ⇛ *?L2.0 x1* (*?r1.0 x2′*))) *?r2.0*; ⋀*x1 x2*. *?L1.0 x1 x2* ⟹ *transitive* (*?L2.0 x1 x2*); *in-dom* (*transport-Mono-Dep-Fun-Rel.L ?L1.0 ?L2.0*) *?f*; *in-codom* (*transport-Mono-Dep-Fun-Rel.L ?R1.0 ?R2.0*) *?g*⟧ ⟹ *Galois-infix ?f* (*transport-Mono-Dep-Fun-Rel.L ?L1.0 ?L2.0*) (*transport-Mono-Dep-Fun-Rel.L ?R1.0 ?R2.0*) (*transport-Dep-Fun-Rel.l ?l1.0 ?r2.0*) *?g* = ([*x x′* :: *galois-rel.Galois ?L1.0 ?R1.0 ?r1.0*] ⇛ *galois-rel.Galois* (*?L2.0 x* (*?r1.0 x′*)) (*?R2.0* (*?l1.0 x*) *x′*) (*?r2.0 x x′*)) *?f ?g*

  - Lemma 8 ⟦*galois.galois-equivalence ?L1.0 ?R1.0 ?l1.0 ?r1.0*; *Preorders.preorder-on* (*in-field ?L1.0*) *?L1.0*; ([*x1 x2* :: $?L1.0^{-1}$] $\Rrightarrow_m$ [*x3 x4* :: *?L1.0*] ⇛ *?L1.0 x1 x3* ⟶ (≤)) *?L2.0*; ⋀*x1 x2*. *?L1.0 x1 x2* ⟹ *partial-equivalence-rel* (*?L2.0 x1 x2*)⟧ ⟹ *transport-Mono-Dep-Fun-Rel.L ?L1.0 ?L2.0* = *transport-Dep-Fun-Rel.L ?L1.0 ?L2.0*
  - Lemma 9: ⟦*reflexive-on* (*in-field ?L1.0*) *?L1.0*; *partial-equivalence-rel ?L2.0*⟧ ⟹ *transport-Mono-Dep-Fun-Rel.L ?L1.0* (λ- -. *?L2.0*) = *transport-Dep-Fun-Rel.L ?L1.0* (λ- -. *?L2.0*)

- Section 4.2: Closure Natural Functors can be found in `Natural_Functors`.
  - Theorem 3: see *Transport.Transport-Natural-Functors*
  - Theorem 4: *galois-rel.Galois* (*transport-natural-functor.L ?L1.0 ?L2.0 ?L3.0*) (*transport-natural-functor.L ?R1.0 ?R2.0 ?R3.0*) (*transport-natural-functor.l ?r1.0 ?r2.0 ?r3.0*) = *Frel* (*galois-rel.Galois ?L1.0 ?R1.0 ?r1.0*) (*galois-rel.Galois ?L2.0 ?R2.0 ?r2.0*) (*galois-rel.Galois ?L3.0 ?R3.0 ?r3.0*)

- Section 4.3: Closure Compositions can be found in `Compositions`.
  - Setup for simple case in *Transport.Transport-Compositions-Agree-Base*
  - Setup for generic case in *Transport.Transport-Compositions-Generic-Base*
  - Theorem 5: ⟦*transport.preorder-equivalence ?L1.0 ?R1.0 ?l1.0 ?r1.0*; *transport.preorder-equivalence ?L2.0 ?R2.0 ?l2.0 ?r2.0*; *transport-comp.middle-compatible-codom ?R1.0 ?L2.0*⟧ ⟹ *transport.preorder-equivalence* (*transport-comp.L ?L1.0 ?R1.0 ?l1.0 ?r1.0 ?L2.0*) (*transport-comp.L ?R2.0 ?L2.0 ?r2.0 ?l2.0 ?R1.0*) (*transport-comp.l ?l1.0 ?l2.0*) (*transport-comp.l ?r2.0 ?r1.0*) and ⟦*transport.partial-equivalence-rel-equivalence ?L1.0 ?R1.0 ?l1.0 ?r1.0*; *transport.partial-equivalence-rel-equivalence ?L2.0 ?R2.0 ?l2.0 ?r2.0*; *transport-comp.middle-compatible-codom ?R1.0 ?L2.0*⟧ ⟹ *transport.partial-equivalence-rel-equivalence* (*transport-comp.L*

?L1.0 ?R1.0 ?l1.0 ?r1.0 ?L2.0) (*transport-comp.L ?R2.0 ?L2.0 ?r2.0 ?l2.0 ?R1.0*) (*transport-comp.l ?l1.0 ?l2.0*) (*transport-comp.l ?r2.0 ?r1.0*)

  - Theorem 6: ⟦*transport.preorder-equivalence ?L1.0 ?R1.0 ?l1.0 ?r1.0*; *transport.preorder-galois-connection ?R2.0 ?L2.0 ?r2.0 ?l2.0*; *transport-comp.middle-compatible-codom ?R1.0 ?L2.0*⟧ ⟹ *galois-rel.Galois* (*transport-comp.L ?L1.0 ?R1.0 ?l1.0 ?r1.0 ?L2.0*) (*transport-comp.L ?R2.0 ?L2.0 ?r2.0 ?l2.0 ?R1.0*) (*transport-comp.l ?r2.0 ?r1.0*) = *galois-rel.Galois ?L1.0 ?R1.0 ?r1.0* ∘∘ *galois-rel.Galois ?L2.0 ?R2.0 ?r2.0*

    (*results from Appendix*)
  - Theorem 9: see `Compositions/Agree`, results in *transport-comp-same*.
  - Theorem 10: ⟦*galois.galois-equivalence ?L1.0 ?R1.0 ?l1.0 ?r1.0*; *Preorders.preorder-on* (*in-field ?R1.0*) *?R1.0*; *galois.galois-equivalence ?L2.0 ?R2.0 ?l2.0 ?r2.0*; *Preorders.preorder-on* (*in-field ?L2.0*) *?L2.0*; *transport-comp.middle-compatible-codom ?R1.0 ?L2.0*⟧ ⟹ *galois.galois-connection* (*transport-comp.L ?L1.0 ?R1.0 ?l1.0 ?r1.0 ?L2.0*) (*transport-comp.L ?R2.0 ?L2.0 ?r2.0 ?l2.0 ?R1.0*) (*transport-comp.l ?l1.0 ?l2.0*) (*transport-comp.l ?r2.0 ?r1.0*)
  - Theorem 11: ⟦(*?R1.0* $\Rrightarrow_m$ *?L1.0*) *?r1.0*; *galois-prop.galois-prop ?L1.0 ?R1.0 ?l1.0 ?r1.0*; *galois-prop.half-galois-prop-right ?R1.0 ?L1.0 ?r1.0 ?l1.0*; *Preorders.preorder-on* (*in-field ?R1.0*) *?R1.0*; (*?L2.0* $\Rrightarrow_m$ *?R2.0*) *?l2.0*; *galois-prop.half-galois-prop-left ?R2.0 ?L2.0 ?r2.0 ?l2.0*; *reflexive-on* (*in-dom ?L2.0*) *?L2.0*; *?R1.0* ∘∘ *?L2.0* ∘∘ *?R1.0* ≤ *?R1.0* ∘∘ *?L2.0*; *in-codom* (*?L2.0* ∘∘ *?R1.0* ∘∘ *?L2.0*) ≤ *in-codom ?R1.0*⟧ ⟹ *galois-rel.Galois* (*transport-comp.L ?L1.0 ?R1.0 ?l1.0 ?r1.0 ?L2.0*) (*transport-comp.L ?R2.0 ?L2.0 ?r2.0 ?l2.0 ?R1.0*) (*transport-comp.l ?r2.0 ?r1.0*) = *galois-rel.Galois ?L1.0 ?R1.0 ?r1.0* ∘∘ *galois-rel.Galois ?L2.0 ?R2.0 ?r2.0*

- Section 5:

  - Implementation *Transport.Transport-Prototype*: Note: the command "trp" from the paper is called **trp-term** and the method "trprover" is called "trp_term_prover".
  - Example 1: *Transport.Transport-Lists-Sets-Examples*
  - Example 2: *Transport.Transport-Dep-Fun-Rel-Examples*
  - Example 3: https://github.com/kappelmann/Isabelle-Set/blob/fdf59444d9a53b5279080fb4d24893c9efa31160/Isabelle_Set/Integers/Integers_Transport.thy

- Proof: Partial Quotient Types are a special case: *Transport.Transport-Partial-Quotient-Types*
- Proof: Typedefs are a special case: *Transport.Transport-Typedef-Base*
- Proof: Set-Extensions are a special case: https://github.com/kappelmann/Isabelle-Set/blob/fdf59444d9a53b5279080fb4d24893c9efa31160/Isabelle_Set/Set_Extensions/Set_Extensions_Transport.thy
- Proof: Bijections as special case: *Transport.Transport-Bijections*

end

Bibliography

[1] Huffman, Brian and Kunčar, Ondřej. Lifting and Transfer: A modular design for quotients in Isabelle/HOL. In *Certified Programs and Proofs: Third International Conference, CPP 2013, Melbourne, VIC, Australia, December 11-13, 2013, Proceedings 3*, pages 131–146. Springer, 2013.

[2] Kevin Kappelmann. Transport via Partial Galois Connections and Equivalences, 2023.