

INSTITUTE OF MATHEMATICS FOR INDUSTRY,
KYUSHU UNIVERSITY

LOGIC AND COMPUTATION PROJECT

Coq Modules for Relational Calculus

(Ver.0.1)

Hisaharu TANAKA
Saga University

Toshiaki MATSUSHIMA
Kyushu University

Shuichi INOKUCHI
Fukuoka Institute of Technology

Yoshihiro MIZOGUCHI
Kyushu University

September 26, 2015

Repository: `svn://sakura.imi.kyushu-u.ac.jp/Coq/RelationalCalculus/branches/20150716/`

Contents

1	Library <code>Basic_Notations</code>	4
1.1	このライブラリについて	4
2	Library <code>Basic_Notations_Rel</code>	5
2.1	定義	5
2.2	関数の定義	7
2.3	関係の公理	8
2.3.1	Dedekind 圏の公理	8
2.3.2	排中律	11
2.3.3	単域	12
2.3.4	点公理	12
2.3.5	関係の有理性	13
2.3.6	直和と直積	13
3	Library <code>Basic_Notations_Set</code>	14
3.1	定義	14
3.2	関数の定義	16
3.3	関係の公理	16
3.3.1	Dedekind 圏の公理	16
3.3.2	排中律	25
3.3.3	単域	25
3.3.4	点公理	26
3.3.5	関係の有理性	27
3.3.6	直和と直積	31
4	Library <code>Basic_Lemmas</code>	35
4.1	束論に関する補題	35
4.1.1	和関係, 共通関係	35
4.1.2	分配法則	44
4.1.3	原子性	46
4.2	Heyting 代数に関する補題	48
4.3	補関係に関する補題	55
4.4	Bool 代数に関する補題	59

5	Library <code>Relation_Properties</code>	62
5.1	関係計算の基本的な性質	62
5.2	<code>comp_inc_compat</code> と派生補題	71
5.3	逆関係に関する補題	74
5.4	合成に関する補題	79
5.5	単域と Tarski の定理	84
6	Library <code>Functions_Mappings</code>	89
6.1	全域性, 一価性, 写像に関する補題	89
6.2	全射, 単射に関する補題	99
7	Library <code>Dedekind</code>	108
7.1	Dedekind formula に関する補題	108
7.2	Dedekind formula と全関係	109
7.3	Dedekind formula と恒等関係	112
8	Library <code>Residual</code>	115
8.1	剰余合成関係の性質	115
8.1.1	基本的な性質	115
8.1.2	単調性と分配法則	118
8.1.3	剰余合成と関数	121
8.2	Galois 同値とその系	124
8.3	その他の性質	125
9	Library <code>Rationality</code>	131
9.1	有理性から導かれる系	131
10	Library <code>Conjugate</code>	133
10.1	共役性の定義	133
10.2	共役の例	134
11	Library <code>Domain</code>	143
11.1	定義域の定義	143
11.2	定義域の性質	143
11.2.1	基本的な性質	143
11.2.2	合成と定義域	146
11.2.3	その他の性質	150
11.2.4	矩形関係	155
12	Library <code>Sum_Product</code>	159
12.1	関係の直和	159
12.1.1	入射対, 関係直和の定義	159
12.1.2	関係直和の性質	161
12.1.3	分配法則	165

12.2	関係の直積	166
12.2.1	射影対, 関係直積の定義	166
12.2.2	関係直積の性質	168
12.2.3	鋭敏性	177
12.2.4	分配法則	183

Chapter 1

Library **Basic_Notations**

1.1 このライブラリについて

- このライブラリは河原康雄先生の“関係の理論 - Dedekind 圏概説 -”をもとに制作されている.
- 現状サポートしているのは,
 - 1.4 節大半, 1.5 - 1.6 節全部
 - 2.1 - 2.3 節全部, 2.4 - 2.5 節大半, 2.6 節全部, 2.7 節大半, 2.8 節有理性
 - 4.2 - 4.3 節全部, 4.4 - 4.5 節大半, 4.6 節命題 4.6.1

といったところである.

- 関係論で話を進めたい場合は, 下の行に `Require Export Basic_Notations_Rel.` を, 集合論で話を進めたい場合は, `Require Export Basic_Notations_Set.` を記述する.

Require Export *Basic_Notations_Rel.*

なお, 証明の書き方が悪いと, まれに“関係論では証明が通ったのに, 集合論では通らない”といったことも起こるようなので, ある程度注意しておく必要がある.

Chapter 2

Library `Basic_Notations_Rel`

`Require Export ssreflect eqtype bigop.`

`Require Export Logic.ClassicalFacts.`

`Axiom prop_extensionality_ok : prop_extensionality.`

2.1 定義

- A, B を `eqType` として, A から B への関係の型を $(\text{Rel } A B)$ と書き, $A \rightarrow B \rightarrow \text{Prop}$ として定義する. 本文中では型 $(\text{Rel } A B)$ を $A \rightarrow B$ と書く.
- 関係 $\alpha : A \rightarrow B$ の逆関係 $\alpha^\# : B \rightarrow A$ は $(\text{inverse } \alpha)$ で, Coq では $(\alpha \#)$ と記述する.
- 2 つの関係 $\alpha : A \rightarrow B, \beta : B \rightarrow C$ の合成関係 $\alpha\beta : A \rightarrow C$ は $(\text{composite } \alpha \beta)$ で, $(\alpha \cdot \beta)$ と記述する.
- 剰余合成関係 $\alpha \triangleright \beta : A \rightarrow C$ は $(\text{residual } \alpha \beta)$ で, $(\alpha \multimap \beta)$ と記述する.
- 恒等関係 $\text{id}_A : A \rightarrow A$ は $(\text{identity } A)$ で, $(\text{Id } A)$ と記述する.
- 空関係 $\phi_{AB} : A \rightarrow B$ は $(\text{empty } A B)$ で, $(\perp A B)$ と記述する.
- 全関係 $\nabla_{AB} : A \rightarrow B$ は $(\text{universal } A B)$ で, $(\top A B)$ と記述する.
- 2 つの関係 $\alpha : A \rightarrow B, \beta : A \rightarrow B$ の和関係 $\alpha \sqcup \beta : A \rightarrow B$ は $(\text{cup } \alpha \beta)$ で, $(\alpha \sqcup \beta)$ と記述する.
- 共通関係 $\alpha \sqcap \beta : A \rightarrow B$ は $(\text{cap } \alpha \beta)$ で, $(\alpha \sqcap \beta)$ と記述する.
- 相対擬補関係 $\alpha \Rightarrow \beta : A \rightarrow B$ は $(\text{rpc } \alpha \beta)$ で, $(\alpha \gg \beta)$ と記述する.
- 関係 $\alpha : A \rightarrow B$ の補関係 $\alpha^- : A \rightarrow B$ は $(\text{complement } \alpha)$ で, Coq では $(\alpha \sim)$ と記述する.

	数式	Coq	Notation
逆関係	$\alpha^\#$	(inverse α)	($\alpha \#$)
合成関係	$\alpha\beta$	(composite $\alpha\beta$)	($\alpha \cdot \beta$)
剰余合成関係	$\alpha \triangleright \beta$	(residual $\alpha\beta$)	($\alpha \ \beta$)
恒等関係	id_A	(identity A)	(Id A)
空関係	ϕ_{AB}	(empty $A B$)	($_ A B$)
全関係	∇_{AB}	(universal $A B$)	($_ A B$)
和関係	$\alpha \sqcup \beta$	(cup $\alpha\beta$)	($\alpha \ \beta$)
共通関係	$\alpha \sqcap \beta$	(cap $\alpha\beta$)	($\alpha \ \beta$)
相対擬補関係	$\alpha \Rightarrow \beta$	(rpc $\alpha \beta$)	($\alpha \gg \beta$)
補関係	α^-	(complement α)	($\alpha \ ^\wedge$)
差関係	$\alpha - \beta$	(difference $\alpha \beta$)	($\alpha \ -- \ \beta$)
添字付和関係	$\sqcup_{\lambda \in \Lambda} \alpha_\lambda$	(cupL L)	($_ _ L$)
添字付共通関係	$\sqcap_{\lambda \in \Lambda} \alpha_\lambda$	(capL L)	($_ _ L$)
条件付和関係	$\sqcup_{P(\lambda)} \alpha_\lambda$	(cupP $L P$)	($_ _ p \ P, L$)

Table 2.1: 関係の表記について

- 2つの関係 $\alpha : A \rightarrow B$, $\beta : A \rightarrow B$ の差関係 $\alpha - \beta : A \rightarrow B$ は (difference $\alpha \beta$) で, ($\alpha \ -- \ \beta$) と記述する.
- (capL) と (cupL) は添字付の共通関係と和関係であり, (cupP) は条件付の和関係である.
- また, 1点集合 $I = \{*\}$ は i と表記する.

表 2.1 に関係の表記についてまとめる.

Definition *Rel* ($A B : \text{eqType}$) := $A \rightarrow B \rightarrow \text{Prop}$.

Parameter *inverse* : ($\forall A B : \text{eqType}, \text{Rel } A B \rightarrow \text{Rel } B A$).

Notation "a #" := (*inverse* _ _ a) (at level 20).

Parameter *composite* : ($\forall A B C : \text{eqType}, \text{Rel } A B \rightarrow \text{Rel } B C \rightarrow \text{Rel } A C$).

Notation "a ' · ' b" := (*composite* _ _ _ a b) (at level 50).

Parameter *residual* : ($\forall A B C : \text{eqType}, \text{Rel } A B \rightarrow \text{Rel } B C \rightarrow \text{Rel } A C$).

Notation "a ' \ \ ' b" := (*residual* _ _ _ a b) (at level 50).

Parameter *identity* : ($\forall A : \text{eqType}, \text{Rel } A A$).

Notation "'Id'" := *identity*.

Parameter *empty* : ($\forall A B : \text{eqType}, \text{Rel } A B$).

Notation "' \ ' " := *empty*.

Parameter *universal* : ($\forall A B : \text{eqType}, \text{Rel } A B$).

Notation "' \ ' " := *universal*.

Parameter *include* : ($\forall A B : \text{eqType}, \text{Rel } A B \rightarrow \text{Rel } A B \rightarrow \text{Prop}$).

Notation "a' ' b" := (*include* _ _ a b) (at **level** 50).

Parameter *cup* : ($\forall A B : eqType, Rel A B \rightarrow Rel A B \rightarrow Rel A B$).

Notation "a' ' b" := (*cup* _ _ a b) (at **level** 50).

Parameter *cap* : ($\forall A B : eqType, Rel A B \rightarrow Rel A B \rightarrow Rel A B$).

Notation "a' ' b" := (*cap* _ _ a b) (at **level** 50).

Parameter *rpc* : ($\forall A B : eqType, Rel A B \rightarrow Rel A B \rightarrow Rel A B$).

Notation "a' »' b" := (*rpc* _ _ a b) (at **level** 50).

Definition *complement* {A B : eqType} (alpha : Rel A B) := alpha » A B.

Notation "a' ^'" := (*complement* a) (at **level** 20).

Definition *difference* {A B : eqType} (alpha beta : Rel A B) := alpha beta ^.

Notation "a - b" := (*difference* a b) (at **level** 50).

Parameter *cupL* : ($\forall A B L : eqType, (L \rightarrow Rel A B) \rightarrow Rel A B$).

Notation "' _' a" := (*cupL* _ _ _ a) (at **level** 50).

Parameter *capL* : ($\forall A B L : eqType, (L \rightarrow Rel A B) \rightarrow Rel A B$).

Notation "' _' a" := (*capL* _ _ _ a) (at **level** 50).

Parameter *cupP* : ($\forall A B L : eqType, (L \rightarrow Rel A B) \rightarrow (L \rightarrow Prop) \rightarrow Rel A B$).

Notation "' p' p' ,' a" := (*cupP* _ _ _ a p) (at **level** 50).

Notation "'i'" := *unit_eqType*.

2.2 関数の定義

$\alpha : A \rightarrow B$ に対し, 全域性 *total_r*, 一価性 *univalent_r*, 関数 *function_r*, 全射 *surjective_r*, 単射 *injective_r*, 全単射 *bijection_r* を以下のように定義する.

- *total_r* : $id_A \sqsubseteq \alpha \cdot \alpha^\#$
- *univalent_r* : $\alpha^\# \cdot \alpha \sqsubseteq id_B$
- *function_r* : $id_A \sqsubseteq \alpha \cdot \alpha^\# \wedge \alpha^\# \cdot \alpha \sqsubseteq id_B$
- *surjection_r* : $id_A \sqsubseteq \alpha \cdot \alpha^\# \wedge \alpha^\# \cdot \alpha = id_B$
- *injection_r* : $id_A = \alpha \cdot \alpha^\# \wedge \alpha^\# \cdot \alpha \sqsubseteq id_B$
- *bijection_r* : $id_A = \alpha \cdot \alpha^\# \wedge \alpha^\# \cdot \alpha = id_B$

Definition *total_r* {A B : eqType} (alpha : Rel A B) := (*Id* A) (alpha · alpha #).

Definition *univalent_r* {A B : eqType} (alpha : Rel A B) := (alpha # · alpha) (*Id* B).

Definition *function_r* {A B : eqType} (alpha : Rel A B)

:= (*total_r* alpha) ∧ (*univalent_r* alpha).

Definition *surjection_r* {A B : eqType} (alpha : Rel A B)

:= (*function_r* alpha) ∧ (*total_r* (alpha #)).

Definition *injection_r* $\{A\ B : \text{eqType}\}$ ($\alpha : \text{Rel } A\ B$)
 $:= (\text{function_r } \alpha) \wedge (\text{univalent_r } (\alpha \#)).$

Definition *bijection_r* $\{A\ B : \text{eqType}\}$ ($\alpha : \text{Rel } A\ B$)
 $:= (\text{function_r } \alpha) \wedge (\text{total_r } (\alpha \#)) \wedge (\text{univalent_r } (\alpha \#)).$

2.3 関係の公理

今後の諸定理の証明は、原則以下の公理群、およびそれらから導かれる補題のみを用いて行っていくことにする。

2.3.1 Dedekind 圏の公理

Axiom 1 (comp_id_l, comp_id_r) *Let* $\alpha : A \rightarrow B$. *Then,*

$$\text{id}_A \cdot \alpha = \alpha \cdot \text{id}_B = \alpha.$$

Definition *axiom1a* $:= \forall (A\ B : \text{eqType})(\alpha : \text{Rel } A\ B), \text{Id } A \cdot \alpha = \alpha.$
Axiom *comp_id_l* : *axiom1a*.

Definition *axiom1b* $:= \forall (A\ B : \text{eqType})(\alpha : \text{Rel } A\ B), \alpha \cdot \text{Id } B = \alpha.$
Axiom *comp_id_r* : *axiom1b*.

Axiom 2 (comp_assoc) *Let* $\alpha : A \rightarrow B$, $\beta : B \rightarrow C$, *and* $\gamma : C \rightarrow D$. *Then,*

$$(\alpha \cdot \beta) \cdot \gamma = \alpha \cdot (\beta \cdot \gamma).$$

Definition *axiom2* $:=$
 $\forall (A\ B\ C\ D : \text{eqType})(\alpha : \text{Rel } A\ B)(\beta : \text{Rel } B\ C)(\gamma : \text{Rel } C\ D),$
 $(\alpha \cdot \beta) \cdot \gamma = \alpha \cdot (\beta \cdot \gamma).$
Axiom *comp_assoc* : *axiom2*.

Axiom 3 (inc_refl) *Let* $\alpha : A \rightarrow B$. *Then,*

$$\alpha \sqsubseteq \alpha.$$

Definition *axiom3* $:= \forall (A\ B : \text{eqType})(\alpha : \text{Rel } A\ B), \alpha \sqsubseteq \alpha.$
Axiom *inc_refl* : *axiom3*.

Axiom 4 (inc_trans) *Let* $\alpha, \beta, \gamma : A \rightarrow B$. *Then,*

$$\alpha \sqsubseteq \beta \wedge \beta \sqsubseteq \gamma \Rightarrow \alpha \sqsubseteq \gamma.$$

Definition *axiom4* $:= \forall (A\ B : \text{eqType})(\alpha\ \beta\ \gamma : \text{Rel } A\ B),$

$\alpha \beta \rightarrow \beta \quad \gamma \rightarrow \alpha \quad \gamma$.
Axiom *inc_trans* : *axiom4*.

Axiom 5 (inc_antisym) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$\alpha \sqsubseteq \beta \wedge \beta \sqsubseteq \alpha \Rightarrow \alpha = \beta.$$

Definition *axiom5* := $\forall (A B : eqType)(\alpha \beta : Rel A B)$,
 $\alpha \beta \rightarrow \beta \quad \alpha \rightarrow \alpha = \beta$.
Axiom *inc_antisym* : *axiom5*.

Axiom 6 (inc_empty_alpha) *Let $\alpha : A \rightarrow B$. Then,*

$$\phi_{AB} \sqsubseteq \alpha.$$

Definition *axiom6* := $\forall (A B : eqType)(\alpha : Rel A B), \quad A B \quad \alpha$.
Axiom *inc_empty_alpha* : *axiom6*.

Axiom 7 (inc_alpha_universal) *Let $\alpha : A \rightarrow B$. Then,*

$$\alpha \sqsubseteq \nabla_{AB}.$$

Definition *axiom7* := $\forall (A B : eqType)(\alpha : Rel A B), \alpha \quad A B$.
Axiom *inc_alpha_universal* : *axiom7*.

Axiom 8 (inc_cap) *Let $\alpha, \beta, \gamma : A \rightarrow B$. Then,*

$$\alpha \sqsubseteq (\beta \sqcap \gamma) \Leftrightarrow \alpha \sqsubseteq \beta \wedge \alpha \sqsubseteq \gamma.$$

Definition *axiom8* := $\forall (A B : eqType)(\alpha \beta \gamma : Rel A B)$,
 $\alpha \quad (\beta \gamma) \Leftrightarrow (\alpha \quad \beta) \wedge (\alpha \quad \gamma)$.
Axiom *inc_cap* : *axiom8*.

Axiom 9 (inc_cup) *Let $\alpha, \beta, \gamma : A \rightarrow B$. Then,*

$$(\beta \sqcup \gamma) \sqsubseteq \alpha \Leftrightarrow \beta \sqsubseteq \alpha \wedge \gamma \sqsubseteq \alpha.$$

Definition *axiom9* := $\forall (A B : eqType)(\alpha \beta \gamma : Rel A B)$,
 $(\beta \gamma) \quad \alpha \Leftrightarrow (\beta \quad \alpha) \wedge (\gamma \quad \alpha)$.
Axiom *inc_cup* : *axiom9*.

Axiom 10 (inc_capL) Let $\alpha, \beta_\lambda : A \rightarrow B$. Then,

$$\alpha \sqsubseteq (\prod_{\lambda \in \Lambda} \beta_\lambda) \Leftrightarrow \forall \lambda \in \Lambda, \alpha \sqsubseteq \beta_\lambda.$$

Definition $axiom10 := \forall (A B L : eqType)(alpha : Rel A B)(beta_L : L \rightarrow Rel A B),$
 $alpha \sqsubseteq (\prod_{l \in L} beta_L l) \Leftrightarrow \forall l : L, alpha \sqsubseteq beta_L l.$

Axiom $inc_capL : axiom10.$

Axiom 11 (inc_cupL) Let $\alpha, \beta_\lambda : A \rightarrow B$. Then,

$$(\bigsqcup_{\lambda \in \Lambda} \beta_\lambda) \sqsubseteq \alpha \Leftrightarrow \forall \lambda \in \Lambda, \beta_\lambda \sqsubseteq \alpha.$$

Definition $axiom11 := \forall (A B L : eqType)(alpha : Rel A B)(beta_L : L \rightarrow Rel A B),$
 $(\bigsqcup_{l \in L} beta_L l) \sqsubseteq alpha \Leftrightarrow \forall l : L, beta_L l \sqsubseteq alpha.$

Axiom $inc_cupL : axiom11.$

Axiom 12 (inc_rpc) Let $\alpha, \beta, \gamma : A \rightarrow B$. Then,

$$\alpha \sqsubseteq (\beta \Rightarrow \gamma) \Leftrightarrow (\alpha \sqcap \beta) \sqsubseteq \gamma.$$

Definition $axiom12 := \forall (A B : eqType)(alpha beta gamma : Rel A B),$
 $alpha \sqsubseteq (beta \gg gamma) \Leftrightarrow (alpha \sqcap beta) \sqsubseteq gamma.$

Axiom $inc_rpc : axiom12.$

Axiom 13 (inv_invol) Let $\alpha : A \rightarrow B$. Then,

$$(\alpha^\#)^\# = \alpha.$$

Definition $axiom13 := \forall (A B : eqType)(alpha : Rel A B), (alpha \#) \# = alpha.$

Axiom $inv_invol : axiom13.$

Axiom 14 (comp_inv) Let $\alpha : A \rightarrow B$ and $\beta : B \rightarrow C$. Then,

$$(\alpha \cdot \beta)^\# = \beta^\# \cdot \alpha^\#.$$

Definition $axiom14 := \forall (A B C : eqType)(alpha : Rel A B)(beta : Rel B C),$
 $(alpha \cdot beta) \# = (beta \# \cdot alpha \#).$

Axiom $comp_inv : axiom14.$

Axiom 15 (inc_inv) Let $\alpha, \beta : A \rightarrow B$. Then,

$$\alpha \sqsubseteq \beta \Rightarrow \alpha^\# \sqsubseteq \beta^\#.$$

Definition *axiom15* :=

$\forall (A\ B : eqType)(\alpha\ \beta : Rel\ A\ B), \alpha\ \beta \rightarrow \alpha^\# \sqsubseteq \beta^\#.$

Axiom *inc_inv* : *axiom15*.

Axiom 16 (dedekind) Let $\alpha : A \rightarrow B$, $\beta : B \rightarrow C$, and $\gamma : A \rightarrow C$. Then,

$$(\alpha \cdot \beta) \sqcap \gamma \sqsubseteq (\alpha \sqcap (\gamma \cdot \beta^\#)) \cdot (\beta \sqcap (\alpha^\# \cdot \gamma)).$$

Definition *axiom16* :=

$\forall (A\ B\ C : eqType)(\alpha : Rel\ A\ B)(\beta : Rel\ B\ C)(\gamma : Rel\ A\ C),$
 $((\alpha \cdot \beta) \sqcap \gamma) \sqsubseteq ((\alpha \sqcap (\gamma \cdot \beta^\#)) \cdot (\beta \sqcap (\alpha^\# \cdot \gamma))).$

Axiom *dedekind* : *axiom16*.

Axiom 17 (inc_residual) Let $\alpha : A \rightarrow B$, $\beta : B \rightarrow C$, and $\gamma : A \rightarrow C$. Then,

$$\gamma \sqsubseteq (\alpha \triangleright \beta) \Leftrightarrow \alpha^\# \cdot \gamma \sqsubseteq \beta.$$

Definition *axiom17* :=

$\forall (A\ B\ C : eqType)(\alpha : Rel\ A\ B)(\beta : Rel\ B\ C)(\gamma : Rel\ A\ C),$
 $\gamma \sqsubseteq (\alpha \triangleright \beta) \Leftrightarrow (\alpha^\# \cdot \gamma) \sqsubseteq \beta.$

Axiom *inc_residual* : *axiom17*.

2.3.2 排中律

Dedekind 圏の公理のほかに、以下の“排中律”を仮定すれば、与えられる圏は Schröder 圏となり、Bool 代数の性質も満たされる。

Axiom 18 (complement_classic) Let $\alpha : A \rightarrow B$. Then,

$$\alpha \sqcup \alpha^- = \nabla_{AB}$$

Definition *axiom18* := $\forall (A\ B : eqType)(\alpha : Rel\ A\ B),$
 $\alpha \sqcup \alpha^- = \nabla_{AB}.$

Axiom *complement_classic* : *axiom18*.

2.3.3 単域

1 点集合 I が単域となるための条件は

$$\phi_{II} \neq id_I \wedge id_I = \nabla_{II} \wedge \forall X, \nabla_{XI} \cdot \nabla_{IX} = \nabla_{XX}$$

だが, Rel の定義から左 2 つは証明できるため, 右の式だけ仮定する.

Axiom 19 (unit_universal)

$$\nabla_{AI} \cdot \nabla_{IA} = \nabla_{AA}$$

Definition $axiom19 := \forall (A : eqType), \quad A \text{ i } \cdot \quad i A = \quad A A.$

Axiom $unit_universal : axiom19.$

2.3.4 点公理

まずは Dedekind 圏にない “条件付和関係” を定義する公理から.

Axiom 20 (inc_cupP) *Let $\alpha, \beta_\lambda : A \rightarrow B$ and $P : predicate$. Then,*

$$(\sqcup_{P(\lambda)} \beta_\lambda) \sqsubseteq \alpha \Leftrightarrow (\forall \lambda \in \Lambda, P(\lambda) \Rightarrow \beta_\lambda \sqsubseteq \alpha).$$

Definition $axiom20 :=$

$\forall (A B L : eqType)(\alpha : Rel A B)(\beta_{\beta_L} : L \rightarrow Rel A B)(P : L \rightarrow Prop),$
 $(\quad p P, \beta_{\beta_L}) \quad \alpha \Leftrightarrow \forall l : L, P l \rightarrow \beta_{\beta_L} l \quad \alpha.$

Axiom $inc_cupP : axiom20.$

この “弱選択公理” を仮定すれば, 排中律と単域の存在 (厳密には全域性公理) を利用して点公理を導出できる.

Axiom 21 (weak_axiom_of_choice) *Let $\alpha : I \rightarrow A$ be a total relation. Then,*

$$\exists \beta : I \rightarrow A, \beta \sqsubseteq \alpha.$$

Definition $axiom21 := \forall (A : eqType)(\alpha : Rel i A),$

$total_r \alpha \rightarrow \exists \beta : Rel i A, function_r \beta \wedge \beta \quad \alpha.$

Axiom $weak_axiom_of_choice : axiom21.$

2.3.5 関係の有理性

集合論では色々インポートしながら頑張って証明したので、できればそちらもご参照ください。

Axiom 22 (rationality) *Let $\alpha : A \rightarrow B$. Then,*

$$\exists R, \exists f : R \rightarrow A, \exists g : R \rightarrow B, \alpha = f^\# \cdot g \wedge f \cdot f^\# \sqcap g \cdot g^\# = id_R.$$

Definition *axiom22* := $\forall (A B : eqType)(alpha : Rel A B),$
 $\exists (R : eqType)(f : Rel R A)(g : Rel R B),$
 $function_r f \wedge function_r g \wedge alpha = f^\# \cdot g \wedge ((f \cdot f^\#) \sqcap (g \cdot g^\#)) = Id R.$
Axiom *rationality* : *axiom22*.

2.3.6 直和と直積

任意の直和に対して、入射対が存在することを仮定する。

Axiom 23 (pair_of_inclusions) $\exists j : A \rightarrow A + B, \exists k : B \rightarrow A + B,$

$$j \cdot j^\# = id_A \wedge k \cdot k^\# = id_B \wedge j \cdot k^\# = \phi_{AB} \wedge j^\# \cdot j \sqcup k^\# \cdot k = id_{A+B}.$$

Definition *axiom23* :=
 $\forall (A B : eqType), \exists (j : Rel A (sum_eqType A B))(k : Rel B (sum_eqType A B)),$
 $j \cdot j^\# = Id A \wedge k \cdot k^\# = Id B \wedge j \cdot k^\# = \phi_{AB} \wedge$
 $(j^\# \cdot j) \sqcup (k^\# \cdot k) = Id (sum_eqType A B).$

Axiom *pair_of_inclusions* : *axiom23*.

任意の直積に対して、射影対が存在することを仮定する。

Axiom 24 (pair_of_projections) $\exists p : A \times B \rightarrow A, \exists q : A \times B \rightarrow B,$

$$p^\# \cdot q = \nabla_{AB} \wedge p \cdot p^\# \sqcap q \cdot q^\# = id_{A \times B}.$$

Definition *axiom24* :=
 $\forall (A B : eqType), \exists (p : Rel (prod_eqType A B) A)(q : Rel (prod_eqType A B) B),$
 $p^\# \cdot q = \nabla_{AB} \wedge (p \cdot p^\#) \sqcap (q \cdot q^\#) = Id (prod_eqType A B) \wedge univalent_r p$
 $\wedge univalent_r q.$

Axiom *pair_of_projections* : *axiom24*.

Chapter 3

Library `Basic_Notations_Set`

```
Require Export ssreflect eqtype bigop.
Require Export Logic.ClassicalFacts.
Require Import Logic.FunctionalExtensionality.
Require Import Logic.Classical_Prop.
Require Import Logic.IndefiniteDescription.
Require Import Logic.ProofIrrelevance.

Axiom prop_extensionality_ok : prop_extensionality.
```

3.1 定義

この章では、関係を集合論的に定義した場合の定義、およびその定義で諸公理が成立することを示す。公理名や記号などは `Basic_Notations` と同じものを使用するため、`Basic_Lemms` 以降ではその代わりにこのライブラリをインポートすることもできる。

```
Definition Rel (A B : eqType) := A → B → Prop.

Definition inverse {A B : eqType} (alpha : Rel A B) : Rel B A
:= (fun (b : B)(a : A) => alpha a b).

Notation "a #" := (inverse a) (at level 20).

Definition composite {A B C : eqType} (alpha : Rel A B) (beta : Rel B C) : Rel A C
:= (fun (a : A)(c : C) => ∃ b : B, alpha a b ∧ beta b c).

Notation "a ' · ' b" := (composite a b) (at level 50).

Definition residual {A B C : eqType} (alpha : Rel A B) (beta : Rel B C) : Rel A C
:= (fun (a : A)(c : C) => ∀ b : B, alpha a b → beta b c).

Notation "a ' ' b" := (residual a b) (at level 50).

Definition identity (A : eqType) : Rel A A := (fun a a0 : A => a = a0).

Notation "'Id'" := identity.

Definition empty (A B : eqType) : Rel A B := (fun (a : A)(b : B) => False).

Notation "' ' " := empty.

Definition universal (A B : eqType) : Rel A B := (fun (a : A)(b : B) => True).
```

Notation " \bigvee " := *universal*.

Definition *include* $\{A B : eqType\}$ ($\alpha \beta : Rel A B$) : **Prop**
 $:= (\forall (a : A)(b : B), \alpha a b \rightarrow \beta a b).$

Notation " $a \bigvee b$ " := (*include* $a b$) (at level 50).

Definition *cup* $\{A B : eqType\}$ ($\alpha \beta : Rel A B$) : $Rel A B$
 $:= (\text{fun } (a : A)(b : B) \Rightarrow \alpha a b \vee \beta a b).$

Notation " $a \bigcup b$ " := (*cup* $a b$) (at level 50).

Definition *cap* $\{A B : eqType\}$ ($\alpha \beta : Rel A B$) : $Rel A B$
 $:= (\text{fun } (a : A)(b : B) \Rightarrow \alpha a b \wedge \beta a b).$

Notation " $a \bigcap b$ " := (*cap* $a b$) (at level 50).

Definition *rpc* $\{A B : eqType\}$ ($\alpha \beta : Rel A B$) : $Rel A B$
 $:= (\text{fun } (a : A)(b : B) \Rightarrow \alpha a b \rightarrow \beta a b).$

Notation " $a \gg b$ " := (*rpc* $a b$) (at level 50).

Definition *complement* $\{A B : eqType\}$ ($\alpha : Rel A B$) := $\alpha \gg A B$.

Notation " $a \wedge$ " := (*complement* a) (at level 20).

Definition *difference* $\{A B : eqType\}$ ($\alpha \beta : Rel A B$) := $\alpha \wedge \beta \wedge$.

Notation " $a - b$ " := (*difference* $a b$) (at level 50).

Definition *cupL* $\{A B L : eqType\}$ ($\alpha_L : L \rightarrow Rel A B$) : $Rel A B$
 $:= (\text{fun } (a : A)(b : B) \Rightarrow \exists l : L, \alpha_L l a b).$

Notation " $\bigcup_l a$ " := (*cupL* a) (at level 50).

Definition *capL* $\{A B L : eqType\}$ ($\alpha_L : L \rightarrow Rel A B$) : $Rel A B$
 $:= (\text{fun } (a : A)(b : B) \Rightarrow \forall l : L, \alpha_L l a b).$

Notation " $\bigcap_l a$ " := (*capL* a) (at level 50).

Definition *cupP* $\{A B L : eqType\}$ ($\alpha_L : L \rightarrow Rel A B$) ($P : L \rightarrow \text{Prop}$) : $Rel A B$
 $:= (\text{fun } (a : A)(b : B) \Rightarrow \exists l : L, P l \wedge \alpha_L l a b).$

Notation " $\bigcup_{p \in P} a$ " := (*cupP* $a p$) (at level 50).

Notation " i " := *unit_eqType*.

3.2 関数の定義

$\alpha : A \rightarrow B$ に対し, 全域性 `total_r`, 一価性 `univalent_r`, 関数 `function_r`, 全射 `surjective_r`, 単射 `injective_r`, 全単射 `bijection_r` を以下のように定義する.

- `total_r` : $id_A \sqsubseteq \alpha \cdot \alpha^\#$
- `univalent_r` : $\alpha^\# \cdot \alpha \sqsubseteq id_B$
- `function_r` : $id_A \sqsubseteq \alpha \cdot \alpha^\# \wedge \alpha^\# \cdot \alpha \sqsubseteq id_B$
- `surjection_r` : $id_A \sqsubseteq \alpha \cdot \alpha^\# \wedge \alpha^\# \cdot \alpha = id_B$
- `injection_r` : $id_A = \alpha \cdot \alpha^\# \wedge \alpha^\# \cdot \alpha \sqsubseteq id_B$
- `bijection_r` : $id_A = \alpha \cdot \alpha^\# \wedge \alpha^\# \cdot \alpha = id_B$

Definition `total_r` $\{A\ B : eqType\}$ ($\alpha : Rel\ A\ B$) := ($Id\ A$) ($\alpha \cdot \alpha^\#$).

Definition `univalent_r` $\{A\ B : eqType\}$ ($\alpha : Rel\ A\ B$) := ($\alpha^\# \cdot \alpha$) ($Id\ B$).

Definition `function_r` $\{A\ B : eqType\}$ ($\alpha : Rel\ A\ B$)
:= (`total_r` α) \wedge (`univalent_r` α).

Definition `surjection_r` $\{A\ B : eqType\}$ ($\alpha : Rel\ A\ B$)
:= (`function_r` α) \wedge (`total_r` ($\alpha^\#$)).

Definition `injection_r` $\{A\ B : eqType\}$ ($\alpha : Rel\ A\ B$)
:= (`function_r` α) \wedge (`univalent_r` ($\alpha^\#$)).

Definition `bijection_r` $\{A\ B : eqType\}$ ($\alpha : Rel\ A\ B$)
:= (`function_r` α) \wedge (`total_r` ($\alpha^\#$)) \wedge (`univalent_r` ($\alpha^\#$)).

3.3 関係の公理

今後の諸定理の証明は, 原則以下の公理群, およびそれらから導かれる補題のみを用いて行っていくことにする.

3.3.1 Dedekind 圏の公理

Lemma 1 (`comp_id_l`, `comp_id_r`) *Let* $\alpha : A \rightarrow B$. *Then,*

$$id_A \cdot \alpha = \alpha \cdot id_B = \alpha.$$

Definition `axiom1a` := $\forall (A\ B : eqType)(\alpha : Rel\ A\ B), Id\ A \cdot \alpha = \alpha$.

Lemma `comp_id_l` : `axiom1a`.

Proof.

`move` $\Rightarrow A\ B\ \alpha$.

```

apply functional_extensionality.
move => a.
apply functional_extensionality.
move => b.
apply prop_extensionality_ok.
split.
elim => a0.
elim => H H0.
rewrite H.
apply H0.
move => H.
exists a.
split.
by [].
apply H.
Qed.

```

Definition *axiom1b* := $\forall (A\ B : \text{eqType})(\alpha : \text{Rel } A\ B), \alpha \cdot \text{Id } B = \alpha$.

Lemma *comp_id_r* : *axiom1b*.

Proof.

```

move => A B alpha.
apply functional_extensionality.
move => a.
apply functional_extensionality.
move => b.
apply prop_extensionality_ok.
split.
elim => b0.
elim => H H0.
rewrite -H0.
apply H.
move => H.
exists b.
split.
apply H.
by [].
Qed.

```

Lemma 2 (comp_assoc) *Let $\alpha : A \rightarrow B$, $\beta : B \rightarrow C$, and $\gamma : C \rightarrow D$. Then,*

$$(\alpha \cdot \beta) \cdot \gamma = \alpha \cdot (\beta \cdot \gamma).$$

Definition *axiom2* :=

$\forall (A\ B\ C\ D : eqType)(\alpha : Rel\ A\ B)(\beta : Rel\ B\ C)(\gamma : Rel\ C\ D),$
 $(\alpha \cdot \beta) \cdot \gamma = \alpha \cdot (\beta \cdot \gamma).$

Lemma *comp_assoc* : *axiom2*.

Proof.

move $\Rightarrow A\ B\ C\ D\ \alpha\ \beta\ \gamma.$

apply *functional_extensionality*.

move $\Rightarrow a.$

apply *functional_extensionality*.

move $\Rightarrow d.$

apply *prop_extensionality_ok*.

split.

elim $\Rightarrow c.$

elim $\Rightarrow H\ H0.$

elim $H \Rightarrow b\ H1.$

$\exists\ b.$

split.

apply $H1.$

$\exists\ c.$

split.

apply $H1.$

apply $H0.$

elim $\Rightarrow b.$

elim $\Rightarrow H.$

elim $\Rightarrow c\ H0.$

$\exists\ c.$

split.

$\exists\ b.$

split.

apply $H.$

apply $H0.$

apply $H0.$

Qed.

Lemma 3 (inc_refl) *Let $\alpha : A \rightarrow B$. Then,*

$$\alpha \sqsubseteq \alpha.$$

Definition *axiom3* := $\forall (A\ B : eqType)(\alpha : Rel\ A\ B),\ \alpha \sqsubseteq \alpha.$

Lemma *inc_refl* : *axiom3*.

Proof.

by [rewrite /*axiom3*/include].

Qed.

Lemma 4 (inc_trans) *Let $\alpha, \beta, \gamma : A \rightarrow B$. Then,*

$$\alpha \sqsubseteq \beta \wedge \beta \sqsubseteq \gamma \Rightarrow \alpha \sqsubseteq \gamma.$$

Definition *axiom4* := $\forall (A B : eqType)(\alpha \beta \gamma : Rel A B),$
 $\alpha \beta \rightarrow \beta \gamma \rightarrow \alpha \gamma$.

Lemma *inc_trans* : *axiom4*.

Proof.

move $\Rightarrow A B \alpha \beta \gamma H H0 a b H1$.

apply (*H0* _ _ (*H* _ _ *H1*)).

Qed.

Lemma 5 (inc_antisym) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$\alpha \sqsubseteq \beta \wedge \beta \sqsubseteq \alpha \Rightarrow \alpha = \beta.$$

Definition *axiom5* := $\forall (A B : eqType)(\alpha \beta : Rel A B),$
 $\alpha \beta \rightarrow \beta \alpha \rightarrow \alpha = \beta$.

Lemma *inc_antisym* : *axiom5*.

Proof.

move $\Rightarrow A B \alpha \beta H H0$.

apply *functional_extensionality*.

move $\Rightarrow a$.

apply *functional_extensionality*.

move $\Rightarrow b$.

apply *prop_extensionality_ok*.

split.

apply *H*.

apply *H0*.

Qed.

Lemma 6 (inc_empty_alpha) *Let $\alpha : A \rightarrow B$. Then,*

$$\phi_{AB} \sqsubseteq \alpha.$$

Definition *axiom6* := $\forall (A B : eqType)(\alpha : Rel A B), \quad A B \quad \alpha$.

Lemma *inc_empty_alpha* : *axiom6*.

Proof.

move $\Rightarrow A B \alpha a b$.

apply *False_ind*.

Qed.

Lemma 7 (inc_alpha_universal) *Let $\alpha : A \rightarrow B$. Then,*

$$\alpha \sqsubseteq \nabla_{AB}.$$

Definition *axiom7* := $\forall (A B : eqType)(\alpha : Rel A B), \alpha \sqsubseteq \nabla_{AB}$.

Lemma *inc_alpha_universal* : *axiom7*.

Proof.

move $\Rightarrow A B \alpha a b H$.

apply *I*.

Qed.

Lemma 8 (inc_cap) *Let $\alpha, \beta, \gamma : A \rightarrow B$. Then,*

$$\alpha \sqsubseteq (\beta \sqcap \gamma) \Leftrightarrow \alpha \sqsubseteq \beta \wedge \alpha \sqsubseteq \gamma.$$

Definition *axiom8* := $\forall (A B : eqType)(\alpha \beta \gamma : Rel A B), (\alpha \sqsubseteq (\beta \sqcap \gamma) \Leftrightarrow (\alpha \sqsubseteq \beta \wedge \alpha \sqsubseteq \gamma))$.

Lemma *inc_cap* : *axiom8*.

Proof.

move $\Rightarrow A B \alpha \beta \gamma$.

split; move $\Rightarrow H$.

split.

move $\Rightarrow a b H0$.

apply (*H a b H0*).

move $\Rightarrow a b H0$.

apply (*H a b H0*).

move $\Rightarrow a b H0$.

split.

apply *H*.

apply *H0*.

apply *H*.

apply *H0*.

Qed.

Lemma 9 (inc_cup) *Let $\alpha, \beta, \gamma : A \rightarrow B$. Then,*

$$(\beta \sqcup \gamma) \sqsubseteq \alpha \Leftrightarrow \beta \sqsubseteq \alpha \wedge \gamma \sqsubseteq \alpha.$$

Definition *axiom9* := $\forall (A B : eqType)(\alpha \beta \gamma : Rel A B), ((\beta \sqcup \gamma) \sqsubseteq \alpha \Leftrightarrow (\beta \sqsubseteq \alpha \wedge \gamma \sqsubseteq \alpha))$.

Lemma *inc_cup* : *axiom9*.

Proof.

move $\Rightarrow A B \alpha \beta \gamma$.

split; move $\Rightarrow H$.
 split.
 move $\Rightarrow a\ b\ H0$.
 apply H .
 left.
 apply $H0$.
 move $\Rightarrow a\ b\ H0$.
 apply H .
 right.
 apply $H0$.
 move $\Rightarrow a\ b$.
 case; apply H .
 Qed.

Lemma 10 (inc_capL) *Let $\alpha, \beta_\lambda : A \rightarrow B$. Then,*

$$\alpha \sqsubseteq (\prod_{\lambda \in \Lambda} \beta_\lambda) \Leftrightarrow \forall \lambda \in \Lambda, \alpha \sqsubseteq \beta_\lambda.$$

Definition $axiom10 := \forall (A\ B\ L : eqType)(alpha : Rel\ A\ B)(beta_L : L \rightarrow Rel\ A\ B),$
 $(_ _ beta_L) \Leftrightarrow \forall l : L, alpha _ _ beta_L\ l.$

Lemma $inc_capL : axiom10$.

Proof.

move $\Rightarrow A\ B\ L\ alpha\ beta_L$.
 split; move $\Rightarrow H$.
 move $\Rightarrow l\ a\ b\ H0$.
 apply $(H _ _ H0)$.
 move $\Rightarrow a\ b\ H0\ l$.
 apply $(H _ _ _ H0)$.
 Qed.

Lemma 11 (inc_cupL) *Let $\alpha, \beta_\lambda : A \rightarrow B$. Then,*

$$(\sqcup_{\lambda \in \Lambda} \beta_\lambda) \sqsubseteq \alpha \Leftrightarrow \forall \lambda \in \Lambda, \beta_\lambda \sqsubseteq \alpha.$$

Definition $axiom11 := \forall (A\ B\ L : eqType)(alpha : Rel\ A\ B)(beta_L : L \rightarrow Rel\ A\ B),$
 $(_ _ beta_L) \sqsubseteq alpha \Leftrightarrow \forall l : L, beta_L\ l \sqsubseteq alpha.$

Lemma $inc_cupL : axiom11$.

Proof.

move $\Rightarrow A\ B\ L\ alpha\ beta_L$.
 split; move $\Rightarrow H$.
 move $\Rightarrow l\ a\ b\ H0$.
 apply H .
 $\exists l$.

apply $H0$.
 move $\Rightarrow a\ b$.
 elim $\Rightarrow l\ H0$.
 apply $(H\ _ _ _ H0)$.
 Qed.

Lemma 12 (inc_rpc) *Let $\alpha, \beta, \gamma : A \rightarrow B$. Then,*

$$\alpha \sqsubseteq (\beta \Rightarrow \gamma) \Leftrightarrow (\alpha \sqcap \beta) \sqsubseteq \gamma.$$

Definition $axiom12 := \forall (A\ B : eqType)(alpha\ beta\ gamma : Rel\ A\ B),$
 $alpha\ (beta \gg gamma) \leftrightarrow (alpha\ beta)\ gamma.$

Lemma $inc_rpc : axiom12$.

Proof.

move $\Rightarrow A\ B\ alpha\ beta\ gamma$.
 split; move $\Rightarrow H$.
 move $\Rightarrow a\ b$.
 elim $\Rightarrow H0\ H1$.
 apply $(H\ _ _ H0\ H1)$.
 move $\Rightarrow a\ b\ H0\ H1$.
 apply H .
 split.
 apply $H0$.
 apply $H1$.
 Qed.

Lemma 13 (inv_invol) *Let $\alpha : A \rightarrow B$. Then,*

$$(\alpha^\#)^\# = \alpha.$$

Definition $axiom13 := \forall (A\ B : eqType)(alpha : Rel\ A\ B), (alpha\ \#) \# = alpha.$

Lemma $inv_invol : axiom13$.

Proof.

by [move $\Rightarrow A\ B\ alpha$].

Qed.

Lemma 14 (comp_inv) *Let $\alpha : A \rightarrow B$ and $\beta : B \rightarrow C$. Then,*

$$(\alpha \cdot \beta)^\# = \beta^\# \cdot \alpha^\#.$$

Definition $axiom14 := \forall (A\ B\ C : eqType)(alpha : Rel\ A\ B)(beta : Rel\ B\ C),$
 $(alpha \cdot beta) \# = (beta \# \cdot alpha \#).$

Lemma $comp_inv : axiom14$.

Proof.

move $\Rightarrow A\ B\ C\ \text{alpha}\ \text{beta}$.

apply *functional_extensionality*.

move $\Rightarrow c$.

apply *functional_extensionality*.

move $\Rightarrow a$.

apply *prop_extensionality_ok*.

split; elim $\Rightarrow b$.

elim $\Rightarrow H\ H0$.

$\exists\ b$.

split.

apply *H0*.

apply *H*.

elim $\Rightarrow H\ H0$.

$\exists\ b$.

split.

apply *H0*.

apply *H*.

Qed.

Lemma 15 (inc_inv) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$\alpha \sqsubseteq \beta \Rightarrow \alpha^\# \sqsubseteq \beta^\#.$$

Definition *axiom15* :=

$\forall (A\ B : \text{eqType})(\text{alpha}\ \text{beta} : \text{Rel}\ A\ B),\ \text{alpha}\ \text{beta} \rightarrow \text{alpha}\ \# \ \text{beta}\ \#.$

Lemma *inc_inv* : *axiom15*.

Proof.

move $\Rightarrow A\ B\ \text{alpha}\ \text{beta}\ H\ b\ a\ H0$.

apply (*H* _ _ *H0*).

Qed.

Lemma 16 (dedekind) *Let $\alpha : A \rightarrow B$, $\beta : B \rightarrow C$, and $\gamma : A \rightarrow C$. Then,*

$$(\alpha \cdot \beta) \sqcap \gamma \sqsubseteq (\alpha \sqcap (\gamma \cdot \beta^\#)) \cdot (\beta \sqcap (\alpha^\# \cdot \gamma)).$$

Definition *axiom16* :=

$\forall (A\ B\ C : \text{eqType})(\text{alpha} : \text{Rel}\ A\ B)(\text{beta} : \text{Rel}\ B\ C)(\text{gamma} : \text{Rel}\ A\ C),$
 $((\text{alpha} \cdot \text{beta}) \sqcap \text{gamma})$
 $((\text{alpha} \sqcap (\text{gamma} \cdot \text{beta}^\#)) \cdot (\text{beta} \sqcap (\text{alpha}^\# \cdot \text{gamma}))).$

Lemma *dedekind* : *axiom16*.

Proof.

move $\Rightarrow A\ B\ C\ \text{alpha}\ \text{beta}\ \text{gamma}\ a\ c$.


```

elim.
elim  $\Rightarrow b$ .
move  $\Rightarrow H\ H0$ .
 $\exists b$ .
repeat split.
apply  $H$ .
 $\exists c$ .
split.
apply  $H0$ .
apply  $H$ .
apply  $H$ .
 $\exists a$ .
split.
apply  $H$ .
apply  $H0$ .
Qed.

```

Lemma 17 (inc_residual) *Let $\alpha : A \rightarrow B$, $\beta : B \rightarrow C$, and $\gamma : A \rightarrow C$. Then,*

$$\gamma \sqsubseteq (\alpha \triangleright \beta) \Leftrightarrow \alpha^\# \cdot \gamma \sqsubseteq \beta.$$

Definition *axiom17* :=

$\forall (A\ B\ C : eqType)(\alpha : Rel\ A\ B)(\beta : Rel\ B\ C)(\gamma : Rel\ A\ C),$
 $\gamma \sqsubseteq (\alpha \triangleright \beta) \Leftrightarrow (\alpha^\# \cdot \gamma) \sqsubseteq \beta.$

Lemma *inc_residual* : *axiom17*.

Proof.

```

move  $\Rightarrow A\ B\ C\ \alpha\ \beta\ \gamma$ .
split; move  $\Rightarrow H$ .
move  $\Rightarrow b\ c$ .
elim  $\Rightarrow a\ H0$ .
apply ( $H\ a$ ).
apply  $H0$ .
apply  $H0$ .
move  $\Rightarrow a\ c\ H0\ b\ H1$ .
apply  $H$ .
 $\exists a$ .
split.
apply  $H1$ .
apply  $H0$ .
Qed.

```

3.3.2 排中律

Dedekind 圏の公理のほかに、以下の“排中律”を仮定すれば、与えられる圏は Schröder 圏となり、Bool 代数の性質も満たされる。

Lemma 18 (complement_classic) *Let $\alpha : A \rightarrow B$. Then,*

$$\alpha \sqcup \alpha^- = \nabla_{AB}$$

Definition *axiom18* := $\forall (A B : eqType)(\alpha : Rel A B),$
 $\alpha \cup \alpha^- = \nabla_{AB}$.

Lemma *complement_classic* : *axiom18*.

Proof.

move $\Rightarrow A B \alpha$.

apply *functional_extensionality*.

move $\Rightarrow a$.

apply *functional_extensionality*.

move $\Rightarrow b$.

apply *prop_extensionality_ok*.

split; move $\Rightarrow H$.

apply *I*.

apply *classic*.

Qed.

3.3.3 単域

1 点集合 I が単域となるための条件は

$$\phi_{II} \neq id_I \wedge id_I = \nabla_{II} \wedge \forall X, \nabla_{XI} \cdot \nabla_{IX} = \nabla_{XX}$$

だが、Rel の定義から左 2 つは証明できるため、右の式だけ仮定する。

Lemma 19 (unit_universal)

$$\nabla_{AI} \cdot \nabla_{IA} = \nabla_{AA}$$

Definition *axiom19* := $\forall (A : eqType), \quad A \cdot i \cdot i A = A A$.

Lemma *unit_universal* : *axiom19*.

Proof.

move $\Rightarrow A$.

apply *functional_extensionality*.

move $\Rightarrow a$.

apply *functional_extensionality*.

```

move ⇒ a0.
apply prop_extensionality_ok.
split; move ⇒ H.
apply I.
∃ tt.
by [].
Qed.
    
```

3.3.4 点公理

まずは Dedekind 圏でない“条件付和関係”を定義する公理から.

Lemma 20 (inc_cupP) *Let $\alpha, \beta_\lambda : A \rightarrow B$ and $P : \text{predicate}$. Then,*

$$(\sqcup_{P(\lambda)} \beta_\lambda) \sqsubseteq \alpha \Leftrightarrow (\forall \lambda \in \Lambda, P(\lambda) \Rightarrow \beta_\lambda \sqsubseteq \alpha).$$

Definition axiom20 :=

$\forall (A\ B\ L : \text{eqType})(\alpha : \text{Rel } A\ B)(\beta_{_L} : L \rightarrow \text{Rel } A\ B)(P : L \rightarrow \text{Prop}),$
 $(\quad p\ P, \beta_{_L}) \quad \alpha \leftrightarrow \forall l : L, P\ l \rightarrow \beta_{_L}\ l \quad \alpha.$

Lemma inc_cupP : axiom20.

Proof.

```

move ⇒ A B L alpha beta_L P.
split.
move ⇒ H l H0 a b H1.
apply H.
∃ l.
split.
apply H0.
apply H1.
move ⇒ H a b.
elim ⇒ l.
elim ⇒ H0 H1.
apply (H l H0 a b H1).
Qed.
    
```

この“弱選択公理”を仮定すれば、排中律と単域の存在 (厳密には全域性公理) を利用して点公理を導出できる.

Lemma 21 (weak_axiom_of_choice) *Let $\alpha : I \rightarrow A$ be a total relation. Then,*

$$\exists \beta : I \rightarrow A, \beta \sqsubseteq \alpha.$$

Definition axiom21 := $\forall (A : \text{eqType})(\alpha : \text{Rel } i\ A),$

total_r alpha $\rightarrow \exists \text{beta} : \text{Rel } i \ A, \text{function_r beta} \wedge \text{beta} \quad \text{alpha}.$

Lemma *weak_axiom_of_choice* : *axiom21*.

Proof.

move \Rightarrow *A alpha*.

rewrite */function_r/total_r/univalent_r/identity/include/composite/inverse*.

move \Rightarrow *H*.

move : (*H tt tt* (*Logic.eq_refl tt*)).

elim \Rightarrow *a H0*.

\exists (**fun** ($_ : i$))(*a0* : *A*) \Rightarrow *a = a0*.

repeat split.

move \Rightarrow *tt tt0 H1*.

by [\exists *a*].

move \Rightarrow *a0 a1*.

elim \Rightarrow *tt0*.

elim \Rightarrow *H1 H2*.

by [rewrite *-H1 -H2*].

induction *a0*.

move \Rightarrow *a0 H1*.

rewrite *-H1*.

apply *H0*.

Qed.

3.3.5 関係の有理性

集合の選択公理 (*Logic.IndefiniteDescription*) や証明の一意性

(*Logic.ProofIrrelevance*) を仮定すれば、集合論上ならごり押しで証明できる。

旧ライブラリの頃は無理だと諦めて *Axiom* を追加していたが、*Standard Library* のインポートだけで解けた。正直びっくり。

Lemma 22 (rationality) *Let* $\alpha : A \rightarrow B$. *Then,*

$$\exists R, \exists f : R \rightarrow A, \exists g : R \rightarrow B, \alpha = f^\# \cdot g \wedge f \cdot f^\# \sqcap g \cdot g^\# = id_R.$$

この付近は、ごり押しのための補題。命題の真偽を選択公理で *bool* 値に変換したり、部分集合の元から上位集合の元を生成する *sval* (*proj1_sig*) の単射性を示したりしている。

Lemma *is_true_inv0* : $\forall P : \text{Prop}, \exists b : \text{bool}, P \leftrightarrow \text{is_true } b$.

Proof.

move \Rightarrow *P*.

case (*classic P*); move \Rightarrow *H*.

\exists *true*.

split; move \Rightarrow *H0*.

```

by [].
apply H.
 $\exists$  false.
split; move  $\Rightarrow$  H0.
apply False_ind.
apply (H H0).
discriminate H0.
Qed.
Definition is_true_inv : Prop  $\rightarrow$  bool.
move  $\Rightarrow$  P.
move : (is_true_inv0 P)  $\Rightarrow$  H.
apply constructive_indefinite_description in H.
apply H.
Defined.
Lemma is_true_id :  $\forall$  P : Prop, is_true (is_true_inv P)  $\leftrightarrow$  P.
Proof.
move  $\Rightarrow$  P.
unfold is_true_inv.
move : (constructive_indefinite_description (fun b : bool  $\Rightarrow$  P  $\leftrightarrow$  is_true b) (is_true_inv0 P))  $\Rightarrow$  x0.
apply (@sig_ind bool (fun b  $\Rightarrow$  (P  $\leftrightarrow$  is_true b)) (fun y  $\Rightarrow$  is_true (let (x, _) := y in x)  $\leftrightarrow$  P)).
move  $\Rightarrow$  x H.
apply iff_sym.
apply H.
Qed.
Lemma sval_inv :  $\forall$  (A : Type)(P : A  $\rightarrow$  Prop)(x : sig P)(a : A), a = sval x  $\rightarrow \exists$  (H : P a), x = exist P a H.
Proof.
move  $\Rightarrow$  A P x a H0.
rewrite H0.
 $\exists$  (proj2_sig x).
apply (@sig_ind A P (fun y  $\Rightarrow$  y = exist P (sval y) (proj2_sig y))).
move  $\Rightarrow$  a0 H.
by [simpl].
Qed.
Lemma sval_injective :  $\forall$  (A : Type)(P : A  $\rightarrow$  Prop)(x y : sig P), sval x = sval y  $\rightarrow$  x = y.
Proof.
move  $\Rightarrow$  A P x y H.
move : (sval_inv A P y (sval x) H).
elim  $\Rightarrow$  H0 H1.
rewrite H1.

```

```

assert ( $H0 = \text{proj2\_sig } x$ ).
apply proof_irrelevance.
rewrite  $H2$ .
apply (@sig_ind  $A P$  ( $\text{fun } y \Rightarrow y = \text{exist } P (sval \ y) (\text{proj2\_sig } y)$ )).
move  $\Rightarrow a0 \ H3$ .
by [simpl].
Qed.

```

Definition *axiom22* := $\forall (A \ B : \text{eqType})(\alpha : \text{Rel } A \ B),$
 $\exists (R : \text{eqType})(f : \text{Rel } R \ A)(g : \text{Rel } R \ B),$
 $\text{function_r } f \wedge \text{function_r } g \wedge \alpha = f \# \cdot g \wedge ((f \cdot f \#) \quad (g \cdot g \#)) = \text{Id } R.$

Lemma *rationality* : *axiom22*.

Proof.

```

move  $\Rightarrow A \ B \ \alpha$ .
rewrite /function_r/total_r/univalent_r/identity/cap/composite/inverse/include.
 $\exists (\text{sig\_eqType } (\text{fun } x : \text{prod\_eqType } A \ B \Rightarrow \text{is\_true\_inv } (\alpha \ (\text{fst } x) \ (\text{snd } x))))$ .
 $\exists (\text{fun } x \ a \Rightarrow a = (\text{fst } (sval \ x)))$ .
 $\exists (\text{fun } x \ b \Rightarrow b = (\text{snd } (sval \ x)))$ .
simpl.
repeat split.
move  $\Rightarrow x \ x0 \ H$ .
 $\exists (\text{fst } (sval \ x))$ .
repeat split.
by [rewrite  $H$ ].
move  $\Rightarrow a \ a0$ .
elim  $\Rightarrow x$ .
elim  $\Rightarrow H \ H0$ .
by [rewrite  $H \ H0$ ].
move  $\Rightarrow x \ x0 \ H$ .
 $\exists (\text{snd } (sval \ x))$ .
repeat split.
by [rewrite  $H$ ].
move  $\Rightarrow b \ b0$ .
elim  $\Rightarrow x$ .
elim  $\Rightarrow H \ H0$ .
by [rewrite  $H \ H0$ ].
apply functional_extensionality.
move  $\Rightarrow a$ .
apply functional_extensionality.
move  $\Rightarrow b$ .
apply prop_extensionality_ok.
split; move  $\Rightarrow H$ .

```

```

assert (is_true (is_true_inv (alpha (fst (a,b)) (snd (a,b))))) .
simpl.
apply is_true_id.
apply H.
 $\exists$  (exist (fun x  $\Rightarrow$  (is_true (is_true_inv (alpha (fst x) (snd x))))) (a,b) H0).
by [simpl].
elim H  $\Rightarrow$  x.
elim  $\Rightarrow$  H0 H1.
rewrite H0 H1.
apply is_true_id.
apply (@sig_ind (A  $\times$  B) (fun x  $\Rightarrow$  is_true (is_true_inv (alpha (fst x) (snd x))))) (fun x
 $\Rightarrow$  is_true (is_true_inv (alpha (fst (sval x)) (snd (sval x))))) .
simpl.
by [move  $\Rightarrow$  x0].
apply functional_extensionality.
move  $\Rightarrow$  y.
apply functional_extensionality.
move  $\Rightarrow$  y0.
apply prop_extensionality_ok.
split; move  $\Rightarrow$  H.
apply sval_injective.
elim H  $\Rightarrow$  H0 H1.
elim H0  $\Rightarrow$  a.
elim  $\Rightarrow$  H2 H3.
elim H1  $\Rightarrow$  b.
elim  $\Rightarrow$  H4 H5.
rewrite (surjective_pairing (sval y0)) -H3 -H5 H2 H4.
apply surjective_pairing.
rewrite H.
split.
 $\exists$  (fst (sval y0)).
repeat split.
 $\exists$  (snd (sval y0)).
repeat split.
Qed.

```

3.3.6 直和と直積

任意の直和に対して, 入射対が存在することを仮定する.

Lemma 23 (*pair_of_inclusions*) $\exists j : A \rightarrow A + B, \exists k : B \rightarrow A + B,$

$$j \cdot j^\# = id_A \wedge k \cdot k^\# = id_B \wedge j \cdot k^\# = \phi_{AB} \wedge j^\# \cdot j \sqcup k^\# \cdot k = id_{A+B}.$$

Definition *axiom23* :=

$\forall (A\ B : eqType), \exists (j : Rel\ A\ (sum_eqType\ A\ B))(k : Rel\ B\ (sum_eqType\ A\ B)),$
 $j \cdot j^\# = Id\ A \wedge k \cdot k^\# = Id\ B \wedge j \cdot k^\# = \phi_{AB} \wedge$
 $(j^\# \cdot j) \cdot (k^\# \cdot k) = Id\ (sum_eqType\ A\ B).$

Lemma *pair_of_inclusions* : *axiom23*.

Proof.

move $\Rightarrow A\ B$.

$\exists (\text{fun } (a : A)(x : sum_eqType\ A\ B) \Rightarrow x = inl\ a).$

$\exists (\text{fun } (b : B)(x : sum_eqType\ A\ B) \Rightarrow x = inr\ b).$

repeat split.

apply *functional_extensionality*.

move $\Rightarrow a$.

apply *functional_extensionality*.

move $\Rightarrow a0$.

apply *prop_extensionality_ok*.

split; move $\Rightarrow H$.

elim $H \Rightarrow x$.

elim $\Rightarrow H0\ H1$.

rewrite $H0$ in $H1$.

by [injection $H1$].

$\exists (inl\ a).$

repeat split.

by [rewrite H].

apply *functional_extensionality*.

move $\Rightarrow b$.

apply *functional_extensionality*.

move $\Rightarrow b0$.

apply *prop_extensionality_ok*.

split; move $\Rightarrow H$.

elim $H \Rightarrow x$.

elim $\Rightarrow H0\ H1$.

rewrite $H0$ in $H1$.

by [injection $H1$].

$\exists (inr\ b).$

repeat split.


```
by [rewrite  $H$ ].
apply functional_extensionality.
move  $\Rightarrow a$ .
apply functional_extensionality.
move  $\Rightarrow b$ .
apply prop_extensionality_ok.
split; move  $\Rightarrow H$ .
elim  $H \Rightarrow x$ .
elim  $\Rightarrow H0\ H1$ .
rewrite  $H0$  in  $H1$ .
discriminate  $H1$ .
apply False_ind.
apply  $H$ .
apply functional_extensionality.
move  $\Rightarrow x$ .
apply functional_extensionality.
move  $\Rightarrow x0$ .
apply prop_extensionality_ok.
split.
case.
elim  $\Rightarrow a$ .
elim  $\Rightarrow H0\ H1$ .
by [rewrite  $H0\ H1$ ].
elim  $\Rightarrow b$ .
elim  $\Rightarrow H0\ H1$ .
by [rewrite  $H0\ H1$ ].
move :  $x0$ .
apply (sum_ind (fun  $x0 \Rightarrow x = x0 \rightarrow (\exists b : A, x = \text{inl } b \wedge x0 = \text{inl } b) \vee (\exists b : B, x = \text{inr } b \wedge x0 = \text{inr } b)$ ))).
move  $\Rightarrow a\ H$ .
left.
 $\exists a$ .
repeat split.
apply  $H$ .
move  $\Rightarrow b\ H$ .
right.
 $\exists b$ .
repeat split.
apply  $H$ .
Qed.
```

任意の直積に対して, 射影対が存在することを仮定する.

Lemma 24 (pair_of_projections) $\exists p : A \times B \rightarrow A, \exists q : A \times B \rightarrow B,$

$$p^\# \cdot q = \nabla_{AB} \wedge p \cdot p^\# \sqcap q \cdot q^\# = id_{A \times B}.$$

Definition *axiom24* :=

$\forall (A B : eqType), \exists (p : Rel (prod_eqType A B) A)(q : Rel (prod_eqType A B) B),$
 $p \# \cdot q = A B \wedge (p \cdot p \#) \quad (q \cdot q \#) = Id (prod_eqType A B) \wedge univalent_r p$
 $\wedge univalent_r q.$

Lemma *pair_of_projections* : *axiom24*.

Proof.

move $\Rightarrow A B$.

\exists (**fun** ($x : prod_eqType A B$)($a : A$) $\Rightarrow a = (fst x)$).

\exists (**fun** ($x : prod_eqType A B$)($b : B$) $\Rightarrow b = (snd x)$).

split.

apply *functional_extensionality*.

move $\Rightarrow a$.

apply *functional_extensionality*.

move $\Rightarrow b$.

apply *prop_extensionality_ok*.

split; move $\Rightarrow H$.

apply *I*.

$\exists (a, b)$.

by [simpl].

split.

apply *functional_extensionality*.

move $\Rightarrow x$.

apply *functional_extensionality*.

move $\Rightarrow x0$.

apply *prop_extensionality_ok*.

split.

repeat elim.

move $\Rightarrow a$.

elim $\Rightarrow H H0$.

elim $\Rightarrow b$.

elim $\Rightarrow H1 H2$.

rewrite (*surjective_pairing* $x0$) -*H0* -*H2* *H* *H1*.

apply *surjective_pairing*.

move $\Rightarrow H$.

rewrite *H*.

split.

by [\exists (*fst* $x0$)].

```
by [∃ (snd x0)].
split.
move ⇒ a a0.
elim ⇒ x.
elim ⇒ H H0.
by [rewrite H H0].
move ⇒ b b0.
elim ⇒ x.
elim ⇒ H H0.
by [rewrite H H0].
Qed.
```

Chapter 4

Library `Basic_Lemmas`

```
Require Import Basic_Notations.  
Require Import Logic.Classical_Prop.
```

4.1 束論に関する補題

4.1.1 和関係, 共通関係

Lemma 25 (cap_l) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$\alpha \sqcap \beta \sqsubseteq \alpha.$$

```
Lemma cap_l {A B : eqType} {alpha beta : Rel A B}: (alpha beta) alpha.
```

Proof.

```
assert ((alpha beta) (alpha beta)).
```

```
apply inc_refl.
```

```
apply inc_cap in H.
```

```
apply H.
```

Qed.

Lemma 26 (cap_r) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$\alpha \sqcap \beta \sqsubseteq \beta.$$

```
Lemma cap_r {A B : eqType} {alpha beta : Rel A B}: (alpha beta) beta.
```

Proof.

```
assert ((alpha beta) (alpha beta)).
```

```
apply inc_refl.
```

```
apply inc_cap in H.
```

```
apply H.
```

Qed.

Lemma 27 (cup_l) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$\alpha \sqsubseteq \alpha \sqcup \beta.$$

Lemma cup_l $\{A B : eqType\} \{alpha beta : Rel A B\} : alpha \sqsubseteq (alpha \sqcup beta).$

Proof.

assert $((alpha \sqsubseteq beta) \rightarrow (alpha \sqsubseteq beta)).$

apply *inc_refl*.

apply *inc_cup* in *H*.

apply *H*.

Qed.

Lemma 28 (cup_r) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$\beta \sqsubseteq \alpha \sqcup \beta.$$

Lemma cup_r $\{A B : eqType\} \{alpha beta : Rel A B\} : beta \sqsubseteq (alpha \sqcup beta).$

Proof.

assert $((alpha \sqsubseteq beta) \rightarrow (alpha \sqsubseteq beta)).$

apply *inc_refl*.

apply *inc_cup* in *H*.

apply *H*.

Qed.

Lemma 29 (inc_def1) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$\alpha = \alpha \sqcap \beta \Leftrightarrow \alpha \sqsubseteq \beta.$$

Lemma inc_def1 $\{A B : eqType\} \{alpha beta : Rel A B\} :$

$$alpha = alpha \sqcap beta \leftrightarrow alpha \sqsubseteq beta.$$

Proof.

split; move \Rightarrow *H*.

assert $(alpha \sqsubseteq (alpha \sqcap beta)).$

rewrite -*H*.

apply *inc_refl*.

apply *inc_cap* in *H0*.

apply *H0*.

apply *inc_antisym*.

apply *inc_cap*.

split.

apply *inc_refl*.

apply *H*.
 apply *cap_l*.
 Qed.

Lemma 30 (inc_def2) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$\beta = \alpha \sqcup \beta \Leftrightarrow \alpha \sqsubseteq \beta.$$

Lemma inc_def2 $\{A\ B : eqType\} \{alpha\ beta : Rel\ A\ B\}$:
 $beta = alpha \quad beta \leftrightarrow alpha \quad beta.$

Proof.

split; move \Rightarrow *H*.
 assert $((alpha \quad beta) \quad beta).$
 rewrite -*H*.
 apply *inc_refl*.
 apply *inc_cup* in *H0*.
 apply *H0*.
 apply *inc_antisym*.
 assert $((alpha \quad beta) \quad (alpha \quad beta)).$
 apply *inc_refl*.
 apply *cup_r*.
 apply *inc_cup*.
 split.
 apply *H*.
 apply *inc_refl*.
 Qed.

Lemma 31 (cap_assoc) *Let $\alpha, \beta, \gamma : A \rightarrow B$. Then,*

$$(\alpha \sqcap \beta) \sqcap \gamma = \alpha \sqcap (\beta \sqcap \gamma).$$

Lemma cap_assoc $\{A\ B : eqType\} \{alpha\ beta\ gamma : Rel\ A\ B\}$:
 $(alpha \quad beta) \quad gamma = alpha \quad (beta \quad gamma).$

Proof.

apply *inc_antisym*.
 rewrite *inc_cap*.
 split.
 apply $(inc_trans _ _ (alpha \quad beta)).$
 apply *cap_l*.
 apply *cap_l*.
 rewrite *inc_cap*.
 split.
 apply $(inc_trans _ _ (alpha \quad beta)).$

```

apply cap_l.
apply cap_r.
apply cap_r.
rewrite inc_cap.
split.
rewrite inc_cap.
split.
apply cap_l.
apply (inc_trans _ _ _ (beta gamma)).
apply cap_r.
apply cap_l.
apply (inc_trans _ _ _ (beta gamma)).
apply cap_r.
apply cap_r.
Qed.

```

Lemma 32 (cup_assoc) *Let $\alpha, \beta, \gamma : A \rightarrow B$. Then,*

$$(\alpha \sqcup \beta) \sqcup \gamma = \alpha \sqcup (\beta \sqcup \gamma).$$

Lemma cup_assoc $\{A B : eqType\} \{alpha \ beta \ gamma : Rel \ A \ B\}$:
 $(alpha \ \beta) \ \gamma = alpha \ (\beta \ \gamma).$

Proof.

```

apply inc_antisym.
rewrite inc_cup.
split.
rewrite inc_cup.
split.
apply cup_l.
apply (inc_trans _ _ _ (beta gamma)).
apply cup_l.
apply cup_r.
apply (inc_trans _ _ _ (beta gamma)).
apply cup_r.
apply cup_r.
rewrite inc_cup.
split.
apply (inc_trans _ _ _ (alpha beta)).
apply cup_l.
apply cup_l.
rewrite inc_cup.
split.
apply (inc_trans _ _ _ (alpha beta)).

```

CHAPTER 4. LIBRARY BASIC_LEMMAS

apply *cup_r*.
apply *cup_l*.
apply *cup_r*.
Qed.

Lemma 33 (cap_comm) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$\alpha \sqcap \beta = \beta \sqcap \alpha.$$

Lemma *cap_comm* {*A B : eqType*} {*alpha beta : Rel A B*}: *alpha* **beta = beta** *alpha*.

Proof.

apply *inc_antisym*.
rewrite *inc_cap*.
split.
apply *cap_r*.
apply *cap_l*.
rewrite *inc_cap*.
split.
apply *cap_r*.
apply *cap_l*.
Qed.

Lemma 34 (cup_comm) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$\alpha \sqcup \beta = \beta \sqcup \alpha.$$

Lemma *cup_comm* {*A B : eqType*} {*alpha beta : Rel A B*}: *alpha* **beta = beta** *alpha*.

Proof.

apply *inc_antisym*.
rewrite *inc_cup*.
split.
apply *cup_r*.
apply *cup_l*.
rewrite *inc_cup*.
split.
apply *cup_r*.
apply *cup_l*.
Qed.

Lemma 35 (cup_cap_abs) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$\alpha \sqcup (\alpha \sqcap \beta) = \alpha.$$

CHAPTER 4. LIBRARY BASIC_LEMMAS

Lemma *cup_cap_abs* {*A B* : *eqType*} {*alpha beta* : *Rel A B*}:
 alpha (alpha beta) = *alpha*.

Proof.

move : (@cap_l _ _ alpha beta) \Rightarrow *H*.

apply *inc_def2* in *H*.

by [rewrite *cup_comm* -*H*].

Qed.

Lemma 36 (*cap_cup_abs*) *Let* $\alpha, \beta : A \rightarrow B$. *Then,*

$$\alpha \sqcap (\alpha \sqcup \beta) = \alpha.$$

Lemma *cap_cup_abs* {*A B* : *eqType*} {*alpha beta* : *Rel A B*}:
 alpha (alpha beta) = *alpha*.

Proof.

move : (@cup_l _ _ alpha beta) \Rightarrow *H*.

apply *inc_def1* in *H*.

by [rewrite -*H*].

Qed.

Lemma 37 (*cap_idem*) *Let* $\alpha : A \rightarrow B$. *Then,*

$$\alpha \sqcap \alpha = \alpha.$$

Lemma *cap_idem* {*A B* : *eqType*} {*alpha* : *Rel A B*}: *alpha* alpha = *alpha*.

Proof.

apply *inc_antisym*.

apply *cap_l*.

apply *inc_cap*.

split; apply *inc_refl*.

Qed.

Lemma 38 (*cup_idem*) *Let* $\alpha : A \rightarrow B$. *Then,*

$$\alpha \sqcup \alpha = \alpha.$$

Lemma *cup_idem* {*A B* : *eqType*} {*alpha* : *Rel A B*}: *alpha* alpha = *alpha*.

Proof.

apply *inc_antisym*.

apply *inc_cup*.

split; apply *inc_refl*.

apply *cup_l*.

Qed.

CHAPTER 4. LIBRARY BASIC_LEMMAS

Lemma 39 (cap_inc_compat) *Let $\alpha, \alpha', \beta, \beta' : A \rightarrow B$. Then,*

$$\alpha \sqsubseteq \alpha' \wedge \beta \sqsubseteq \beta' \Rightarrow \alpha \sqcap \beta \sqsubseteq \alpha' \sqcap \beta'.$$

Lemma `cap_inc_compat` $\{A\ B : eqType\} \{alpha\ alpha'\ beta\ beta' : Rel\ A\ B\}$:
 $alpha\ alpha' \rightarrow beta\ beta' \rightarrow (alpha\ beta)\ (alpha'\ beta')$.

Proof.

`move` \Rightarrow $H\ H0$.

`rewrite` `-inc_def1`.

`apply` `inc_def1` in H .

`apply` `inc_def1` in $H0$.

`rewrite` `cap_assoc` `-(@cap_assoc - - beta)`.

`rewrite` `(@cap_comm - - beta)`.

`rewrite` `cap_assoc` `-(@cap_assoc - - alpha)`.

`by` [`rewrite` `-H -H0`].

Qed.

Lemma 40 (cap_inc_compat_l) *Let $\alpha, \beta, \beta' : A \rightarrow B$. Then,*

$$\beta \sqsubseteq \beta' \Rightarrow \alpha \sqcap \beta \sqsubseteq \alpha \sqcap \beta'.$$

Lemma `cap_inc_compat_l` $\{A\ B : eqType\} \{alpha\ beta\ beta' : Rel\ A\ B\}$:
 $beta\ beta' \rightarrow (alpha\ beta)\ (alpha\ beta')$.

Proof.

`move` \Rightarrow H .

`apply` `(@cap_inc_compat - - - - - (@inc_refl - - alpha) H)`.

Qed.

Lemma 41 (cap_inc_compat_r) *Let $\alpha, \alpha', \beta : A \rightarrow B$. Then,*

$$\alpha \sqsubseteq \alpha' \Rightarrow \alpha \sqcap \beta \sqsubseteq \alpha' \sqcap \beta.$$

Lemma `cap_inc_compat_r` $\{A\ B : eqType\} \{alpha\ alpha'\ beta : Rel\ A\ B\}$:
 $alpha\ alpha' \rightarrow (alpha\ beta)\ (alpha'\ beta)$.

Proof.

`move` \Rightarrow H .

`apply` `(@cap_inc_compat - - - - - H (@inc_refl - - beta))`.

Qed.

Lemma 42 (cup_inc_compat) *Let $\alpha, \alpha', \beta, \beta' : A \rightarrow B$. Then,*

$$\alpha \sqsubseteq \alpha' \wedge \beta \sqsubseteq \beta' \Rightarrow \alpha \sqcup \beta \sqsubseteq \alpha' \sqcup \beta'.$$

CHAPTER 4. LIBRARY BASIC_LEMMAS

Lemma *cup_inc_compat* {*A B : eqType*} {*alpha alpha' beta beta' : Rel A B*}:
alpha alpha' → beta beta' → (alpha beta) (alpha' beta').

Proof.

move \Rightarrow *H H0*.

rewrite *-inc_def2*.

apply *inc_def2* in *H*.

apply *inc_def2* in *H0*.

rewrite *cup_assoc* -(@*cup_assoc* - - *beta*).

rewrite (@*cup_comm* - - *beta*).

rewrite *cup_assoc* -(@*cup_assoc* - - *alpha*).

by [rewrite *-H -H0*].

Qed.

Lemma 43 (*cup_inc_compat_l*) *Let $\alpha, \beta, \beta' : A \rightarrow B$. Then,*

$$\beta \sqsubseteq \beta' \Rightarrow \alpha \sqcup \beta \sqsubseteq \alpha \sqcup \beta'.$$

Lemma *cup_inc_compat_l* {*A B : eqType*} {*alpha beta beta' : Rel A B*}:
beta beta' → (alpha beta) (alpha beta').

Proof.

move \Rightarrow *H*.

apply (@*cup_inc_compat* - - - - - (@*inc_refl* - - *alpha*) *H*).

Qed.

Lemma 44 (*cup_inc_compat_r*) *Let $\alpha, \alpha', \beta : A \rightarrow B$. Then,*

$$\alpha \sqsubseteq \alpha' \Rightarrow \alpha \sqcup \beta \sqsubseteq \alpha' \sqcup \beta.$$

Lemma *cup_inc_compat_r* {*A B : eqType*} {*alpha alpha' beta : Rel A B*}:
alpha alpha' → (alpha beta) (alpha' beta).

Proof.

move \Rightarrow *H*.

apply (@*cup_inc_compat* - - - - - *H* (@*inc_refl* - - *beta*)).

Qed.

Lemma 45 (*cap_empty*) *Let $\alpha : A \rightarrow B$. Then,*

$$\alpha \sqcap \phi_{AB} = \phi_{AB}.$$

Lemma *cap_empty* {*A B : eqType*} {*alpha : Rel A B*}: *alpha A B = A B*.

Proof.

apply *inc_antisym*.

apply *cap_r*.

CHAPTER 4. LIBRARY BASIC_LEMMAS

apply *inc_empty_alpha*.

Qed.

Lemma 46 (cup_empty) *Let $\alpha : A \rightarrow B$. Then,*

$$\alpha \sqcup \phi_{AB} = \alpha.$$

Lemma *cup_empty* { $A B : eqType$ } { $\alpha : Rel A B$ }: α $A B = \alpha$.

Proof.

apply *inc_antisym*.

apply *inc_cup*.

split.

apply *inc_refl*.

apply *inc_empty_alpha*.

apply *cup_l*.

Qed.

Lemma 47 (cap_universal) *Let $\alpha : A \rightarrow B$. Then,*

$$\alpha \sqcap \nabla_{AB} = \alpha.$$

Lemma *cap_universal* { $A B : eqType$ } { $\alpha : Rel A B$ }: α $A B = \alpha$.

Proof.

apply *inc_antisym*.

apply *cap_l*.

apply *inc_cap*.

split.

apply *inc_refl*.

apply *inc_alpha_universal*.

Qed.

Lemma 48 (cup_universal) *Let $\alpha : A \rightarrow B$. Then,*

$$\alpha \sqcup \nabla_{AB} = \nabla_{AB}.$$

Lemma *cup_universal* { $A B : eqType$ } { $\alpha : Rel A B$ }: α $A B = \nabla_{AB}$.

Proof.

apply *inc_antisym*.

apply *inc_cup*.

split.

apply *inc_alpha_universal*.

apply *inc_refl*.

apply *cup_r*.

Qed.

Lemma 49 (inc_lower) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$\alpha = \beta \Leftrightarrow (\forall \gamma : A \rightarrow B, \gamma \sqsubseteq \alpha \Leftrightarrow \gamma \sqsubseteq \beta).$$

Lemma inc_lower $\{A\ B : eqType\} \{alpha\ beta : Rel\ A\ B\}$:

$$alpha = beta \Leftrightarrow (\forall\ gamma : Rel\ A\ B, gamma\ alpha \Leftrightarrow gamma\ beta).$$

Proof.

split; move $\Rightarrow H$.

move $\Rightarrow gamma$.

by [rewrite H].

apply *inc_antisym*.

rewrite $-H$.

apply *inc_refl*.

rewrite H .

apply *inc_refl*.

Qed.

Lemma 50 (inc_upper) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$\alpha = \beta \Leftrightarrow (\forall \gamma : A \rightarrow B, \alpha \sqsubseteq \gamma \Leftrightarrow \beta \sqsubseteq \gamma).$$

Lemma inc_upper $\{A\ B : eqType\} \{alpha\ beta : Rel\ A\ B\}$:

$$alpha = beta \Leftrightarrow (\forall\ gamma : Rel\ A\ B, alpha\ gamma \Leftrightarrow beta\ gamma).$$

Proof.

split; move $\Rightarrow H$.

move $\Rightarrow gamma$.

by [rewrite H].

apply *inc_antisym*.

rewrite H .

apply *inc_refl*.

rewrite $-H$.

apply *inc_refl*.

Qed.

4.1.2 分配法則

Lemma 51 (cap_cup_distr_l) *Let $\alpha, \beta, \gamma : A \rightarrow B$. Then,*

$$\alpha \sqcap (\beta \sqcup \gamma) = (\alpha \sqcap \beta) \sqcup (\alpha \sqcap \gamma).$$

CHAPTER 4. LIBRARY BASIC_LEMMAS

Lemma *cap_cup_distr_l* {A B : eqType} {alpha beta gamma : Rel A B}:
 $\alpha \text{ (beta gamma) } = (\alpha \text{ beta}) \text{ (alpha gamma)}$.

Proof.

apply *inc_upper*.
 move \Rightarrow *delta*.
 split; move \Rightarrow *H*.
 rewrite *cap_comm* (@*cap_comm* _ _ *gamma*).
 apply *inc_cup*.
 rewrite -*inc_rpc* -*inc_rpc*.
 apply *inc_cup*.
 rewrite *inc_rpc* *cap_comm*.
 apply *H*.
 rewrite *cap_comm* -*inc_rpc*.
 apply *inc_cup*.
 rewrite *inc_rpc* *inc_rpc*.
 apply *inc_cup*.
 rewrite *cap_comm* (@*cap_comm* _ _ *gamma*).
 apply *H*.

Qed.

Lemma 52 (cap_cup_distr_r) *Let $\alpha, \beta, \gamma : A \rightarrow B$. Then,*

$$(\alpha \sqcup \beta) \sqcap \gamma = (\alpha \sqcap \gamma) \sqcup (\beta \sqcap \gamma).$$

Lemma *cap_cup_distr_r* {A B : eqType} {alpha beta gamma : Rel A B}:
 $(\alpha \text{ beta}) \text{ gamma} = (\alpha \text{ gamma}) \text{ (beta gamma)}$.

Proof.

rewrite (@*cap_comm* _ _ (*alpha* *beta*)) (@*cap_comm* _ _ *alpha*) (@*cap_comm* _ _ *beta*).
 apply *cap_cup_distr_l*.

Qed.

Lemma 53 (cup_cap_distr_l) *Let $\alpha, \beta, \gamma : A \rightarrow B$. Then,*

$$\alpha \sqcup (\beta \sqcap \gamma) = (\alpha \sqcup \beta) \sqcap (\alpha \sqcup \gamma).$$

Lemma *cup_cap_distr_l* {A B : eqType} {alpha beta gamma : Rel A B}:
 $\alpha \text{ (beta gamma) } = (\alpha \text{ beta}) \text{ (alpha gamma)}$.

Proof.

rewrite *cap_cup_distr_l*.
 rewrite (@*cap_comm* _ _ (*alpha* *beta*)) *cap_cup_abs* (@*cap_comm* _ _ (*alpha* *beta*)).
 rewrite *cap_cup_distr_l*.
 rewrite -*cup_assoc* (@*cap_comm* _ _ *gamma*) *cup_cap_abs*.
 by [rewrite *cap_comm*].

Qed.

Lemma 54 (cup_cap_distr_r) *Let $\alpha, \beta, \gamma : A \rightarrow B$. Then,*

$$(\alpha \sqcap \beta) \sqcup \gamma = (\alpha \sqcup \gamma) \sqcap (\beta \sqcup \gamma).$$

Lemma cup_cap_distr_r {A B : eqType} {alpha beta gamma : Rel A B}:
(alpha beta) gamma = (alpha gamma) (beta gamma).

Proof.

rewrite (@cup_comm _ _ (alpha beta)) (@cup_comm _ _ alpha) (@cup_comm _ _ beta).
apply cup_cap_distr_l.

Qed.

Lemma 55 (cap_cup_unique) *Let $\alpha, \beta, \gamma : A \rightarrow B$. Then,*

$$\alpha \sqcap \beta = \alpha \sqcap \gamma \wedge \alpha \sqcup \beta = \alpha \sqcup \gamma \Rightarrow \beta = \gamma.$$

Lemma cap_cup_unique {A B : eqType} {alpha beta gamma : Rel A B}:
alpha beta = alpha gamma \rightarrow alpha beta = alpha gamma \rightarrow beta = gamma.

Proof.

move \Rightarrow H H0.
rewrite -(@cap_cup_abs _ _ beta alpha) cup_comm H0.
rewrite cap_cup_distr_l.
rewrite cap_comm H.
rewrite -cap_cup_distr_r.
rewrite H0 cap_comm cup_comm.
apply cap_cup_abs.

Qed.

4.1.3 原子性

空関係でない $\alpha : A \rightarrow B$ が, 任意の $\beta : A \rightarrow B$ について

$$\beta \sqsubseteq \alpha \Rightarrow \beta = \phi_{AB} \vee \beta = \alpha$$

を満たすとき, α は原子的 (atomic) であると言われる.

Definition atomic {A B : eqType} (alpha : Rel A B):=
alpha \neq $\phi_{AB} \wedge (\forall \beta : Rel A B, \beta \sqsubseteq \alpha \rightarrow \beta = \phi_{AB} \vee \beta = \alpha).$

CHAPTER 4. LIBRARY BASIC_LEMMAS

Lemma 56 (atomic_cap_empty) *Let $\alpha, \beta : A \rightarrow B$ are atomic and $\alpha \neq \beta$. Then,*

$$\alpha \sqcap \beta = \phi_{AB}.$$

Lemma *atomic_cap_empty* { $A B : eqType$ } { $\alpha \beta : Rel A B$ }:
 $atomic \ \alpha \rightarrow atomic \ \beta \rightarrow \alpha \neq \beta \rightarrow \alpha \sqcap \beta = \phi_{AB}.$

Proof.

```
move  $\Rightarrow$   $H \ H0$ .
apply or_to_imply.
case (classic ( $\alpha \sqcap \beta = \phi_{AB}$ )); move  $\Rightarrow$   $H1$ .
right.
apply  $H1$ .
left.
move  $\Rightarrow$   $H2$ .
apply  $H2$ .
apply inc_antisym.
apply inc_def1.
elim  $H \Rightarrow H3 \ H4$ .
case ( $H4 \ (\alpha \sqcap \beta) \ (@cap\_l \ \_ \ \_ \ \_)$ ); move  $\Rightarrow$   $H5$ .
apply False_ind.
apply ( $H1 \ H5$ ).
by [rewrite  $H5$ ].
apply inc_def1.
elim  $H0 \Rightarrow H3 \ H4$ .
case ( $H4 \ (\alpha \sqcap \beta) \ (@cap\_r \ \_ \ \_ \ \_)$ ); move  $\Rightarrow$   $H5$ .
apply False_ind.
apply ( $H1 \ H5$ ).
by [rewrite cap_comm  $H5$ ].
```

Qed.

Lemma 57 (atomic_cup) *Let $\alpha, \beta, \gamma : A \rightarrow B$ and α is atomic. Then,*

$$\alpha \sqsubseteq \beta \sqcup \gamma \Rightarrow \alpha \sqsubseteq \beta \vee \alpha \sqsubseteq \gamma.$$

Lemma *atomic_cup* { $A B : eqType$ } { $\alpha \beta \gamma : Rel A B$ }:
 $atomic \ \alpha \rightarrow \alpha \sqsubseteq \beta \sqcup \gamma \rightarrow \alpha \sqsubseteq \beta \vee \alpha \sqsubseteq \gamma.$

Proof.

```
move  $\Rightarrow$   $H \ H0$ .
apply inc_def1 in  $H0$ .
rewrite cap_cup_distr_l in  $H0$ .
elim  $H \Rightarrow H1 \ H2$ .
rewrite  $H0$  in  $H1$ .
assert ( $\alpha \sqcap \beta \neq \alpha \sqcap \gamma \vee \alpha \sqcap \beta \neq \alpha \sqcap \gamma$ ).
```



```

apply not_and_or.
elim  $\Rightarrow$   $H3$   $H4$ .
rewrite  $H3$   $H4$  in  $H1$ .
apply  $H1$ .
by [rewrite cup_empty].
case  $H3$ ; move  $\Rightarrow$   $H4$ .
left.
apply inc_def1.
case ( $H2$  ( $\alpha$   $\beta$ ) (@cap_l _ _ _)); move  $\Rightarrow$   $H5$ .
apply False_ind.
apply ( $H4$   $H5$ ).
by [rewrite  $H5$ ].
right.
apply inc_def1.
case ( $H2$  ( $\alpha$   $\gamma$ ) (@cap_l _ _ _)); move  $\Rightarrow$   $H5$ .
apply False_ind.
apply ( $H4$   $H5$ ).
by [rewrite  $H5$ ].
Qed.

```

4.2 Heyting 代数に関する補題

Lemma 58 (rpc_universal) *Let $\alpha : A \rightarrow B$. Then,*

$$(\alpha \Rightarrow \alpha) = \nabla_{AB}.$$

Lemma *rpc_universal* { A B : eqType} { α : Rel A B }: ($\alpha \gg \alpha$) = A B .

Proof.

```

apply inc_lower.
move  $\Rightarrow$   $\gamma$ .
split; move  $\Rightarrow$   $H$ .
apply inc_alpha_universal.
apply inc_rpc.
apply cap_r.
Qed.

```

Lemma 59 (rpc_r) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$(\alpha \Rightarrow \beta) \sqcap \beta = \beta.$$

Lemma *rpc_r* { A B : eqType} { α β : Rel A B }: ($\alpha \gg \beta$) β = β .

Proof.

```
assert (beta (alpha » beta)).
apply inc_rpc.
apply cap_l.
apply inc_def1 in H.
by [rewrite cap_comm -H].
Qed.
```

Lemma 60 (inc_def3) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$(\alpha \Rightarrow \beta) = \nabla_{AB} \Leftrightarrow \alpha \sqsubseteq \beta.$$

Lemma inc_def3 $\{A B : eqType\} \{alpha beta : Rel A B\}$:
 $(alpha \Rightarrow beta) = A B \leftrightarrow alpha \sqsubseteq beta$.

Proof.

```
split; move => H.
rewrite -(@rpc_universal _ _ alpha) in H.
assert ((alpha » alpha) (alpha » beta)).
rewrite H.
apply inc_refl.
apply inc_rpc in H0.
rewrite rpc_r in H0.
apply H0.
apply inc_antisym.
apply inc_alpha_universal.
rewrite -(@rpc_universal _ _ alpha).
apply inc_rpc.
rewrite rpc_r.
apply H.
Qed.
```

Lemma 61 (rpc_l) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$\alpha \sqcap (\alpha \Rightarrow \beta) = \alpha \sqcap \beta.$$

Lemma rpc_l $\{A B : eqType\} \{alpha beta : Rel A B\}$:
 $alpha \sqcap (alpha \Rightarrow beta) = alpha \sqcap beta$.

Proof.

```
apply inc_lower.
move => gamma.
split; move => H.
apply inc_cap.
apply inc_cap in H.
```

```

split.
apply H.
elim H ⇒ H0 H1.
apply inc_rpc in H1.
rewrite -(@cap_idem _ _ gamma).
apply (inc_trans _ _ _ (gamma alpha)).
apply cap_inc_compat.
apply inc_refl.
apply H0.
apply H1.
apply inc_cap.
apply inc_cap in H.
split.
apply H.
apply inc_rpc.
apply (inc_trans _ _ _ gamma).
apply cap_l.
apply H.
Qed.

```

Lemma 62 (rpc_inc_compat) *Let $\alpha, \alpha', \beta, \beta' : A \rightarrow B$. Then,*

$$\alpha' \sqsubseteq \alpha \wedge \beta \sqsubseteq \beta' \Rightarrow (\alpha \Rightarrow \beta) \sqsubseteq (\alpha' \Rightarrow \beta').$$

Lemma *rpc_inc_compat* {A B : eqType} {alpha alpha' beta beta' : Rel A B}:
 alpha' alpha → beta beta' → (alpha » beta) (alpha' » beta').

Proof.

```

move ⇒ H H0.
apply inc_rpc.
apply (@inc_trans _ _ _ ((alpha » beta) alpha)).
apply (@cap_inc_compat_l _ _ _ _ H).
rewrite cap_comm rpc_l.
apply (@inc_trans _ _ _ beta).
apply cap_r.
apply H0.
Qed.

```

Lemma 63 (rpc_inc_compat_l) *Let $\alpha, \beta, \beta' : A \rightarrow B$. Then,*

$$\beta \sqsubseteq \beta' \Rightarrow (\alpha \Rightarrow \beta) \sqsubseteq (\alpha \Rightarrow \beta').$$

Lemma *rpc_inc_compat_l* {A B : eqType} {alpha beta beta' : Rel A B}:
 beta beta' → (alpha » beta) (alpha » beta').

Proof.

move $\Rightarrow H$.

apply (@rpc_inc_compat _ _ _ _ (@inc_refl _ alpha) H).

Qed.

Lemma 64 (rpc_inc_compat_r) *Let $\alpha, \alpha', \beta : A \rightarrow B$. Then,*

$$\alpha' \sqsubseteq \alpha \Rightarrow (\alpha \Rightarrow \beta) \sqsubseteq (\alpha' \Rightarrow \beta).$$

Lemma *rpc_inc_compat_r* {A B : eqType} {alpha alpha' beta : Rel A B}:
 alpha' alpha \rightarrow (alpha » beta) (alpha' » beta).

Proof.

move $\Rightarrow H$.

apply (@rpc_inc_compat _ _ _ _ H (@inc_refl _ beta)).

Qed.

Lemma 65 (rpc_universal_alpha) *Let $\alpha : A \rightarrow B$. Then,*

$$\nabla_{AB} \Rightarrow \alpha = \alpha.$$

Lemma *rpc_universal_alpha* {A B : eqType} {alpha : Rel A B}: A B » alpha = alpha.

Proof.

apply inc_lower.

move \Rightarrow gamma.

split; move $\Rightarrow H$.

apply inc_rpc in H.

rewrite cap_universal in H.

apply H.

apply inc_rpc.

rewrite cap_universal.

apply H.

Qed.

Lemma 66 (rpc_lemma1) *Let $\alpha, \beta, \gamma : A \rightarrow B$. Then,*

$$(\alpha \Rightarrow \beta) \sqsubseteq ((\alpha \sqcap \gamma) \Rightarrow (\beta \sqcap \gamma)).$$

Lemma *rpc_lemma1* {A B : eqType} {alpha beta gamma : Rel A B}:
 (alpha » beta) ((alpha gamma) » (beta gamma)).

Proof.

apply inc_rpc.

rewrite -cap_assoc (@cap_comm _ _ alpha).

rewrite rpc_l.

$$(\alpha \Rightarrow \beta) \sqcap (\alpha \Rightarrow \gamma) = (\alpha \Rightarrow (\beta \sqcap \gamma)).$$

Proof.

$$(\alpha \Rightarrow \beta) \sqcap (\beta \Rightarrow \gamma) \sqsubseteq ((\alpha \sqcup \beta) \Rightarrow (\beta \sqcap \gamma)).$$

Proof.

52

CHAPTER 4. LIBRARY BASIC_LEMMAS

Lemma 69 (rpc_lemma4) *Let $\alpha, \beta, \gamma : A \rightarrow B$. Then,*

$$(\alpha \Rightarrow \beta) \sqcap (\beta \Rightarrow \gamma) \sqsubseteq (\alpha \Rightarrow \gamma).$$

Lemma *rpc_lemma4* {A B : eqType} {alpha beta gamma : Rel A B}:
 ((alpha » beta) (beta » gamma)) (alpha » gamma).

Proof.

apply (@inc_trans _ _ _ ((alpha beta) » (beta gamma))).

apply rpc_lemma3.

apply rpc_inc_compat.

apply cup_l.

apply cap_r.

Qed.

Lemma 70 (rpc_lemma5) *Let $\alpha, \beta, \gamma : A \rightarrow B$. Then,*

$$\alpha \Rightarrow (\beta \Rightarrow \gamma) = (\alpha \sqcap \beta) \Rightarrow \gamma.$$

Lemma *rpc_lemma5* {A B : eqType} {alpha beta gamma : Rel A B}:
 alpha » (beta » gamma) = (alpha beta) » gamma.

Proof.

apply inc_lower.

move => delta.

split; move => H.

apply inc_rpc.

rewrite -cap_assoc.

rewrite -inc_rpc -inc_rpc.

apply H.

rewrite inc_rpc inc_rpc.

rewrite cap_assoc.

apply inc_rpc.

apply H.

Qed.

Lemma 71 (rpc_lemma6) *Let $\alpha, \beta, \gamma : A \rightarrow B$. Then,*

$$\alpha \Rightarrow (\beta \Rightarrow \gamma) \sqsubseteq (\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma).$$

Lemma *rpc_lemma6* {A B : eqType} {alpha beta gamma : Rel A B}:
 (alpha » (beta » gamma)) ((alpha » beta) » (alpha » gamma)).

Proof.

rewrite inc_rpc inc_rpc.

rewrite cap_assoc (@cap_comm _ _ _ alpha).

```

rewrite rpc_l.
rewrite -cap_assoc (@cap_comm _ _ alpha).
rewrite rpc_l.
rewrite cap_assoc (@cap_comm _ _ beta).
rewrite rpc_l.
rewrite -cap_assoc.
apply cap_r.
Qed.

```

Lemma 72 (rpc_lemma7) *Let $\alpha, \beta, \gamma, \delta : A \rightarrow B$ and $\beta \sqsubseteq \alpha \sqsubseteq \gamma$. Then,*

$$(\alpha \sqcap \delta = \beta) \wedge (\alpha \sqcup \delta = \gamma) \Leftrightarrow (\gamma \sqsubseteq \alpha \sqcup (\alpha \Rightarrow \beta)) \wedge (\delta = \gamma \sqcap (\alpha \Rightarrow \beta)).$$

Lemma *rpc_lemma7* {A B : eqType} {alpha beta gamma delta : Rel A B}:
 beta alpha → alpha gamma → (alpha delta = beta ∧ alpha delta = gamma
 ↔ gamma (alpha (alpha » beta)) ∧ delta = gamma (alpha » beta)).

Proof.

```

move ⇒ H H0.
split; elim; move ⇒ H1 H2; split.
rewrite -H2.
apply cup_inc_compat_l.
apply inc_rpc.
rewrite cap_comm H1.
apply inc_refl.
rewrite -H2.
rewrite cap_cup_distr_r rpc_l.
assert (delta (alpha » beta)).
apply inc_rpc.
rewrite cap_comm H1.
apply inc_refl.
apply inc_def1 in H3.
rewrite -H3 -H1.
rewrite -cap_assoc cap_idem.
by [rewrite cap_comm cup_comm cup_cap_abs].
rewrite H2.
rewrite (@cap_comm _ _ gamma) -cap_assoc rpc_l.
apply inc_antisym.
apply (@inc_trans _ _ (beta gamma)).
apply cap_inc_compat_r.
apply cap_r.
apply cap_l.
move : (@inc_trans _ _ _ H H0) ⇒ H3.
apply inc_def1 in H.

```

```

apply inc_def1 in H3.
rewrite cap_comm in H.
rewrite -H -H3.
apply inc_refl.
rewrite H2.
rewrite cup_cap_distr_l.
apply inc_def2 in H0.
rewrite -H0.
apply inc_def1 in H1.
by [rewrite -H1].
Qed.

```

4.3 補関係に関する補題

Lemma 73 (complement_universal)

$$\nabla_{AB}^- = \phi_{AB}.$$

Lemma *complement_universal* $\{A\ B : eqType\}$: $A\ B\ ^\wedge = A\ B$.

Proof.

```

apply rpc_universal_alpha.

```

Qed.

Lemma 74 (complement_alpha_universal) *Let $\alpha : A \rightarrow B$. Then,*

$$\alpha^- = \nabla_{AB} \Leftrightarrow \alpha = \phi_{AB}.$$

Lemma *complement_alpha_universal* $\{A\ B : eqType\}\ \{\alpha : Rel\ A\ B\}$:
 $\alpha\ ^\wedge = A\ B \Leftrightarrow \alpha = A\ B$.

Proof.

```

split; move => H.
apply inc_antisym.
rewrite -(@cap_universal _ _ alpha) cap_comm.
apply inc_rpc.
rewrite -H.
apply inc_refl.
apply inc_empty_alpha.
apply inc_antisym.
apply inc_alpha_universal.
apply inc_rpc.
rewrite cap_comm cap_universal.
rewrite H.

```


apply *inc_refl*.

Qed.

Lemma 75 (complement_empty)

$$\phi_{AB}^- = \nabla_{AB}.$$

Lemma *complement_empty* {A B : eqType}: $\phi_{AB}^- = \nabla_{AB}$.

Proof.

by [apply *complement_alpha_universal*].

Qed.

Lemma 76 (complement_invol_inc) *Let $\alpha : A \rightarrow B$. Then,*

$$\alpha \sqsubseteq (\alpha^-)^-.$$

Lemma *complement_invol_inc* {A B : eqType} {alpha : Rel A B}: $\alpha \sqsubseteq (\alpha^-)^-$.

Proof.

apply *inc_rpc*.

rewrite *cap_comm*.

apply *inc_rpc*.

apply *inc_refl*.

Qed.

Lemma 77 (cap_complement_empty) *Let $\alpha : A \rightarrow B$. Then,*

$$\alpha \sqcap \alpha^- = \phi_{AB}.$$

Lemma *cap_complement_empty* {A B : eqType} {alpha : Rel A B}:

$$\alpha \sqcap \alpha^- = \phi_{AB}.$$

Proof.

apply *inc_antisym*.

rewrite *cap_comm*.

apply *inc_rpc*.

apply *inc_refl*.

apply *inc_empty_alpha*.

Qed.

Lemma 78 (complement_invol) *Let $\alpha : A \rightarrow B$. Then,*

$$(\alpha^-)^- = \alpha.$$

Lemma *complement_invol* {A B : eqType} {alpha : Rel A B}: $(\alpha^-)^- = \alpha$.

CHAPTER 4. LIBRARY BASIC_LEMMAS

Proof.

```
rewrite -(@cap_universal _ _ ((alpha ^) ^)).
rewrite -(@complement_classic _ _ alpha).
rewrite cap_cup_distr_l.
rewrite (@cap_comm _ _ (alpha ^)) cap_complement_empty.
rewrite cup_empty cap_comm.
apply Logic.eq_sym.
apply inc_def1.
apply complement_invol_inc.
```

Qed.

Lemma 79 (complement_move) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$\alpha = \beta^- \Leftrightarrow \alpha^- = \beta.$$

Lemma complement_move $\{A\ B : eqType\} \{alpha\ beta : Rel\ A\ B\}$:
 $alpha = beta^{\wedge} \Leftrightarrow alpha^{\wedge} = beta.$

Proof.

```
split; move => H.
by [rewrite H complement_invol].
by [rewrite -H complement_invol].
```

Qed.

Lemma 80 (contraposition) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$(\alpha \Rightarrow \beta) = (\beta^- \Rightarrow \alpha^-).$$

Lemma contraposition $\{A\ B : eqType\} \{alpha\ beta : Rel\ A\ B\}$:
 $alpha \gg beta = beta^{\wedge} \gg alpha^{\wedge}.$

Proof.

```
apply inc_antisym.
apply inc_rpc.
apply rpc_lemma4.
replace (alpha >> beta) with ((alpha ^) ^ >> (beta ^) ^).
apply inc_rpc.
apply rpc_lemma4.
by [rewrite complement_invol complement_invol].
```

Qed.

Lemma 81 (de_morgan1) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$(\alpha \sqcup \beta)^- = \alpha^- \sqcap \beta^-.$$

CHAPTER 4. LIBRARY BASIC_LEMMAS

Lemma *de_morgan1* {A B : eqType} {alpha beta : Rel A B}:
 $(\alpha \sqcap \beta)^\wedge = \alpha^\wedge \sqcap \beta^\wedge$.

Proof.

apply *inc_lower*.
 move \Rightarrow *gamma*.
 split; move \Rightarrow *H*.
 apply *inc_cap*.
 rewrite *inc_rpc inc_rpc*.
 apply *inc_cup*.
 rewrite *-cap_cup_distr_l*.
 apply *inc_rpc*.
 apply *H*.
 apply *inc_rpc*.
 rewrite *cap_cup_distr_l*.
 apply *inc_cup*.
 rewrite *-inc_rpc -inc_rpc*.
 apply *inc_cap*.
 apply *H*.

Qed.

Lemma 82 (de_morgan2) Let $\alpha, \beta : A \rightarrow B$. Then,

$$(\alpha \sqcap \beta)^\neg = \alpha^\neg \sqcup \beta^\neg.$$

Lemma *de_morgan2* {A B : eqType} {alpha beta : Rel A B}:
 $(\alpha \sqcap \beta)^\wedge = \alpha^\wedge \sqcap \beta^\wedge$.

Proof.

by [rewrite *-complement_move de_morgan1 complement_invol complement_invol*].

Qed.

Lemma 83 (cup_to_rpc) Let $\alpha, \beta : A \rightarrow B$. Then,

$$\alpha^\neg \sqcup \beta = (\alpha \Rightarrow \beta).$$

Lemma *cup_to_rpc* {A B : eqType} {alpha beta : Rel A B}:
 $\alpha^\wedge \sqcap \beta = \alpha \gg \beta$.

Proof.

apply *inc_antisym*.
 apply *inc_rpc*.
 rewrite *cap_cup_distr_r cap_comm*.
 rewrite *cap_complement_empty cup_comm cup_empty*.
 apply *cap_l*.
 rewrite *-(@cap_universal _ _ (alpha » beta)) cap_comm*.

CHAPTER 4. LIBRARY BASIC_LEMMAS

```
rewrite -(@complement_classic _ _ alpha).
rewrite cap_cup_distr_r cup_comm.
apply cup_inc_compat.
apply cap_l.
rewrite rpc_l.
apply cap_r.
Qed.
```

Lemma 84 (beta_contradiction) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$(\alpha \Rightarrow \beta) \sqcap (\alpha \Rightarrow \beta^-) = \alpha^-.$$

Lemma beta_contradiction $\{A B : eqType\} \{alpha\ beta : Rel\ A\ B\}$:
 $(alpha \gg beta) \quad (alpha \gg beta^-) = alpha^-.$

Proof.

```
rewrite -cup_to_rpc -cup_to_rpc.
rewrite -cup_cap_distr_l.
by [rewrite cap_complement_empty cup_empty].
Qed.
```

4.4 Bool 代数に関する補題

Lemma 85 (bool_lemma1) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$\alpha \sqsubseteq \beta \Leftrightarrow \nabla_{AB} = \alpha^- \sqcup \beta.$$

Lemma bool_lemma1 $\{A B : eqType\} \{alpha\ beta : Rel\ A\ B\}$:
 $alpha \sqsubseteq beta \Leftrightarrow A\ B = alpha^- \sqcup beta.$

Proof.

```
split; move => H.
apply inc_antisym.
rewrite -(@complement_classic _ _ alpha) cup_comm.
apply cup_inc_compat_l.
apply H.
apply inc_alpha_universal.
apply inc_def3.
rewrite H.
apply (Logic.eq_sym cup_to_rpc).
Qed.
```

CHAPTER 4. LIBRARY BASIC_LEMMAS

Lemma 86 (bool_lemma2) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$\alpha \sqsubseteq \beta \Leftrightarrow \alpha \sqcap \beta^- = \phi_{AB}.$$

Lemma *bool_lemma2* {*A B : eqType*} {*alpha beta : Rel A B*}:

alpha beta \leftrightarrow *alpha beta* \wedge = *A B*.

Proof.

split; move \Rightarrow *H*.

rewrite -(@cap_universal _ _ (*alpha beta* \wedge)).

apply *bool_lemma1* in *H*.

rewrite *H*.

rewrite *cap_cup_distr_l*.

rewrite (@cap_comm _ _ *alpha*) *cap_assoc cap_complement_empty cap_empty*.

rewrite *cap_comm -cap_assoc cap_complement_empty cap_comm cap_empty*.

by [rewrite *cup_empty*].

rewrite -(@cap_universal _ _ *alpha*).

rewrite -(@complement_classic _ _ *beta*).

rewrite *cap_cup_distr_l*.

rewrite *H cup_empty*.

apply *cap_r*.

Qed.

Lemma 87 (bool_lemma3) *Let $\alpha, \beta, \gamma : A \rightarrow B$. Then,*

$$\alpha \sqsubseteq \beta \sqcup \gamma \Leftrightarrow \alpha \sqcap \beta^- \sqsubseteq \gamma.$$

Lemma *bool_lemma3* {*A B : eqType*} {*alpha beta gamma : Rel A B*}:

alpha (*beta gamma*) \leftrightarrow (*alpha beta* \wedge) *gamma*.

Proof.

split; move \Rightarrow *H*.

apply (@inc_trans _ _ _ ((*beta gamma*) *beta* \wedge)).

apply *cap_inc_compat_r*.

apply *H*.

rewrite *cap_cup_distr_r*.

rewrite *cap_complement_empty cup_comm cup_empty*.

apply *cap_l*.

apply (@inc_trans _ _ _ (*beta* (*alpha beta* \wedge))).

rewrite *cup_cap_distr_l*.

rewrite *complement_classic cap_universal*.

apply *cup_r*.

apply *cup_inc_compat_l*.

apply *H*.

Qed.

CHAPTER 4. LIBRARY BASIC_LEMMAS

Lemma 88 (bool_lemma4) *Let $\alpha, \beta, \gamma : A \rightarrow B$. Then,*

$$\alpha \sqsubseteq \beta \sqcup \gamma \Leftrightarrow \beta^- \sqsubseteq \alpha^- \sqcup \gamma.$$

Lemma *bool_lemma4* { $A B : eqType$ } { $\alpha \beta \gamma : Rel A B$ }:
 $\alpha \sqsubseteq (\beta \sqcup \gamma) \Leftrightarrow \beta^- \sqsubseteq (\alpha^- \sqcup \gamma).$

Proof.

rewrite *bool_lemma3*.

rewrite *cap_comm*.

apply *iff_sym*.

replace ($\beta^- \sqsubseteq \alpha$) with ($\beta^- \sqsubseteq (\alpha^- \sqcup \gamma)$).

apply *bool_lemma3*.

by [rewrite *complement_invol*].

Qed.

Lemma 89 (bool_lemma5) *Let $\alpha, \beta, \gamma : A \rightarrow B$. Then,*

$$\alpha \sqsubseteq \beta \sqcup \gamma \Leftrightarrow \nabla_{AB} = \alpha^- \sqcup \beta \sqcup \gamma.$$

Lemma *bool_lemma5* { $A B : eqType$ } { $\alpha \beta \gamma : Rel A B$ }:
 $\alpha \sqsubseteq (\beta \sqcup \gamma) \Leftrightarrow A B = (\alpha^- \sqcup \beta) \sqcup \gamma.$

Proof.

rewrite *bool_lemma1*.

by [rewrite *cup_assoc*].

Qed.

Chapter 5

Library **Relation_Properties**

```
Require Import Basic_Notations.  
Require Import Basic_Lemmas.  
Require Import Logic.FunctionalExtensionality.  
Require Import Logic.Classical_Prop.
```

5.1 関係計算の基本的な性質

Lemma 90 (RelAB_unique)

$$\phi_{AB} = \nabla_{AB} \Leftrightarrow \forall \alpha, \beta : A \rightarrow B, \alpha = \beta.$$

Lemma *RelAB_unique* {A B : eqType}:

$$A B = A B \Leftrightarrow (\forall \text{ alpha beta} : \text{Rel } A B, \text{ alpha} = \text{ beta}).$$

Proof.

split; move \Rightarrow *H*.

move \Rightarrow *alpha beta*.

replace *beta* with (*A B*).

apply *inc_antisym*.

rewrite *H*.

apply *inc_alpha_universal*.

apply *inc_empty_alpha*.

apply *inc_antisym*.

apply *inc_empty_alpha*.

rewrite *H*.

apply *inc_alpha_universal*.

apply *H*.

Qed.

Lemma 91 (either_empty)

$$\phi_{AB} = \nabla_{AB} \Leftrightarrow A = \emptyset \vee B = \emptyset.$$

Lemma *either_empty* {*A B : eqType*}: $A \ B = \ A \ B \Leftrightarrow (A \rightarrow \text{False}) \vee (B \rightarrow \text{False})$.

Proof.

rewrite *RelAB_unique*.

split; move $\Rightarrow H$.

case (*classic* ($\exists _ : A, \text{True}$)).

elim $\Rightarrow a \ H0$.

right.

move $\Rightarrow b$.

remember (*fun* ($_ : A$) ($_ : B$) $\Rightarrow \text{True}$) **as** *T*.

remember (*fun* ($_ : A$) ($_ : B$) $\Rightarrow \text{False}$) **as** *F*.

move : (*H T F*) $\Rightarrow H1$.

assert (*T a b = F a b*).

by [rewrite *H1*].

rewrite *HeqT HeqF* in *H2*.

rewrite -*H2*.

apply *I*.

move $\Rightarrow H0$.

left.

move $\Rightarrow a$.

apply *H0*.

$\exists a$.

apply *I*.

move $\Rightarrow \text{alpha beta}$.

assert ($A \rightarrow B \rightarrow \text{False}$).

move $\Rightarrow a \ b$.

case *H*; move $\Rightarrow H0$.

apply (*H0 a*).

apply (*H0 b*).

apply *functional_extensionality*.

move $\Rightarrow a$.

apply *functional_extensionality*.

move $\Rightarrow b$.

apply *False_ind*.

apply (*H0 a b*).

Qed.

Lemma 92 (unit_empty_not_universal)

$$\phi_{II} \neq \nabla_{II}.$$

Lemma *unit_empty_not_universal* : $\phi_{II} \neq \nabla_{II}$.

Proof.

move $\Rightarrow H$.

apply *either_empty* in H .

case H ; move $\Rightarrow H0$.

apply ($H0\ tt$).

apply ($H0\ tt$).

Qed.

Lemma 93 (unit_empty_or_universal) *Let $\alpha : I \rightarrow I$. Then,*

$$\alpha = \phi_{II} \vee \alpha = \nabla_{II}.$$

Lemma *unit_empty_or_universal* { $\alpha : Rel\ i\ i$ } : $\alpha = \phi_{II} \vee \alpha = \nabla_{II}$.

Proof.

assert ($\forall\ \text{beta} : Rel\ i\ i, \text{beta} = (\text{fun } (-) : i \Rightarrow True) \vee \text{beta} = (\text{fun } (-) : i \Rightarrow False)$).

move $\Rightarrow \text{beta}$.

case (*classic* ($\text{beta}\ tt\ tt$)); move $\Rightarrow H$.

left.

apply *functional_extensionality*.

induction x .

apply *functional_extensionality*.

induction x .

apply *prop_extensionality_ok*.

split; move $\Rightarrow H0$.

apply I .

apply H .

right.

apply *functional_extensionality*.

induction x .

apply *functional_extensionality*.

induction x .

apply *prop_extensionality_ok*.

split.

apply H .

apply *False_ind*.

assert ($(\text{fun } (-) : i \Rightarrow True) \neq (\text{fun } (-) : i \Rightarrow False)$).

move $\Rightarrow H0$.

remember ($\text{fun } (-) : i \Rightarrow True$) as T .

```

remember (fun _ _ : i ⇒ False) as F.
assert (T tt tt = F tt tt).
by [rewrite H0].
rewrite HeqT HeqF in H1.
rewrite -H1.
apply I.
case (H ( i i)); move ⇒ H1.
case (H ( i i)); move ⇒ H2.
apply False_ind.
apply unit_empty_not_universal.
by [rewrite H1 H2].
case (H alpha); move ⇒ H3.
left.
by [rewrite H3 H1].
right.
by [rewrite H3 H2].
case (H ( i i)); move ⇒ H2.
case (H alpha); move ⇒ H3.
right.
by [rewrite H3 H2].
left.
by [rewrite H3 H1].
apply False_ind.
apply unit_empty_not_universal.
by [rewrite H1 H2].
Qed.

```

Lemma 94 (unit_identity_is_universal)

$$id_I = \nabla_{II}.$$

Lemma *unit_identity_is_universal* : $Id\ i = \quad i\ i$.

Proof.

```

case (@unit_empty_or_universal (Id i)); move ⇒ H.
apply False_ind.
assert (Id i ( i i # i i)).
rewrite H.
apply inc_empty_alpha.
apply inc_residual in H0.
rewrite inv_invol_comp_id_r in H0.
apply unit_empty_not_universal.
apply inc_antisym.
apply inc_empty_alpha.

```

apply *H0*.

apply *H*.

Qed.

Lemma 95 (unit_identity_not_empty)

$$id_I \neq \phi_{II}.$$

Lemma *unit_identity_not_empty* : $Id\ i \neq \phi\ i\ i$.

Proof.

move \Rightarrow *H*.

apply *unit_empty_not_universal*.

rewrite *-H*.

apply *unit_identity_is_universal*.

Qed.

Lemma 96 (cupL_emptyset) *Let $\alpha_\lambda : A \rightarrow B$ and $E = \emptyset$. Then,*

$$\sqcup_{\lambda \in E} \alpha_\lambda = \phi_{AB}.$$

Lemma *cupL_emptyset* {*A B L : eqType*} {*alpha_L : L → Rel A B*}:

$(L \rightarrow False) \rightarrow \sqcup_{\lambda \in E} \alpha_\lambda = \phi_{AB}$.

Proof.

move \Rightarrow *H*.

apply *inc_antisym*.

apply *inc_cupL*.

move \Rightarrow *l*.

apply *False_ind*.

apply (*H l*).

apply *inc_empty_alpha*.

Qed.

Lemma 97 (capL_emptyset) *Let $\alpha_\lambda : A \rightarrow B$ and $E = \emptyset$. Then,*

$$\sqcap_{\lambda \in E} \alpha_\lambda = \nabla_{AB}.$$

Lemma *capL_emptyset* {*A B L : eqType*} {*alpha_L : L → Rel A B*}:

$(L \rightarrow False) \rightarrow \sqcap_{\lambda \in E} \alpha_\lambda = \nabla_{AB}$.

Proof.

move \Rightarrow *H*.

apply *inc_antisym*.

apply *inc_alpha_universal*.

apply *inc_capL*.

move $\Rightarrow l$.
 apply *False_ind*.
 apply (*H l*).
 Qed.

Lemma 98 (cap_cupL_distr_l) *Let $\alpha, \beta_\lambda : A \rightarrow B$. Then,*

$$\alpha \sqcap (\sqcup_{\lambda \in \Lambda} \beta_\lambda) = \sqcup_{\lambda \in \Lambda} (\alpha \sqcap \beta_\lambda).$$

Lemma cap_cupL_distr_l

$\{A\ B\ L : \text{eqType}\} \{ \alpha : \text{Rel } A\ B \} \{ \beta_{\text{L}} : L \rightarrow \text{Rel } A\ B \} :$
 $\alpha \sqcap (\sqcup_{\text{L}} \beta_{\text{L}}) = \sqcup_{\text{L}} (\text{fun } l : L \Rightarrow \alpha \sqcap \beta_{\text{L}}\ l).$

Proof.

apply *inc_upper*.
 move $\Rightarrow \text{gamma}$.
 split; move $\Rightarrow H$.
 apply *inc_cupL*.
 move $\Rightarrow l$.
 apply (@*inc_trans* _ _ _ ($\alpha \sqcap \beta_{\text{L}}\ l$)).
 apply *cap_inc_compat_l*.
 apply *inc_cupL*.
 apply *inc_refl*.
 apply *H*.
 assert ($\forall l : L, (\alpha \sqcap \beta_{\text{L}}\ l) \sqsubseteq \text{gamma}$).
 apply *inc_cupL*.
 apply *H*.
 assert ($\forall l : L, \beta_{\text{L}}\ l \sqsubseteq (\alpha \sqcap \beta_{\text{L}}\ l)$).
 move $\Rightarrow l$.
 rewrite *inc_rpc_cap_comm*.
 apply *H0*.
 rewrite *cap_comm_inc_rpc*.
 apply *inc_cupL*.
 apply *H1*.
 Qed.

Lemma 99 (cap_cupL_distr_r) *Let $\alpha_\lambda, \beta : A \rightarrow B$. Then,*

$$(\sqcup_{\lambda \in \Lambda} \alpha_\lambda) \sqcap \beta = \sqcup_{\lambda \in \Lambda} (\alpha_\lambda \sqcap \beta).$$

Lemma cap_cupL_distr_r

$\{A\ B\ L : \text{eqType}\} \{ \beta : \text{Rel } A\ B \} \{ \alpha_{\text{L}} : L \rightarrow \text{Rel } A\ B \} :$
 $(\sqcup_{\text{L}} \alpha_{\text{L}}) \sqcap \beta = \sqcup_{\text{L}} (\text{fun } l : L \Rightarrow \alpha_{\text{L}}\ l \sqcap \beta).$

Proof.

```

rewrite cap_comm.
replace (fun l : L => alpha_L l    beta) with (fun l : L => beta    alpha_L l).
apply cap_cupL_distr_l.
apply functional_extensionality.
move => l.
by [rewrite cap_comm].
Qed.

```

Lemma 100 (cup_capL_distr_l) *Let $\alpha, \beta_\lambda : A \rightarrow B$. Then,*

$$\alpha \sqcup (\prod_{\lambda \in \Lambda} \beta_\lambda) = \prod_{\lambda \in \Lambda} (\alpha \sqcup \beta_\lambda).$$

Lemma cup_capL_distr_l

$\{A\ B\ L : \text{eqType}\} \{alpha : \text{Rel}\ A\ B\} \{beta_L : L \rightarrow \text{Rel}\ A\ B\} :$
 $alpha \sqcup (_ \beta_L) = _ (fun\ l : L \Rightarrow alpha \sqcup beta_L\ l).$

Proof.

```

apply inc_lower.
move => gamma.
split; move => H.
apply inc_capL.
move => l.
apply (@inc_trans _ _ _ (alpha _ beta_L)).
apply H.
apply cup_inc_compat_l.
apply inc_capL.
apply inc_refl.
rewrite bool_lemma3.
assert (forall l : L, gamma (alpha beta_L l)).
apply inc_capL.
apply H.
apply inc_capL.
move => l.
rewrite -bool_lemma3.
apply H0.
Qed.

```

Lemma 101 (cup_capL_distr_r) *Let $\alpha_\lambda, \beta : A \rightarrow B$. Then,*

$$(\prod_{\lambda \in \Lambda} \alpha_\lambda) \sqcup \beta = \prod_{\lambda \in \Lambda} (\alpha_\lambda \sqcup \beta).$$

Lemma cup_capL_distr_r

$\{A\ B\ L : \text{eqType}\} \{beta : \text{Rel}\ A\ B\} \{alpha_L : L \rightarrow \text{Rel}\ A\ B\} :$
 $(_ \alpha_L) \sqcup beta = _ (fun\ l : L \Rightarrow alpha_L\ l \sqcup beta).$

Proof.

```

rewrite cup_comm.
replace (fun l : L => alpha_L l    beta) with (fun l : L => beta    alpha_L l).
apply cup_capL_distr_l.
apply functional_extensionality.
move => l.
by [rewrite cup_comm].
Qed.
    
```

Lemma 102 (de_morgan3) *Let $\alpha_\lambda : A \rightarrow B$. Then,*

$$(\bigsqcup_{\lambda \in \Lambda} \alpha_\lambda)^- = (\prod_{\lambda \in \Lambda} \alpha_\lambda^-).$$

Lemma de_morgan3

$\{A\ B\ L : \text{eqType}\} \{ \text{alpha_L} : L \rightarrow \text{Rel}\ A\ B \} :$
 $(\ _ \ \text{alpha_L})^\wedge = \ _ \ (\text{fun } l : L \Rightarrow \text{alpha_L } l^\wedge).$

Proof.

```

apply inc_lower.
move => gamma.
rewrite inc_capL.
split; move => H.
move => l.
rewrite bool_lemma1 -de_morgan2 complement_move complement_universal.
apply bool_lemma2 in H.
apply inc_antisym.
apply inc_empty_alpha.
rewrite -H complement_invol.
apply cap_inc_compat_l.
apply inc_cupL.
apply inc_refl.
rewrite bool_lemma2 complement_invol.
rewrite cap_cupL_distr_l.
replace (fun l : L => gamma    alpha_L l) with (fun l : L =>    A B).
apply inc_antisym.
apply inc_cupL.
move => l.
apply inc_refl.
apply inc_empty_alpha.
apply functional_extensionality.
move => l.
rewrite -(@complement_invol _ _ (alpha_L l)).
apply Logic.eq_sym.
rewrite -bool_lemma2.
    
```

apply *H*.

Qed.

Lemma 103 (de_morgan4) *Let $\alpha_\lambda : A \rightarrow B$. Then,*

$$(\bigwedge_{\lambda \in \Lambda} \alpha_\lambda)^- = (\bigvee_{\lambda \in \Lambda} \alpha_\lambda^-).$$

Lemma de_morgan4

$\{A\ B\ L : eqType\} \{alpha_L : L \rightarrow Rel\ A\ B\}:$
 $(_ \alpha_L)^\wedge = _ (\text{fun } l : L \Rightarrow alpha_L\ l^\wedge).$

Proof.

rewrite -complement_move de_morgan3.

replace (fun *l* : *L* \Rightarrow (*alpha_L l*[^])[^]) with *alpha_L*.

by [].

apply functional_extensionality.

move \Rightarrow *l*.

by [rewrite complement_invol].

Qed.

Lemma 104 (cup_to_cupL, cap_to_capL) *We can prove \sqcup and \sqcap lemmas as $\sqcup_{\lambda \in \Lambda}$ and $\sqcap_{\lambda \in \Lambda}$.*

Lemma cup_to_cupL $\{A\ B : eqType\} \{alpha\ beta : Rel\ A\ B\}:$

$(alpha\ \beta) = _ (\text{fun } b : bool_eqType \Rightarrow \text{if } b \text{ then } alpha \text{ else } beta).$

Proof.

apply inc_upper.

move \Rightarrow *gamma*.

split; move \Rightarrow *H*.

apply inc_cupL.

apply inc_cup in *H*.

induction *l*.

apply *H*.

apply *H*.

apply inc_cup.

assert ($\forall b : bool_eqType, (\text{fun } b : bool_eqType \Rightarrow \text{if } b \text{ then } alpha \text{ else } beta)\ b\ \gamma$).

apply inc_cupL.

apply *H*.

split.

apply (*H0 true*).

apply (*H0 false*).

Qed.

Lemma cap_to_capL $\{A\ B : eqType\} \{alpha\ beta : Rel\ A\ B\}:$

```

(alpha  beta) =  _ (fun b : bool_eqType => if b then alpha else beta).
Proof.
apply inc_lower.
move => gamma.
split; move => H.
apply inc_capL.
apply inc_cap in H.
induction l.
apply H.
apply H.
apply inc_cap.
assert (forall b : bool_eqType, gamma  (fun b : bool_eqType => if b then alpha else beta)
b).
apply inc_capL.
apply H.
split.
apply (H0 true).
apply (H0 false).
Qed.

```

5.2 comp_inc_compat と派生補題

Lemma 105 (comp_inc_compat_ab_ab') *Let $\alpha : A \rightarrow B$ and $\beta, \beta' : B \rightarrow C$. Then,*

$$\beta \sqsubseteq \beta' \Rightarrow \alpha \cdot \beta \sqsubseteq \alpha \cdot \beta'.$$

Lemma comp_inc_compat_ab_ab'
 $\{A\ B\ C : eqType\} \{alpha : Rel\ A\ B\} \{beta\ beta' : Rel\ B\ C\} :$
 $beta\ beta' \rightarrow (alpha \cdot beta) \sqsubseteq (alpha \cdot beta').$
Proof.
move => H.
replace (alpha beta) with ((alpha #) # beta).
apply inc_residual.
apply (@inc_trans _ _ beta').
apply H.
apply inc_residual.
rewrite inv_invol.
apply inc_refl.
by [rewrite inv_invol].
Qed.

CHAPTER 5. LIBRARY RELATION_PROPERTIES

Lemma 106 (comp_inc_compat_ab_a'b) *Let $\alpha, \alpha' : A \rightarrow B$ and $\beta : B \rightarrow C$. Then,*

$$\alpha \sqsubseteq \alpha' \Rightarrow \alpha \cdot \beta \sqsubseteq \alpha' \cdot \beta.$$

Lemma *comp_inc_compat_ab_a'b*

$\{A\ B\ C : eqType\} \{alpha\ alpha' : Rel\ A\ B\} \{beta : Rel\ B\ C\} :$
 $alpha\ alpha' \rightarrow (alpha \cdot beta) \quad (alpha' \cdot beta).$

Proof.

move $\Rightarrow H$.
 rewrite $-(@inv_involver_ (alpha \cdot beta))$.
 rewrite $-(@inv_involver_ (alpha' \cdot beta))$.
 apply *inc_inv*.
 rewrite *comp_inv comp_inv*.
 apply *comp_inc_compat_ab_ab'*.
 apply *inc_inv*.
 apply *H*.
 Qed.

Lemma 107 (comp_inc_compat) *Let $\alpha, \alpha' : A \rightarrow B$ and $\beta, \beta' : B \rightarrow C$. Then,*

$$\alpha \sqsubseteq \alpha' \wedge \beta \sqsubseteq \beta' \Rightarrow \alpha \cdot \beta \sqsubseteq \alpha' \cdot \beta'.$$

Lemma *comp_inc_compat*

$\{A\ B\ C : eqType\} \{alpha\ alpha' : Rel\ A\ B\} \{beta\ beta' : Rel\ B\ C\} :$
 $alpha\ alpha' \rightarrow beta\ beta' \rightarrow (alpha \cdot beta) \quad (alpha' \cdot beta').$

Proof.

move $\Rightarrow H\ H0$.
 apply $(@inc_trans_ (alpha' \cdot beta))$.
 apply $(@comp_inc_compat_ab_a'b_ H)$.
 apply $(@comp_inc_compat_ab_ab'_ H0)$.
 Qed.

Lemma 108 (comp_inc_compat_ab_a) *Let $\alpha : A \rightarrow B$ and $\beta : B \rightarrow B$. Then,*

$$\beta \sqsubseteq id_B \Rightarrow \alpha \cdot \beta \sqsubseteq \alpha.$$

Lemma *comp_inc_compat_ab_a* $\{A\ B : eqType\} \{alpha : Rel\ A\ B\} \{beta : Rel\ B\ B\} :$
 $beta\ Id\ B \rightarrow (alpha \cdot beta) \quad alpha.$

Proof.

move $\Rightarrow H$.
 move : $(@comp_inc_compat_ab_ab'_ alpha_ H) \Rightarrow H0$.
 rewrite *comp_id_r* in *H0*.
 apply *H0*.

Qed.

Lemma 109 (comp_inc_compat_a_ab) *Let $\alpha : A \rightarrow B$ and $\beta : B \rightarrow B$. Then,*

$$id_B \sqsubseteq \beta \Rightarrow \beta \sqsubseteq \alpha \cdot \beta.$$

Lemma comp_inc_compat_a_ab $\{A\ B : eqType\} \{alpha : Rel\ A\ B\} \{beta : Rel\ B\ B\} :$
 $Id\ B \quad beta \rightarrow alpha \quad (alpha \cdot beta).$

Proof.

move $\Rightarrow H$.

move : ($@comp_inc_compat_ab_ab' _ _ _ alpha _ _ H$) $\Rightarrow H0$.

rewrite $comp_id_r$ in $H0$.

apply $H0$.

Qed.

Lemma 110 (comp_inc_compat_ab_b) *Let $\alpha : A \rightarrow A$ and $\beta : A \rightarrow B$. Then,*

$$\alpha \sqsubseteq id_A \Rightarrow \alpha \cdot \beta \sqsubseteq \beta.$$

Lemma comp_inc_compat_ab_b $\{A\ B : eqType\} \{alpha : Rel\ A\ A\} \{beta : Rel\ A\ B\} :$
 $alpha \quad Id\ A \rightarrow (alpha \cdot beta) \quad beta.$

Proof.

move $\Rightarrow H$.

move : ($@comp_inc_compat_ab_a'b _ _ _ _ beta\ H$) $\Rightarrow H0$.

rewrite $comp_id_l$ in $H0$.

apply $H0$.

Qed.

Lemma 111 (comp_inc_compat_b_ab) *Let $\alpha : A \rightarrow A$ and $\beta : A \rightarrow B$. Then,*

$$id_A \sqsubseteq \alpha \Rightarrow \beta \sqsubseteq \alpha \cdot \beta.$$

Lemma comp_inc_compat_b_ab $\{A\ B : eqType\} \{alpha : Rel\ A\ A\} \{beta : Rel\ A\ B\} :$
 $Id\ A \quad alpha \rightarrow beta \quad (alpha \cdot beta).$

Proof.

move $\Rightarrow H$.

move : ($@comp_inc_compat_ab_a'b _ _ _ _ beta\ H$) $\Rightarrow H0$.

rewrite $comp_id_l$ in $H0$.

apply $H0$.

Qed.

5.3 逆関係に関する補題

Lemma 112 (inv_move) *Let $\alpha : A \rightarrow B$ and $\beta : B \rightarrow A$. Then,*

$$\alpha = \beta^\# \Leftrightarrow \alpha^\# = \beta.$$

Lemma inv_move $\{A\ B : eqType\} \{alpha : Rel\ A\ B\} \{beta : Rel\ B\ A\} :$
 $alpha = beta^\# \leftrightarrow alpha^\# = beta.$

Proof.

split; move $\Rightarrow H$.

by [rewrite $H\ inv_invol$].

by [rewrite $-H\ inv_invol$].

Qed.

Lemma 113 (comp_inv_inv) *Let $\alpha : A \rightarrow B$ and $\beta : B \rightarrow C$. Then,*

$$\alpha \cdot \beta = (\beta^\# \cdot \alpha^\#)^\#.$$

Lemma comp_inv_inv $\{A\ B\ C : eqType\} \{alpha : Rel\ A\ B\} \{beta : Rel\ B\ C\} :$
 $alpha \cdot beta = (beta^\# \cdot alpha^\#)^\#.$

Proof.

apply *inv_move*.

apply *comp_inv*.

Qed.

Lemma 114 (inv_inc_move) *Let $\alpha : A \rightarrow B$ and $\beta : B \rightarrow A$. Then,*

$$\alpha \sqsubseteq \beta^\# \Leftrightarrow \alpha^\# \sqsubseteq \beta.$$

Lemma inv_inc_move $\{A\ B : eqType\} \{alpha : Rel\ A\ B\} \{beta : Rel\ B\ A\} :$
 $alpha \sqsubseteq beta^\# \leftrightarrow alpha^\# \sqsubseteq beta.$

Proof.

split; move $\Rightarrow H$.

rewrite $-(@inv_invol\ _\ _\ beta)$.

apply *inc_inv*.

apply H .

rewrite $-(@inv_invol\ _\ _\ alpha)$.

apply *inc_inv*.

apply H .

Qed.

Lemma 115 (inv_invol2) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$\alpha^\# = \beta^\# \Rightarrow \alpha = \beta.$$

Lemma *inv_invol2* { $A B : eqType$ } { $\alpha \beta : Rel A B$ }:

$\alpha \# = \beta \# \rightarrow \alpha = \beta$.

Proof.

move $\Rightarrow H$.

rewrite $-(@inv_invol _ _ \alpha)$ $-(@inv_invol _ _ \beta)$.

apply f_equal.

apply H .

Qed.

Lemma 116 (inv_inc_invol) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$\alpha^\# \sqsubseteq \beta^\# \Rightarrow \alpha \sqsubseteq \beta.$$

Lemma *inv_inc_invol* { $A B : eqType$ } { $\alpha \beta : Rel A B$ }:

$\alpha \# \sqsubseteq \beta \# \rightarrow \alpha \sqsubseteq \beta$.

Proof.

move $\Rightarrow H$.

rewrite $-(@inv_invol _ _ \alpha)$ $-(@inv_invol _ _ \beta)$.

apply *inc_inv*.

apply H .

Qed.

Lemma 117 (inv_cupL_distr, inv_cup_distr) *Let $\alpha_\lambda : A \rightarrow B$. Then,*

$$(\bigsqcup_{\lambda \in \Lambda} \alpha_\lambda)^\# = (\bigsqcup_{\lambda \in \Lambda} \alpha_\lambda^\#).$$

Lemma *inv_cupL_distr* { $A B L : eqType$ } { $\alpha_L : L \rightarrow Rel A B$ }:

$(_ _ \alpha_L) \# = (_ _ (\text{fun } l : L \Rightarrow \alpha_L l \#))$.

Proof.

apply *inc_antisym*.

rewrite *inv_inc_move*.

apply *inc_cupL*.

assert $(\forall l : L, \alpha_L l \# _ _ (\text{fun } l0 : L \Rightarrow \alpha_L l0 \#))$.

apply *inc_cupL*.

apply *inc_refl*.

move $\Rightarrow l$.

rewrite *inv_inc_move*.

apply H .

apply *inc_cupL*.

move \Rightarrow *l*.

apply *inc_inv*.

apply *inc_cupL*.

apply *inc_refl*.

Qed.

Lemma *inv_cup_distr* {*A B* : *eqType*} {*alpha beta* : *Rel A B*}:
 (*alpha* *beta*) # = *alpha* # *beta* #.

Proof.

rewrite *cup_to_cupL cup_to_cupL*.

rewrite *inv_cupL_distr*.

apply *f_equal*.

apply *functional_extensionality*.

induction *x*.

by [].

by [].

Qed.

Lemma 118 (*inv_capL_distr*, *inv_cap_distr*) *Let* $\alpha_\lambda : A \rightarrow B$. *Then,*

$$(\prod_{\lambda \in \Lambda} \alpha_\lambda)^\# = (\prod_{\lambda \in \Lambda} \alpha_\lambda^\#).$$

Lemma *inv_capL_distr* {*A B L* : *eqType*} {*alpha_L* : *L* \rightarrow *Rel A B*}:
 (*alpha_L*) # = ((fun *l* : *L* \Rightarrow *alpha_L l* #)).

Proof.

apply *inc_antisym*.

apply *inc_capL*.

move \Rightarrow *l*.

apply *inc_inv*.

apply *inc_capL*.

apply *inc_refl*.

rewrite *inv_inc_move*.

apply *inc_capL*.

assert (\forall *l* : *L*, (fun *l0* : *L* \Rightarrow *alpha_L l0* #) *alpha_L l* #).

apply *inc_capL*.

apply *inc_refl*.

move \Rightarrow *l*.

rewrite *inv_inc_move*.

apply *H*.

Qed.

Lemma *inv_cap_distr* {*A B* : *eqType*} {*alpha beta* : *Rel A B*}:
 (*alpha* *beta*) # = *alpha* # *beta* #.

Proof.

```

rewrite cap_to_capL cap_to_capL.
rewrite inv_capL_distr.
apply f_equal.
apply functional_extensionality.
induction x.
by [].
by [].
Qed.

```

Lemma 119 (rpc_inv_distr) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$(\alpha \Rightarrow \beta)^\# = \alpha^\# \Rightarrow \beta^\#.$$

Lemma *rpc_inv_distr* $\{A\ B : \text{eqType}\} \{ \text{alpha}\ \text{beta} : \text{Rel}\ A\ B \}$:
 $(\text{alpha} \gg \text{beta}) \# = \text{alpha} \# \gg \text{beta} \#.$

Proof.

```

apply inc_lower.
move => gamma.
split; move => H.
apply inc_rpc.
rewrite inv_inc_move inv_cap_distr inv_invol.
rewrite -inc_rpc -inv_inc_move.
apply H.
rewrite inv_inc_move inc_rpc.
rewrite -(@inv_invol _ _ alpha) -inv_cap_distr -inv_inc_move.
apply inc_rpc.
apply H.
Qed.

```

Lemma 120 (inv_empty)

$$\phi_{AB}^\# = \phi_{BA}.$$

Lemma *inv_empty* $\{A\ B : \text{eqType}\}$: $A\ B \# = B\ A.$

Proof.

```

apply inc_antisym.
rewrite -inv_inc_move.
apply inc_empty_alpha.
apply inc_empty_alpha.
Qed.

```

Lemma 121 (inv_universal)

$$\nabla_{AB}^\# = \nabla_{BA}.$$

Lemma *inv_universal* {*A B : eqType*}: $A B \# = B A$.

Proof.

apply *inc_antisym*.

apply *inc_alpha_universal*.

rewrite *inv_inc_move*.

apply *inc_alpha_universal*.

Qed.

Lemma 122 (inv_id)

$$id_A^\# = id_A.$$

Lemma *inv_id* {*A : eqType*}: $(Id A) \# = Id A$.

Proof.

replace $(Id A \#)$ with $((Id A \#) \# \cdot Id A \#)$.

by [rewrite *-comp_inv comp_id_l inv_invol*].

by [rewrite *inv_invol comp_id_l*].

Qed.

Lemma 123 (inv_complement) *Let $\alpha : A \rightarrow B$. Then,*

$$(\alpha^-)^\# = (\alpha^\#)^-.$$

Lemma *inv_complement* {*A B : eqType*} {*alpha : Rel A B*}: $(alpha \wedge) \# = (alpha \#) \wedge$.

Proof.

apply *inc_antisym*.

apply *inc_rpc*.

rewrite *-inv_cap_distr*.

rewrite *cap_comm -inv_inc_move inv_empty*.

rewrite *cap_complement_empty*.

apply *inc_refl*.

rewrite *inv_inc_move*.

apply *inc_rpc*.

replace $((alpha \#) \wedge) \# \quad alpha$ with $((alpha \#) \wedge) \# \quad (alpha \#) \#$.

rewrite *-inv_cap_distr*.

rewrite *cap_comm -inv_inc_move inv_empty*.

rewrite *cap_complement_empty*.

apply *inc_refl*.

by [rewrite *inv_invol*].

Qed.

Lemma 124 (inv_difference_distr) *Let $\alpha, \beta : A \rightarrow B$. Then,*

$$(\alpha - \beta)^\# = \alpha^\# - \beta^\#.$$

Lemma *inv_difference_distr* { $A B : eqType$ } { $\alpha \beta : Rel A B$ }:
 $(\alpha - \beta) \# = \alpha \# - \beta \#$.

Proof.

rewrite *inv_cap_distr*.

by [rewrite *inv_complement*].

Qed.

5.4 合成に関する補題

Lemma 125 (comp_cupL_distr_l, comp_cup_distr_l) *Let $\alpha : A \rightarrow B$ and $\beta_\lambda : B \rightarrow C$. Then,*

$$\alpha \cdot (\sqcup_{\lambda \in \Lambda} \beta_\lambda) = \sqcup_{\lambda \in \Lambda} (\alpha \cdot \beta_\lambda).$$

Lemma *comp_cupL_distr_l*

{ $A B C L : eqType$ } { $\alpha : Rel A B$ } { $\beta_{-L} : L \rightarrow Rel B C$ }:
 $\alpha \cdot (_ \beta_{-L}) = _ (\text{fun } l : L \Rightarrow (\alpha \cdot \beta_{-L} l)).$

Proof.

apply *inc_upper*.

move \Rightarrow *gamma*.

split; move \Rightarrow *H*.

rewrite $-(\text{@inv_invol } _ _ \alpha)$ in *H*.

apply *inc_residual* in *H*.

apply *inc_cupL*.

assert $(\forall l : L, \beta_{-L} l \quad (\alpha \# _ \text{gamma}))$.

apply *inc_cupL*.

apply *H*.

move \Rightarrow *l*.

rewrite $-(\text{@inv_invol } _ _ \alpha)$.

apply *inc_residual*.

apply *H0*.

rewrite $-(\text{@inv_invol } _ _ \alpha)$.

apply *inc_residual*.

apply *inc_cupL*.

assert $(\forall l : L, (\alpha \cdot \beta_{-L} l) \quad \text{gamma})$.

apply *inc_cupL*.

apply *H*.

move \Rightarrow *l*.

apply *inc_residual*.
 rewrite *inv_invol*.
 apply *H0*.
 Qed.

Lemma *comp_cup_distr_l*

$\{A\ B\ C : eqType\} \{alpha : Rel\ A\ B\} \{beta\ gamma : Rel\ B\ C\}:$
 $alpha \cdot (beta\ gamma) = (alpha \cdot beta) \quad (alpha \cdot gamma).$

Proof.

rewrite *cup_to_cupL cup_to_cupL*.
 rewrite *comp_cupL_distr_l*.
 apply *f_equal*.
 apply *functional_extensionality*.
 induction *x*.
 by [].
 by [].
 Qed.

Lemma 126 (*comp_cupL_distr_r, comp_cup_distr_r*) *Let $\alpha_\lambda : A \rightarrow B$ and $\beta : B \rightarrow C$. Then,*

$$(\sqcup_{\lambda \in \Lambda} \alpha_\lambda) \cdot \beta = \sqcup_{\lambda \in \Lambda} (\alpha_\lambda \cdot \beta).$$

Lemma *comp_cupL_distr_r*

$\{A\ B\ C\ L : eqType\} \{alpha_L : L \rightarrow Rel\ A\ B\} \{beta : Rel\ B\ C\}:$
 $(_ \ alpha_L) \cdot beta = _ \ (fun\ l : L \Rightarrow (alpha_L\ l \cdot beta)).$

Proof.

replace $(fun\ l : L \Rightarrow alpha_L\ l \cdot beta)$ with $(fun\ l : L \Rightarrow (beta \# \cdot alpha_L\ l \#) \#)$.
 rewrite *inv_cupL_distr*.
 rewrite *comp_cupL_distr_l*.
 rewrite *inv_cupL_distr*.
 rewrite *comp_inv*.
 by [rewrite *inv_invol inv_invol*].
 apply *functional_extensionality*.
 move $\Rightarrow l$.
 rewrite *comp_inv*.
 by [rewrite *inv_invol inv_invol*].
 Qed.

Lemma *comp_cup_distr_r*

$\{A\ B\ C : eqType\} \{alpha\ beta : Rel\ A\ B\} \{gamma : Rel\ B\ C\}:$
 $(alpha\ beta) \cdot gamma = (alpha \cdot gamma) \quad (beta \cdot gamma).$

Proof.

rewrite *cup_to_cupL cup_to_cupL*.
 rewrite *comp_cupL_distr_r*.

apply f_equal.
 apply *functional_extensionality*.
 induction *x*.
 by [].
 by [].
 Qed.

Lemma 127 (comp_capL_distr) *Let $\alpha : A \rightarrow B$, $\beta_\lambda : B \rightarrow C$ and $\gamma : C \rightarrow D$. Then,*

$$\alpha \cdot (\prod_{\lambda \in \Lambda} \beta_\lambda) \cdot \gamma \sqsubseteq \prod_{\lambda \in \Lambda} (\alpha \cdot \beta_\lambda \cdot \gamma).$$

Lemma *comp_capL_distr* {*A B C D L : eqType*}
 {*alpha : Rel A B*} {*beta_L : L → Rel B C*} {*gamma : Rel C D*}:
 (*alpha* · (_ *beta_L*)) · *gamma*
 _ (fun *l* : *L* ⇒ ((*alpha* · *beta_L l*) · *gamma*)).

Proof.

apply *inc_capL*.
 move ⇒ *l*.
 apply *comp_inc_compat_ab_a'b*.
 apply *comp_inc_compat_ab_ab'*.
 apply *inc_capL*.
 apply *inc_refl*.
 Qed.

Lemma 128 (comp_capL_distr_l, comp_cap_distr_l) *Let $\alpha : A \rightarrow B$, $\beta_\lambda : B \rightarrow C$. Then,*

$$\alpha \cdot (\prod_{\lambda \in \Lambda} \beta_\lambda) \sqsubseteq \prod_{\lambda \in \Lambda} (\alpha \cdot \beta_\lambda).$$

Lemma *comp_capL_distr_l*
 {*A B C L : eqType*} {*alpha : Rel A B*} {*beta_L : L → Rel B C*}:
 (*alpha* · (_ *beta_L*)) _ (fun *l* : *L* ⇒ (*alpha* · *beta_L l*)).

Proof.

move : (@*comp_capL_distr* _ _ _ _ *alpha beta_L (Id C)*) ⇒ *H*.
 rewrite *comp_id_r* in *H*.
 replace (fun *l* : *L* ⇒ (*alpha* · *beta_L l*) · *Id C*) with (fun *l* : *L* ⇒ (*alpha* · *beta_L l*))
 in *H*.
 apply *H*.
 apply *functional_extensionality*.
 move ⇒ *l*.
 by [rewrite *comp_id_r*].
 Qed.

Lemma *comp_cap_distr_l*

$\{A\ B\ C : eqType\} \{alpha : Rel\ A\ B\} \{beta\ gamma : Rel\ B\ C\}:$
 $(alpha \cdot (beta\ gamma)) \quad ((alpha \cdot beta) \quad (alpha \cdot gamma)).$

Proof.

rewrite *cap_to_capL cap_to_capL*.

apply (@*inc_trans* _ _ _ _ *comp_capL_distr_l*).

replace (_ (*fun* *l* : *bool_eqType* \Rightarrow *alpha* \cdot (*if* *l* then *beta* else *gamma*))) with (_ (*fun* *b* : *bool_eqType* \Rightarrow *if* *b* then *alpha* \cdot *beta* else *alpha* \cdot *gamma*)).

apply *inc_refl*.

apply *f_equal*.

apply *functional_extensionality*.

induction *x*.

by [].

by [].

Qed.

Lemma 129 (*comp_capL_distr_r, comp_cap_distr_r*) *Let $\alpha_\lambda : A \rightarrow B$, $\beta : B \rightarrow C$. Then,*

$$(\prod_{\lambda \in \Lambda} \alpha_\lambda) \cdot \beta \sqsubseteq \prod_{\lambda \in \Lambda} (\alpha_\lambda \cdot \beta).$$

Lemma *comp_capL_distr_r*

$\{A\ B\ C\ L : eqType\} \{beta : Rel\ B\ C\} \{alpha_L : L \rightarrow Rel\ A\ B\}:$
 $((_ \ alpha_L) \cdot beta) \quad _ \ (fun\ l : L \Rightarrow (alpha_L\ l \cdot beta)).$

Proof.

move : (@*comp_capL_distr* _ _ _ _ (*Id* *A*) *alpha_L* *beta*) \Rightarrow *H*.

rewrite *comp_id_l* in *H*.

replace (*fun* *l* : *L* \Rightarrow (*Id* *A* \cdot *alpha_L* *l*) \cdot *beta*) with (*fun* *l* : *L* \Rightarrow *alpha_L* *l* \cdot *beta*) in *H*.

apply *H*.

apply *functional_extensionality*.

move \Rightarrow *l*.

by [rewrite *comp_id_l*].

Qed.

Lemma *comp_cap_distr_r*

$\{A\ B\ C : eqType\} \{alpha\ beta : Rel\ A\ B\} \{gamma : Rel\ B\ C\}:$
 $((alpha\ beta) \cdot gamma) \quad ((alpha \cdot gamma) \quad (beta \cdot gamma)).$

Proof.

rewrite *cap_to_capL cap_to_capL*.

apply (@*inc_trans* _ _ _ _ *comp_capL_distr_r*).

replace (_ (*fun* *l* : *bool_eqType* \Rightarrow (*if* *l* then *alpha* else *beta*) \cdot *gamma*)) with (_ (*fun* *b* : *bool_eqType* \Rightarrow *if* *b* then *alpha* \cdot *gamma* else *beta* \cdot *gamma*)).

apply *inc_refl*.

apply *f_equal*.

CHAPTER 5. LIBRARY RELATION_PROPERTIES

apply *functional_extensionality*.
 induction *x*.
 by [].
 by [].
 Qed.

Lemma 130 (comp_empty_l, comp_empty_r) *Let $\alpha : A \rightarrow B$, $\beta : B \rightarrow C$. Then,*

$$\alpha \cdot \phi_{BC} = \phi_{AB} \cdot \beta = \phi_{AC}.$$

Lemma comp_empty_r $\{A\ B\ C : eqType\} \{alpha : Rel\ A\ B\}$: $alpha \cdot \quad B\ C = \quad A\ C$.

Proof.

apply *inc_antisym*.
 rewrite -(@inv_invol _ _ alpha).
 apply *inc_residual*.
 apply *inc_empty_alpha*.
 apply *inc_empty_alpha*.
 Qed.

Lemma comp_empty_l $\{A\ B\ C : eqType\} \{beta : Rel\ B\ C\}$: $A\ B \cdot beta = \quad A\ C$.

Proof.

rewrite -(@inv_invol _ _ (A B · beta)).
 rewrite -inv_move comp_inv inv_empty inv_empty.
 apply *comp_empty_r*.
 Qed.

Lemma 131 (comp_either_empty) *Let $\alpha : A \rightarrow B$, $\beta : B \rightarrow C$. Then,*

$$\alpha = \phi_{AB} \vee \beta = \phi_{BC} \Rightarrow \alpha \cdot \beta = \phi_{AC}.$$

Lemma comp_either_empty $\{A\ B\ C : eqType\} \{alpha : Rel\ A\ B\} \{beta : Rel\ B\ C\}$:
 $alpha = \quad A\ B \vee beta = \quad B\ C \rightarrow alpha \cdot beta = \quad A\ C$.

Proof.

case; move $\Rightarrow H$.
 rewrite *H*.
 apply *comp_empty_l*.
 rewrite *H*.
 apply *comp_empty_r*.
 Qed.

Lemma 132 (comp_neither_empty) *Let $\alpha : A \rightarrow B$, $\beta : B \rightarrow C$. Then,*

$$\alpha \cdot \beta \neq \phi_{AC} \Rightarrow \alpha \neq \phi_{AB} \wedge \beta \neq \phi_{BC}.$$

Lemma *comp_neither_empty* $\{A\ B\ C : eqType\} \{alpha : Rel\ A\ B\} \{beta : Rel\ B\ C\}$:
 $alpha \cdot beta \neq A\ C \rightarrow alpha \neq A\ B \wedge beta \neq B\ C$.

Proof.

move $\Rightarrow H$.
 split; move $\Rightarrow H0$.
 apply H .
 rewrite $H0$.
 apply *comp_empty_l*.
 apply H .
 rewrite $H0$.
 apply *comp_empty_r*.
Qed.

5.5 単域と Tarski の定理

Lemma 133 (*lemma_for_tarski1*) *Let $\alpha : A \rightarrow B$ and $\alpha \neq \phi_{AB}$. Then,*

$$\nabla_{IA} \cdot \alpha \cdot \nabla_{BI} = id_I.$$

Lemma *lemma_for_tarski1* $\{A\ B : eqType\} \{alpha : Rel\ A\ B\}$:
 $alpha \neq A\ B \rightarrow ((i\ A \cdot alpha) \cdot B\ i) = Id\ i$.

Proof.

move $\Rightarrow H$.
 assert $((i\ A \cdot alpha) \cdot B\ i) \neq i\ i$.
 move $\Rightarrow H0$.
 apply H .
 apply *inc_antisym*.
 apply $(@inc_trans\ _ _ _ ((A\ i \cdot ((i\ A \cdot alpha) \cdot B\ i)) \cdot i\ B))$.
 rewrite *comp_assoc comp_assoc unit_universal*.
 rewrite *-comp_assoc -comp_assoc unit_universal*.
 apply $(@inc_trans\ _ _ _ (Id\ A \cdot alpha) \cdot Id\ B)$.
 rewrite *comp_id_l comp_id_r*.
 apply *inc_refl*.
 apply *comp_inc_compat*.
 apply *comp_inc_compat_ab_a'b*.
 apply *inc_alpha_universal*.
 apply *inc_alpha_universal*.
 rewrite $H0\ comp_empty_r\ comp_empty_l$.
 apply *inc_refl*.
 apply *inc_empty_alpha*.
 case $(@unit_empty_or_universal ((i\ A \cdot alpha) \cdot B\ i))$; move $\Rightarrow H1$.

apply *False_ind*.
 apply (*H0 H1*).
 rewrite *unit_identity_is_universal*.
 apply *H1*.
 Qed.

Lemma 134 (lemma_for_tarski2)

$$\nabla_{AI} \cdot \nabla_{IB} = \nabla_{AB}.$$

Lemma *lemma_for_tarski2* {*A B : eqType*}: $\nabla_{AI} \cdot \nabla_{IB} = \nabla_{AB}$.

Proof.

apply *inc_antisym*.
 apply *inc_alpha_universal*.
 apply (@*inc_trans* _ _ _ ($\nabla_{AI} \cdot \nabla_{IB}$) ∇_{AB})).
 apply (@*inc_trans* _ _ _ (*Id A* · ∇_{AB})).
 rewrite *comp_id_l*.
 apply *inc_refl*.
 apply *comp_inc_compat_ab_a'b*.
 apply *inc_alpha_universal*.
 rewrite -(@*unit_universal A*) *comp_assoc*.
 apply *comp_inc_compat_ab_ab'*.
 apply *inc_alpha_universal*.
 Qed.

Lemma 135 (tarski) *Let $\alpha : A \rightarrow B$ and $\alpha \neq \phi_{AB}$. Then,*

$$\nabla_{AA} \cdot \alpha \cdot \nabla_{BB} = \nabla_{AB}.$$

Lemma *tarski* {*A B : eqType*} {*alpha : Rel A B*}:
 $\alpha \neq \phi_{AB} \rightarrow ((\nabla_{AA} \cdot \alpha) \cdot \nabla_{BB}) = \nabla_{AB}$.

Proof.

move \Rightarrow *H*.
 rewrite -(@*unit_universal A*) -(@*unit_universal B*).
 move : (@*lemma_for_tarski1* _ _ *alpha H*) \Rightarrow *H0*.
 rewrite -*comp_assoc* (@*comp_assoc* _ _ _ _ (∇_{AA}) *i*) (@*comp_assoc* _ _ _ _ (∇_{BB}) *i*).
 rewrite *H0 comp_id_r*.
 apply *lemma_for_tarski2*.
 Qed.

CHAPTER 5. LIBRARY RELATION_PROPERTIES

Lemma 136 (comp_universal1) *Let $B \neq \emptyset$. Then,*

$$\nabla_{AB} \cdot \nabla_{BC} = \nabla_{AC}.$$

Lemma *comp_universal* { $A B C : eqType$ } : $B \rightarrow A B \cdot B C = A C$.

Proof.

move $\Rightarrow b$.

replace ($A B$) with ($A B \cdot B B$).

rewrite -(@lemma_for_tarski2 $A B$) -(@lemma_for_tarski2 $B C$).

rewrite (@comp_assoc _ _ _ ($A i$)) (@comp_assoc _ _ _ ($A i$)) -(@comp_assoc _ _ _ ($B i$)).

rewrite lemma_for_tarski1.

rewrite comp_id_l.

apply lemma_for_tarski2.

apply not_eq_sym.

move $\Rightarrow H$.

apply either_empty in H .

case H ; move $\Rightarrow H0$.

apply ($H0 b$).

apply ($H0 b$).

apply inc_antisym.

apply inc_alpha_universal.

apply (@inc_trans _ _ _ ($A B \cdot Id B$)).

rewrite comp_id_r.

apply inc_refl.

apply comp_inc_compat_ab_ab'.

apply inc_alpha_universal.

Qed.

Lemma 137 (comp_universal2)

$$\nabla_{IA}^\# \cdot \nabla_{IB} = \nabla_{AB}.$$

Lemma *comp_universal2* { $A B : eqType$ } : $i A \# \cdot i B = A B$.

Proof.

rewrite inv_universal.

apply lemma_for_tarski2.

Qed.

Lemma 138 (empty_equivalence1, empty_equivalence2, empty_equivalence3)

$$A = \emptyset \Leftrightarrow \nabla_{IA} = \phi_{IA} \Leftrightarrow \nabla_{AA} = \phi_{AA} \Leftrightarrow id_A = \phi_{AA}.$$

Lemma *empty_equivalence1* $\{A : eqType\} : (A \rightarrow False) \leftrightarrow \quad i \ A = \quad i \ A$.

Proof.

move : ($@either_empty \ i \ A$) $\Rightarrow H$.

split; move $\Rightarrow H0$.

apply *Logic.eq-sym*.

apply *H*.

right.

apply *H0*.

apply *Logic.eq-sym* in *H0*.

apply *H* in *H0*.

case *H0*.

move $\Rightarrow H1 \ H2$.

apply *H1*.

apply *tt*.

by [].

Qed.

Lemma *empty_equivalence2* $\{A : eqType\} : (A \rightarrow False) \leftrightarrow \quad A \ A = \quad A \ A$.

Proof.

move : ($@either_empty \ A \ A$) $\Rightarrow H$.

split; move $\Rightarrow H0$.

apply *Logic.eq-sym*.

apply *H*.

left.

apply *H0*.

apply *Logic.eq-sym* in *H0*.

apply *H* in *H0*.

case *H0*.

by [].

by [].

Qed.

Lemma *empty_equivalence3* $\{A : eqType\} : (A \rightarrow False) \leftrightarrow Id \ A = \quad A \ A$.

Proof.

split; move $\Rightarrow H$.

assert ($\quad A \ A = \quad A \ A$).

apply *empty_equivalence2*.

apply *H*.

apply *RelAB-unique*.

apply *Logic.eq-sym*.

apply *H0*.

assert ($\quad A \ A = \quad A \ A$).

by [rewrite $-(@comp_id_r _ _ (\quad A \ A)) \ H \ comp_empty_r$].

apply *either_empty* in *H0*.

case *H0*.

by [].

by [].

Qed.

Chapter 6

Library **Functions_Mappings**

```
Require Import Basic_Notations.  
Require Import Basic_Lemmas.  
Require Import Relation_Properties.  
Require Import Logic.FunctionalExtensionality.
```

6.1 全域性, 一価性, 写像に関する補題

Lemma 139 (id_function) $id_A : A \rightarrow A$ is a function.

```
Lemma id_function {A : eqType}: function_r (Id A).  
Proof.  
rewrite /function_r/total_r/univalent_r.  
rewrite inv_id comp_id_l.  
split.  
apply inc_refl.  
apply inc_refl.  
Qed.
```

Lemma 140 (unit_function) $\nabla_{AI} : A \rightarrow I$ is a function.

```
Lemma unit_function {A : eqType}: function_r ( A i).  
Proof.  
rewrite /function_r/total_r/univalent_r.  
rewrite inv_universal lemma_for_tarski2 unit_identity_is_universal.  
split.  
apply inc_alpha_universal.  
apply inc_alpha_universal.  
Qed.
```

CHAPTER 6. LIBRARY FUNCTIONS_MAPPINGS

Lemma 141 (total_comp) *Let $\alpha : A \rightarrow B$ and $\beta : B \rightarrow C$ be total relations, then $\alpha \cdot \beta$ is also a total relation.*

Lemma *total_comp* {A B C : eqType} {alpha : Rel A B} {beta : Rel B C}:

total_r alpha \rightarrow *total_r beta* \rightarrow *total_r (alpha • beta)*.

Proof.

rewrite /total_r.

move \Rightarrow *H H0*.

rewrite comp_inv comp_assoc -(@comp_assoc - - - - beta).

apply (@inc_trans - - - - H).

apply comp_inc_compat_ab_ab'.

apply comp_inc_compat_b_ab.

apply H0.

Qed.

Lemma 142 (univalent_comp) *Let $\alpha : A \rightarrow B$ and $\beta : B \rightarrow C$ be univalent relations, then $\alpha \cdot \beta$ is also a univalent relation.*

Lemma *univalent_comp* {A B C : eqType} {alpha : Rel A B} {beta : Rel B C}:

univalent_r alpha \rightarrow *univalent_r beta* \rightarrow *univalent_r (alpha • beta)*.

Proof.

rewrite /univalent_r.

move \Rightarrow *H H0*.

rewrite comp_inv comp_assoc -(@comp_assoc - - - - (alpha #)).

apply (fun H' \Rightarrow @inc_trans - - - - H' H0).

apply comp_inc_compat_ab_ab'.

apply comp_inc_compat_ab_b.

apply H.

Qed.

Lemma 143 (function_comp) *Let $\alpha : A \rightarrow B$ and $\beta : B \rightarrow C$ be functions, then $\alpha \cdot \beta$ is also a function.*

Lemma *function_comp* {A B C : eqType} {alpha : Rel A B} {beta : Rel B C}:

function_r alpha \rightarrow *function_r beta* \rightarrow *function_r (alpha • beta)*.

Proof.

elim \Rightarrow *H H0*.

elim \Rightarrow *H1 H2*.

split.

apply (total_comp H H1).

apply (univalent_comp H0 H2).

Qed.

CHAPTER 6. LIBRARY FUNCTIONS_MAPPINGS

Lemma 144 (total_comp2) *Let $\alpha : A \rightarrow B$, $\beta : B \rightarrow C$ and $\alpha \cdot \beta$ be a total relation, then α is also a total relation.*

Lemma `total_comp2` $\{A\ B\ C : \text{eqType}\} \{ \text{alpha} : \text{Rel}\ A\ B\} \{ \text{beta} : \text{Rel}\ B\ C\}$:
`total_r (alpha · beta) → total_r alpha.`

Proof.

`move ⇒ H.`

`apply inc_def1 in H.`

`rewrite comp_inv cap_comm comp_assoc in H.`

`rewrite /total_r.`

`rewrite H.`

`apply (@inc_trans _ _ _ _ (@dedekind _ _ _ _ _)).`

`apply comp_inc_compat.`

`apply cap_l.`

`rewrite comp_id_r.`

`apply cap_r.`

Qed.

Lemma 145 (univalent_comp2) *Let $\alpha : A \rightarrow B$, $\beta : B \rightarrow C$, $\alpha \cdot \beta$ be a univalent relation and $\alpha^\#$ be a total relation, then β is a univalent relation.*

Lemma `univalent_comp2` $\{A\ B\ C : \text{eqType}\} \{ \text{alpha} : \text{Rel}\ A\ B\} \{ \text{beta} : \text{Rel}\ B\ C\}$:
`univalent_r (alpha · beta) → total_r (alpha #) → univalent_r beta.`

Proof.

`move ⇒ H H0.`

`apply (fun H' ⇒ @inc_trans _ _ _ _ H' H).`

`rewrite comp_inv comp_assoc -(@comp_assoc _ _ _ _ alpha).`

`apply comp_inc_compat_ab_ab'.`

`rewrite /total_r in H0.`

`rewrite inv_invol in H0.`

`apply (comp_inc_compat_b_ab H0).`

Qed.

Lemma 146 (total_inc) *Let $\alpha : A \rightarrow B$ be a total relation and $\alpha \sqsubseteq \beta$, then β is also a total relation.*

Lemma `total_inc` $\{A\ B : \text{eqType}\} \{ \text{alpha}\ \text{beta} : \text{Rel}\ A\ B\}$:
`total_r alpha → alpha beta → total_r beta.`

Proof.

`move ⇒ H H0.`

`apply (@inc_trans _ _ _ _ H).`

`apply comp_inc_compat.`

`apply H0.`

CHAPTER 6. LIBRARY FUNCTIONS_MAPPINGS

apply (@inc_inv _ _ _ _ H0).

Qed.

Lemma 147 (univalent_inc) *Let $\alpha : A \rightarrow B$ be a univalent relation and $\beta \sqsubseteq \alpha$, then β is also a univalent relation.*

Lemma univalent_inc {A B : eqType} {alpha beta : Rel A B}:
 univalent_r alpha \rightarrow beta alpha \rightarrow univalent_r beta.

Proof.

move \Rightarrow H H0.

apply (fun H' \Rightarrow @inc_trans _ _ _ _ H' H).

apply comp_inc_compat.

apply (@inc_inv _ _ _ _ H0).

apply H0.

Qed.

Lemma 148 (function_inc) *Let $\alpha, \beta : A \rightarrow B$ be functions and $\alpha \sqsubseteq \beta$. Then,*

$$\alpha = \beta.$$

Lemma function_inc {A B : eqType} {alpha beta : Rel A B}:
 function_r alpha \rightarrow function_r beta \rightarrow alpha beta \rightarrow alpha = beta.

Proof.

move \Rightarrow H H0 H1.

apply inc_antisym.

apply H1.

apply (@inc_trans _ _ _ ((alpha \cdot alpha #) \cdot beta)).

apply comp_inc_compat_b_ab.

apply H.

move : (@inc_inv _ _ _ _ H1) \Rightarrow H2.

apply (@inc_trans _ _ _ ((alpha \cdot beta #) \cdot beta)).

apply comp_inc_compat_ab_a'b.

apply comp_inc_compat_ab_ab'.

apply H2.

rewrite comp_assoc.

apply comp_inc_compat_ab_a.

apply H0.

Qed.

Lemma 149 (total_universal) *If ∇_{IB} be a total relation, then*

$$\nabla_{AB} \cdot \nabla_{BC} = \nabla_{AC}.$$

CHAPTER 6. LIBRARY FUNCTIONS_MAPPINGS

Lemma *total_universal* $\{A\ B\ C : eqType\}$:
 $total_r\ (_ \ i\ B) \rightarrow _ \ A\ B \cdot _ \ B\ C = _ \ A\ C.$

Proof.

move $\Rightarrow H$.

rewrite $-(@lemma_for_tarski2\ A\ B)\ -(@lemma_for_tarski2\ B\ C).$

rewrite *comp_assoc* $-(@comp_assoc\ _ \ _ \ _ \ (_ \ i\ B)).$

replace $(_ \ i\ B \cdot _ \ B\ i)$ with $(Id\ i).$

rewrite *comp_id_l*.

apply *lemma_for_tarski2*.

apply *inc_antisym*.

rewrite */total_r* in H .

rewrite *inv_universal* in H .

apply H .

rewrite *unit_identity_is_universal*.

apply *inc_alpha_universal*.

Qed.

Lemma 150 (function_rel_inv_rel) *Let $\alpha : A \rightarrow B$ be function. Then,*

$$\alpha \cdot \alpha^\# \cdot \alpha = \alpha.$$

Lemma *function_rel_inv_rel* $\{A\ B : eqType\}\ \{\alpha : Rel\ A\ B\}$:
 $function_r\ \alpha \rightarrow (\alpha \cdot \alpha^\#) \cdot \alpha = \alpha.$

Proof.

move $\Rightarrow H$.

apply *inc_antisym*.

rewrite *comp_assoc*.

apply *comp_inc_compat_ab_a*.

apply H .

apply *comp_inc_compat_b_ab*.

apply H .

Qed.

Lemma 151 (function_capL_distr) *Let $f : A \rightarrow B, g : D \rightarrow C$ be functions and $\alpha_\lambda : B \rightarrow C$. Then,*

$$f \cdot (\sqcap_{\lambda \in \Lambda} \alpha_\lambda) \cdot g^\# = \sqcap_{\lambda \in \Lambda} (f \cdot \alpha_\lambda \cdot g^\#).$$

Lemma *function_capL_distr*

$\{A\ B\ C\ D\ L : eqType\}\ \{f : Rel\ A\ B\}\ \{g : Rel\ D\ C\}\ \{\alpha_L : L \rightarrow Rel\ B\ C\}$:

$function_r\ f \rightarrow function_r\ g \rightarrow$

$(f \cdot (_ \ _ \ \alpha_L)) \cdot g^\# = _ \ (\text{fun } l : L \Rightarrow (f \cdot \alpha_L\ l) \cdot g^\#).$

Proof.

elim $\Rightarrow H\ H0$.

```

elim  $\Rightarrow$  H1 H2.
apply inc_antisym.
apply comp_capL_distr.
apply (@inc_trans _ _ _ (((f · f #) · ( _ (fun l : L  $\Rightarrow$  (f · alpha_L l) · g #)) · (g · g #)))).
apply (@inc_trans _ _ _ (((f · f #) · ( _ (fun l : L  $\Rightarrow$  (f · alpha_L l) · g #)))).
apply (comp_inc_compat_b_ab H).
apply (comp_inc_compat_a_ab H1).
rewrite (@comp_assoc _ _ _ _ (f #)) comp_assoc - (@comp_assoc _ _ _ _ g) - comp_assoc.
apply comp_inc_compat_ab_a'b.
apply comp_inc_compat_ab_ab'.
apply (@inc_trans _ _ _ ( _ (fun l : L  $\Rightarrow$  (f # · ((f · alpha_L l) · g #)) · g))).
apply comp_capL_distr.
replace (fun l : L  $\Rightarrow$  (f # · ((f · alpha_L l) · g #)) · g) with (fun l : L  $\Rightarrow$  ((f # · f) · alpha_L l) · (g # · g)).
apply inc_capL.
move  $\Rightarrow$  l.
apply (@inc_trans _ _ _ ((f # · f) · alpha_L l)).
apply (@inc_trans _ _ _ (((f # · f) · alpha_L l) · (g # · g))).
move : l.
apply inc_capL.
apply inc_refl.
apply (comp_inc_compat_ab_a H2).
apply (comp_inc_compat_ab_b H0).
apply functional_extensionality.
move  $\Rightarrow$  l.
by [rewrite comp_assoc comp_assoc comp_assoc comp_assoc comp_assoc].
Qed.

```

Lemma 152 (function_cap_distr, function_cap_distr_l, function_cap_distr_r)

Let $f : A \rightarrow B, g : D \rightarrow C$ be functions and $\alpha, \beta : B \rightarrow C$. Then,

$$f \cdot (\alpha \sqcap \beta) \cdot g^\# = (f \cdot \alpha \cdot g^\#) \sqcap (f \cdot \beta \cdot g^\#).$$

Lemma function_cap_distr

$\{A\ B\ C\ D : \text{eqType}\} \{f : \text{Rel } A\ B\} \{\alpha\ \beta : \text{Rel } B\ C\} \{g : \text{Rel } D\ C\} :$
 $\text{function_r } f \rightarrow \text{function_r } g \rightarrow$
 $(f \cdot (\alpha \sqcap \beta)) \cdot g^\# = ((f \cdot \alpha) \cdot g^\#) \sqcap ((f \cdot \beta) \cdot g^\#).$

Proof.

```

rewrite cap_to_capL cap_to_capL.
move  $\Rightarrow$  H H0.
rewrite (function_capL_distr H H0).
apply f_equal.

```

apply *functional_extensionality*.

induction *x*.

by [].

by [].

Qed.

Lemma *function_cap_distr_l*

$\{A\ B\ C : eqType\} \{f : Rel\ A\ B\} \{alpha\ beta : Rel\ B\ C\} :$
function_r *f* \rightarrow
 $f \cdot (alpha \quad beta) = (f \cdot alpha) \quad (f \cdot beta).$

Proof.

move : (@*id_function* *C*) \Rightarrow *H*.

move \Rightarrow *H0*.

apply (@*function_cap_distr* _ _ _ *f* *alpha* *beta*) in *H*.

rewrite *inv_id comp_id_r comp_id_r comp_id_r* in *H*.

apply *H*.

apply *H0*.

Qed.

Lemma *function_cap_distr_r*

$\{B\ C\ D : eqType\} \{alpha\ beta : Rel\ B\ C\} \{g : Rel\ D\ C\} :$
function_r *g* \rightarrow
 $(alpha \quad beta) \cdot g \# = (alpha \cdot g \#) \quad (beta \cdot g \#).$

Proof.

move : (@*id_function* *B*) \Rightarrow *H*.

move \Rightarrow *H0*.

apply (@*function_cap_distr* _ _ _ _ *alpha* *beta* *g*) in *H*.

rewrite *comp_id_l comp_id_l comp_id_l* in *H*.

apply *H*.

apply *H0*.

Qed.

Lemma 153 (function_move1) *Let $\alpha : A \rightarrow B$ be a function, $\beta : B \rightarrow C$ and $\gamma : A \rightarrow C$. Then,*

$$\gamma \sqsubseteq \alpha \cdot \beta \Leftrightarrow \alpha^\# \cdot \gamma \sqsubseteq \beta.$$

Lemma *function_move1* $\{A\ B\ C : eqType\} \{alpha : Rel\ A\ B\} \{beta : Rel\ B\ C\} \{gamma : Rel\ A\ C\} :$

function_r *alpha* $\rightarrow (gamma \quad (alpha \cdot beta) \leftrightarrow (alpha \# \cdot gamma) \quad beta).$

Proof.

move \Rightarrow *H*.

split; move \Rightarrow *H0*.

apply (@*inc_trans* _ _ _ ((*alpha* # \cdot *alpha*) \cdot *beta*)).

rewrite *comp_assoc*.


```

apply (comp_inc_compat_ab_ab' H0).
apply comp_inc_compat_ab_b.
apply H.
apply (@inc_trans _ _ _ ((alpha · alpha #) · gamma)).
apply comp_inc_compat_b_ab.
apply H.
rewrite comp_assoc.
apply (comp_inc_compat_ab_ab' H0).
Qed.

```

Lemma 154 (function_move2) *Let $\beta : B \rightarrow C$ be a function, $\alpha : A \rightarrow B$ and $\gamma : A \rightarrow C$. Then,*

$$\alpha \cdot \beta \sqsubseteq \gamma \Leftrightarrow \alpha \sqsubseteq \gamma \cdot \beta^\#.$$

Lemma function_move2 $\{A\ B\ C : \text{eqType}\} \{alpha : \text{Rel } A\ B\} \{beta : \text{Rel } B\ C\} \{gamma : \text{Rel } A\ C\}$:

$\text{function_r } beta \rightarrow ((alpha \cdot beta) \quad gamma \leftrightarrow alpha \quad (gamma \cdot beta \#)).$

Proof.

```

move  $\Rightarrow$  H.
split; move  $\Rightarrow$  H0.
apply (@inc_trans _ _ _ ((alpha · beta) · beta #)).
rewrite comp_assoc.
apply comp_inc_compat_a_ab.
apply H.
apply (comp_inc_compat_ab_a'b H0).
apply (@inc_trans _ _ _ ((gamma · beta #) · beta)).
apply (comp_inc_compat_ab_a'b H0).
rewrite comp_assoc.
apply comp_inc_compat_ab_a.
apply H.
Qed.

```

Lemma 155 (function_rpc_distr) *Let $f : A \rightarrow B, g : D \rightarrow C$ be functions and $\alpha, \beta : B \rightarrow C$. Then,*

$$f \cdot (\alpha \Rightarrow \beta) \cdot g^\# = (f \cdot \alpha \cdot g^\#) \Rightarrow (f \cdot \beta \cdot g^\#).$$

Lemma function_rpc_distr

$\{A\ B\ C\ D : \text{eqType}\} \{f : \text{Rel } A\ B\} \{alpha\ beta : \text{Rel } B\ C\} \{g : \text{Rel } D\ C\}$:

$\text{function_r } f \rightarrow \text{function_r } g \rightarrow$

$(f \cdot (alpha \gg beta)) \cdot g \# = ((f \cdot alpha) \cdot g \#) \gg ((f \cdot beta) \cdot g \#).$

Proof.

$\text{move} \Rightarrow H\ H0.$

```
apply inc_lower.
move  $\Rightarrow$  gamma.
split; move  $\Rightarrow$  H1.
apply inc_rpc.
apply (function_move2 H0).
apply (function_move1 H).
apply (@inc_trans _ _ _ (((f #  $\cdot$  gamma)  $\cdot$  g) ((f #  $\cdot$  ((f  $\cdot$  alpha)  $\cdot$  g #))  $\cdot$  g))).
rewrite -comp_assoc.
apply (fun H'  $\Rightarrow$  @inc_trans _ _ _ _ H' (@comp_cap_distr_r _ _ _ _ _)).
apply comp_inc_compat_ab_a'b.
apply comp_cap_distr_l.
apply (function_move2 H0) in H1.
apply (function_move1 H) in H1.
rewrite -inc_rpc comp_assoc.
apply (@inc_trans _ _ _ _ H1).
apply rpc_inc_compat_r.
rewrite comp_assoc comp_assoc comp_assoc -comp_assoc.
apply (@inc_trans _ _ _ (alpha  $\cdot$  (g #  $\cdot$  g))).
apply comp_inc_compat_ab_b.
apply H.
apply comp_inc_compat_ab_a.
apply H0.
apply (function_move2 H0).
apply (function_move1 H).
apply inc_rpc.
apply (@inc_trans _ _ _ _ (@dedekind _ _ _ _ _)).
apply (@inc_trans _ _ _ (f #  $\cdot$  ((gamma  $\cdot$  g) ((f #) #  $\cdot$  alpha)))).
apply comp_inc_compat_ab_a'b.
apply cap_l.
rewrite inv_invol.
apply (@inc_trans _ _ _ ((f #  $\cdot$  (gamma ((f  $\cdot$  alpha)  $\cdot$  g #)))  $\cdot$  g)).
rewrite comp_assoc.
apply comp_inc_compat_ab_ab'.
apply (@inc_trans _ _ _ _ (@dedekind _ _ _ _ _)).
apply comp_inc_compat_ab_ab'.
apply cap_l.
apply (function_move2 H0).
apply (function_move1 H).
rewrite -inc_rpc -comp_assoc.
apply H1.
Qed.
```

Lemma 156 (function_inv_rel1, function_inv_rel2) *Let $f : A \rightarrow B$ be a function. Then,*

$$f^\# \cdot f = id_B \sqcap f^\# \cdot \nabla_{AA} \cdot f = id_B \sqcap \nabla_{BA} \cdot f.$$

Lemma *function_inv_rel1* { $A\ B : eqType$ } { $f : Rel\ A\ B$ }:
function_r $f \rightarrow f \# \cdot f = Id\ B \quad ((f \# \cdot \quad A\ A) \cdot f).$

Proof.

move $\Rightarrow H$.
 apply *inc_antisym*.
 apply *inc_cap*.
 split.
 apply H .
 apply *comp_inc_compat_ab_a'b*.
 apply *comp_inc_compat_a_ab*.
 apply *inc_alpha_universal*.
 apply (*@inc_trans* _ _ _ (*Id* B ($B\ A \cdot f$))).
 apply *cap_inc_compat_l*.
 apply *comp_inc_compat_ab_a'b*.
 apply *inc_alpha_universal*.
 rewrite *cap_comm*.
 apply (*@inc_trans* _ _ _ _ (*@dedekind* _ _ _ _ _)).
 rewrite *comp_id_l comp_id_r cap_comm inv_universal*.
 rewrite *cap_universal cap_universal*.
 apply *inc_refl*.

Qed.

Lemma *function_inv_rel2* { $A\ B : eqType$ } { $f : Rel\ A\ B$ }:
function_r $f \rightarrow f \# \cdot f = Id\ B \quad (\quad B\ A \cdot f).$

Proof.

move $\Rightarrow H$.
 apply *inc_antisym*.
 rewrite (*@function_inv_rel1* _ _ _ H).
 apply *cap_inc_compat_l*.
 apply *comp_inc_compat_ab_a'b*.
 apply *inc_alpha_universal*.
 rewrite *cap_comm*.
 apply (*@inc_trans* _ _ _ _ (*@dedekind* _ _ _ _ _)).
 rewrite *comp_id_l comp_id_r cap_comm inv_universal*.
 rewrite *cap_universal cap_universal*.
 apply *inc_refl*.

Qed.

Lemma 157 (function_dedekind1, function_dedekind2) *Let $f : A \rightarrow B$ be a function, $\mu : C \rightarrow A$ and $\rho : C \rightarrow B$. Then,*

$$(\mu \sqcap \rho \cdot f^\#) \cdot f = \mu \cdot f \sqcap \rho \wedge \rho \cdot f^\# \cdot f = \nabla_{CA} \cdot f \sqcap \rho.$$

Lemma function_dedekind1

$\{A\ B\ C : \text{eqType}\} \{f : \text{Rel}\ A\ B\} \{\mu : \text{Rel}\ C\ A\} \{\rho : \text{Rel}\ C\ B\} :$
 $\text{function_r } f \rightarrow (\mu \quad (\rho \cdot f^\#)) \cdot f = (\mu \cdot f) \quad \rho.$

Proof.

move $\Rightarrow H$.

apply *inc_antisym*.

apply (*@inc_trans* _ _ _ _ (*@comp_cap_distr_r* _ _ _ _ _)).

apply *cap_inc_compat_l*.

rewrite *comp_assoc*.

apply *comp_inc_compat_ab_a*.

apply *H*.

apply (*@inc_trans* _ _ _ _ (*@dedekind* _ _ _ _ _)).

apply *comp_inc_compat_ab_ab'*.

apply *cap_l*.

Qed.

Lemma function_dedekind2 $\{A\ B\ C : \text{eqType}\} \{f : \text{Rel}\ A\ B\} \{\rho : \text{Rel}\ C\ B\} :$
 $\text{function_r } f \rightarrow (\rho \cdot f^\#) \cdot f = (\quad C\ A \cdot f) \quad \rho.$

Proof.

move $\Rightarrow H$.

move : (*@function_dedekind1* _ _ _ *f* (*\ C\ A*) *\ rho* *H*) $\Rightarrow H0$.

rewrite *cap_comm cap_universal* in *H0*.

apply *H0*.

Qed.

6.2 全射, 単射に関する補題

Lemma 158 (surjection_comp) *Let $\alpha : A \rightarrow B$ and $\beta : B \rightarrow C$ be surjections, then $\alpha \cdot \beta$ is also a surjection.*

Lemma surjection_comp $\{A\ B\ C : \text{eqType}\} \{\alpha : \text{Rel}\ A\ B\} \{\beta : \text{Rel}\ B\ C\} :$
 $\text{surjection_r } \alpha \rightarrow \text{surjection_r } \beta \rightarrow \text{surjection_r } (\alpha \cdot \beta).$

Proof.

rewrite */surjection_r*.

elim $\Rightarrow H\ H0$.

elim $\Rightarrow H1\ H2$.

split.

```

apply (function_comp H H1).
rewrite comp_inv.
apply (total_comp H2 H0).
Qed.

```

Lemma 159 (injection_comp) *Let $\alpha : A \rightarrow B$ and $\beta : B \rightarrow C$ be injections, then $\alpha \cdot \beta$ is also an injection.*

Lemma *injection_comp* {A B C : eqType} {alpha : Rel A B} {beta : Rel B C}:
injection_r alpha → injection_r beta → injection_r (alpha • beta).

Proof.
rewrite /injection_r.
elim ⇒ H H0.
elim ⇒ H1 H2.
split.
apply (function_comp H H1).
rewrite comp_inv.
apply (univalent_comp H2 H0).
Qed.

Lemma 160 (bijection_comp) *Let $\alpha : A \rightarrow B$ and $\beta : B \rightarrow C$ be bijections, then $\alpha \cdot \beta$ is also a bijection.*

Lemma *bijection_comp* {A B C : eqType} {alpha : Rel A B} {beta : Rel B C}:
bijection_r alpha → bijection_r beta → bijection_r (alpha • beta).

Proof.
rewrite /bijection_r.
elim ⇒ H.
elim ⇒ H0 H1.
elim ⇒ H2.
elim ⇒ H3 H4.
split.
apply (function_comp H H2).
rewrite comp_inv.
split.
apply (total_comp H3 H0).
apply (univalent_comp H4 H1).
Qed.

Lemma 161 (surjection_unique1) *Let $e : A \twoheadrightarrow B$ be a surjection, $f : A \rightarrow C$ be a function and $e \cdot e^\sharp \sqsubseteq f \cdot f^\sharp$, then there exists a unique function $g : B \rightarrow C$ s.t. $f = eg$.*

Lemma *surjection_unique1* {A B C : eqType} {e : Rel A B} {f : Rel A C}:

$surjection_r\ e \rightarrow function_r\ f \rightarrow (e \cdot e \#) \quad (f \cdot f \#) \rightarrow$
 $(\exists! g : Rel\ B\ C, function_r\ g \wedge f = e \cdot g).$

Proof.

```
rewrite /surjection_r/function_r/total_r/univalent_r.
elim.
elim  $\Rightarrow H\ H0\ H1$ .
elim  $\Rightarrow H2\ H3\ H4$ .
 $\exists (e \# \cdot f)$ .
repeat split.
rewrite comp_inv comp_assoc -(@comp_assoc _ _ _ f).
apply (@inc_trans _ _ _ _ H1).
apply comp_inc_compat_ab_ab'.
apply comp_inc_compat_b_ab.
apply H2.
rewrite comp_inv inv_invol comp_assoc -(@comp_assoc _ _ _ e).
apply (@inc_trans _ _ _ (f #  $\cdot ((f \cdot f \#) \cdot f)))$ .
apply comp_inc_compat_ab_ab'.
apply (comp_inc_compat_ab_a'b H4).
rewrite comp_assoc -comp_assoc.
apply (fun H'  $\Rightarrow$  @inc_trans _ _ _ _ H' H3).
apply (comp_inc_compat_ab_a H3).
apply function_inc.
split.
apply H2.
apply H3.
split.
rewrite /total_r.
rewrite comp_inv comp_inv inv_invol.
rewrite -(@comp_assoc _ _ _ e) (@comp_assoc _ _ _ e) (@comp_assoc _ _ _ f)
-(@comp_assoc _ _ _ f).
apply (@inc_trans _ _ _ _ H).
apply comp_inc_compat_a_ab.
apply (@inc_trans _ _ _ _ H2).
apply (comp_inc_compat_a_ab H).
rewrite /univalent_r.
rewrite comp_inv comp_inv inv_invol.
rewrite (@comp_assoc _ _ _ e) -(@comp_assoc _ _ _ e) comp_assoc -(@comp_assoc
_ _ _ _ f).
apply (@inc_trans _ _ _ (f #  $\cdot (((f \cdot f \#) \cdot (f \cdot f \#)) \cdot f)))$ .
apply comp_inc_compat_ab_ab'.
apply comp_inc_compat_ab_a'b.
apply comp_inc_compat.
```

```

apply H4.
apply H4.
rewrite comp_assoc (@comp_assoc _ _ _ _ f) - (@comp_assoc _ _ _ _ (f #)) - (@comp_assoc
_ _ _ _ (f #)) (@comp_assoc _ _ _ _ (f #)) - (@comp_assoc _ _ _ _ (f #)).
apply (fun H' => @inc_trans _ _ _ _ H' H3).
apply comp_inc_compat_ab_a.
apply (fun H' => @inc_trans _ _ _ _ H' H3).
apply (comp_inc_compat_ab_a H3).
rewrite -comp_assoc.
apply (comp_inc_compat_b_ab H).
move => g.
elim.
elim => H5 H6 H7.
replace g with (e # • (e • g)).
apply f_equal.
apply H7.
rewrite -comp_assoc.
apply inc_antisym.
apply (comp_inc_compat_ab_b H0).
rewrite inv_invol in H1.
apply (comp_inc_compat_b_ab H1).
Qed.

```

Lemma 162 (surjection_unique2) *Let $e : A \twoheadrightarrow B$ be a surjection, $f : A \rightarrow C$ be a function and $e \cdot e^\# = f \cdot f^\#$, then function $e^\# f$ is an injection.*

Lemma *surjection_unique2* {A B C : eqType} {e : Rel A B} {f : Rel A C}:
 surjection_r e → function_r f → (e • e #) = (f • f #) → injection_r (e # • f).

Proof.

```

rewrite /surjection_r/injection_r/function_r/total_r/univalent_r.
elim.
elim => H H0 H1.
elim => H2 H3 H4.
repeat split.
rewrite comp_inv comp_assoc - (@comp_assoc _ _ _ _ f).
apply (@inc_trans _ _ _ _ H1).
apply comp_inc_compat_ab_ab'.
apply comp_inc_compat_b_ab.
apply H2.
rewrite comp_inv inv_invol comp_assoc - (@comp_assoc _ _ _ _ e).
rewrite H4.
rewrite comp_assoc -comp_assoc.
apply (fun H' => @inc_trans _ _ _ _ H' H3).

```

```

apply (comp_inc_compat_ab_a H3).
rewrite inv_invol comp_inv inv_invol comp_assoc -(@comp_assoc _ _ _ f).
rewrite -H4.
rewrite comp_assoc -comp_assoc.
apply (fun H' => @inc_trans _ _ _ _ H' H0).
apply comp_inc_compat_ab_a.
apply H0.
Qed.

```

Lemma 163 (injection_unique1) *Let $m : B \rightarrowtail A$ be an injection, $f : C \rightarrow A$ be a function and $f^\# \cdot f \sqsubseteq m^\# \cdot m$, then there exists a unique function $g : C \rightarrow B$ s.t. $f = gm$.*

Lemma *injection_unique1* {A B C : eqType} {m : Rel B A} {f : Rel C A}:
 injection_r m → function_r f → (f # · f) (m # · m) →
 (∃! g : Rel C B, function_r g ∧ f = g · m).

Proof.

```

rewrite /injection_r/function_r/total_r/univalent_r.
elim.
elim => H H0 H1.
elim => H2 H3 H4.
∃ (f · m #).
repeat split.
rewrite comp_inv inv_invol comp_assoc -(@comp_assoc _ _ _ _ m).
apply (@inc_trans _ _ _ (f · ((f # · f) · f #))).
rewrite comp_assoc -comp_assoc.
apply (@inc_trans _ _ _ _ H2).
apply (comp_inc_compat_a_ab H2).
apply comp_inc_compat_ab_ab'.
apply (comp_inc_compat_ab_a'b H4).
rewrite comp_inv comp_assoc -(@comp_assoc _ _ _ _ f).
apply (fun H' => @inc_trans _ _ _ _ H' H1).
apply comp_inc_compat_ab_ab'.
apply (comp_inc_compat_ab_b H3).
rewrite comp_assoc.
apply Logic.eq-sym.
apply function_inc.
split.
rewrite /total_r.
rewrite comp_inv comp_inv inv_invol.
apply (@inc_trans _ _ _ _ H2).
apply comp_inc_compat.
apply (@inc_trans _ _ _ (f · (f # · f))).
rewrite -comp_assoc.

```



```

apply (comp_inc_compat_b_ab H2).
apply (comp_inc_compat_ab_ab' H4).
apply (@inc_trans _ _ _ ((f # · f) · f #)).
rewrite comp_assoc.
apply (comp_inc_compat_a_ab H2).
apply (comp_inc_compat_ab_a'b H4).
rewrite /univalent_r.
rewrite comp_inv comp_inv inv_invol comp_assoc -(@comp_assoc _ _ _ _ f).
apply (fun H' ⇒ @inc_trans _ _ _ _ H' H0).
apply comp_inc_compat_ab_a.
apply (fun H' ⇒ @inc_trans _ _ _ _ H' H3).
apply (comp_inc_compat_ab_a H0).
split.
apply H2.
apply H3.
apply (comp_inc_compat_ab_a H0).
move ⇒ g.
elim.
elim ⇒ H5 H6 H7.
rewrite H7 comp_assoc.
apply inc_antisym.
rewrite inv_invol in H1.
apply (comp_inc_compat_ab_a H1).
apply (comp_inc_compat_a_ab H).
Qed.

```

Lemma 164 (injection_unique2) *Let $m : B \rightarrowtail A$ be an injection, $f : C \rightarrow A$ be a function and $f^\# \cdot f = m^\# \cdot m$, then function $f \cdot m^\#$ is a surjection.*

Lemma *injection_unique2* {A B C : eqType} {m : Rel B A} {f : Rel C A}:
 injection_r m → function_r f → (f # · f) = (m # · m) → surjection_r (f · m #).

Proof.

```

rewrite /surjection_r/injection_r/function_r/total_r/univalent_r.
elim.
elim ⇒ H H0 H1.
elim ⇒ H2 H3 H4.
repeat split.
rewrite comp_inv inv_invol comp_assoc -(@comp_assoc _ _ _ _ m).
apply (@inc_trans _ _ _ (f · ((f # · f) · f #))).
rewrite comp_assoc -comp_assoc.
apply (@inc_trans _ _ _ _ H2).
apply (comp_inc_compat_a_ab H2).
apply comp_inc_compat_ab_ab'.

```

```

rewrite  $H_4$ .
apply inc_refl.
rewrite comp_inv comp_assoc -(@comp_assoc _ _ _ _  $f$ ).
apply (fun  $H' \Rightarrow$  @inc_trans _ _ _ _  $H' H1$ ).
apply comp_inc_compat_ab_ab'.
apply (comp_inc_compat_ab_b  $H3$ ).
rewrite inv_invol comp_inv inv_invol comp_assoc -(@comp_assoc _ _ _ _  $f$ ).
apply (@inc_trans _ _ _ _  $H$ ).
apply comp_inc_compat_ab_ab'.
rewrite  $H_4$  comp_assoc.
apply (comp_inc_compat_a_ab  $H$ ).
Qed.

```

Lemma 165 (*bijection_inv*) *Let $\alpha : A \rightarrow B$, $\beta : B \rightarrow A$, $\alpha \cdot \beta = id_A$ and $\beta \cdot \alpha = id_B$, then α and β are bijections and $\beta = \alpha^\#$.*

Lemma *bijection_inv* { $A\ B : eqType$ } { $\alpha : Rel\ A\ B$ } { $\beta : Rel\ B\ A$ }:
 $\alpha \cdot \beta = Id\ A \rightarrow \beta \cdot \alpha = Id\ B \rightarrow$ *bijection_r* $\alpha \wedge$ *bijection_r* $\beta \wedge$
 $\beta = \alpha^\#$.

Proof.

```

move  $\Rightarrow$   $H\ H0$ .
move : (@id_function  $A$ )  $\Rightarrow$   $H1$ .
move : (@id_function  $B$ )  $\Rightarrow$   $H2$ .
assert (bijection_r  $\alpha \wedge$  bijection_r  $\beta$ ).
assert (total_r  $\alpha \wedge$  total_r ( $\alpha^\#$ )  $\wedge$  total_r  $\beta \wedge$  total_r ( $\beta^\#$ )).
repeat split.
apply (@total_comp2 _ _ _ _  $\beta$ ).
rewrite  $H$ .
apply  $H1$ .
apply (@total_comp2 _ _ _ _ ( $\beta^\#$ )).
rewrite -comp_inv  $H0$  inv_id.
apply  $H2$ .
apply (@total_comp2 _ _ _ _  $\alpha$ ).
rewrite  $H0$ .
apply  $H2$ .
apply (@total_comp2 _ _ _ _ ( $\alpha^\#$ )).
rewrite -comp_inv  $H$  inv_id.
apply  $H1$ .
repeat split.
apply  $H3$ .
apply (@univalent_comp2 _ _ _ _  $\beta$ ).
rewrite  $H0$ .
apply  $H2$ .

```

```

apply H3.
apply H3.
apply (@univalent_comp2 _ _ _ (beta #)).
rewrite -comp_inv H inv_id.
apply H1.
rewrite inv_invol.
apply H3.
apply H3.
apply (@univalent_comp2 _ _ _ alpha).
rewrite H.
apply H1.
apply H3.
apply H3.
apply (@univalent_comp2 _ _ _ (alpha #)).
rewrite -comp_inv H0 inv_id.
apply H2.
rewrite inv_invol.
apply H3.
split.
apply H3.
split.
apply H3.
rewrite -(@comp_id_r _ _ beta) -(@comp_id_l _ _ (alpha #)).
rewrite -H0 comp_assoc.
apply f_equal.
apply inc_antisym.
apply H3.
rewrite comp_inv_inv -inv_inc_move inv_id.
apply H3.
Qed.

```

Lemma 166 (bijection_inv_corollary) *Let $\alpha : A \rightarrow B$ be a bijection, then $\alpha^\#$ is also a bijection.*

Lemma *bijection_inv_corollary* $\{A\ B : \text{eqType}\} \{\alpha : \text{Rel } A\ B\}$:
bijection_r alpha \rightarrow bijection_r (alpha #).

Proof.

```

move : (@bijection_inv _ _ alpha (alpha #))  $\Rightarrow$  H.
move  $\Rightarrow$  H0.
rewrite /bijection_r/function_r/total_r/univalent_r in H0.
rewrite inv_invol in H0.
apply H.

```

apply *inc_antisym*.

apply *H0*.

apply *H0*.

apply *inc_antisym*.

apply *H0*.

apply *H0*.

Qed.

Chapter 7

Library **Dedekind**

```
Require Import Basic_Notations.  
Require Import Basic_Lemmas.  
Require Import Relation_Properties.  
Require Import Functions_Mappings.
```

7.1 Dedekind formula に関する補題

Lemma 167 (dedekind1) *Let $\alpha : A \rightarrow B$, $\beta : B \rightarrow C$ and $\gamma : A \rightarrow C$. Then*

$$\alpha \cdot \beta \sqcap \gamma \sqsubseteq \alpha \cdot (\beta \sqcap \alpha^\# \cdot \gamma).$$

Lemma dedekind1

```
{A B C : eqType} {alpha : Rel A B} {beta : Rel B C} {gamma : Rel A C} :  
((alpha · beta) gamma) (alpha · (beta (alpha # · gamma)))
```

Proof.

```
apply (@inc_trans _ _ _ _ (@dedekind _ _ _ _ _)).  
apply comp_inc_compat_ab_a'b.  
apply cap_l.
```

Qed.

Lemma 168 (dedekind2) *Let $\alpha : A \rightarrow B$, $\beta : B \rightarrow C$ and $\gamma : A \rightarrow C$. Then*

$$\alpha \cdot \beta \sqcap \gamma \sqsubseteq (\alpha \sqcap \gamma \cdot \beta^\#) \cdot \beta.$$

Lemma dedekind2

```
{A B C : eqType} {alpha : Rel A B} {beta : Rel B C} {gamma : Rel A C} :  
((alpha · beta) gamma) ((alpha (gamma · beta #)) · beta)
```

Proof.

```
apply (@inc_trans _ _ _ _ (@dedekind _ _ _ _ _)).
```

CHAPTER 7. LIBRARY DEDEKIND

apply *comp_inc_compat_ab_ab'*.
 apply *cap_l*.
 Qed.

Lemma 169 (relation_rel_inv_rel) *Let $\alpha : A \rightarrow B$. Then*

$$\alpha \sqsubseteq \alpha \cdot \alpha^\# \cdot \alpha.$$

Lemma *relation_rel_inv_rel* {*A B : eqType*} {*alpha : Rel A B*}:
alpha ((*alpha* · *alpha* #) · *alpha*).

Proof.

move : (@dedekind1 _ _ _ *alpha* (*Id B*) *alpha*) \Rightarrow *H*.
 rewrite *comp_id_r cap_idem* in *H*.
 apply (@*inc_trans* _ _ _ _ *H*).
 rewrite *comp_assoc*.
 apply *comp_inc_compat_ab_ab'*.
 apply *cap_r*.
 Qed.

7.2 Dedekind formula と全関係

Lemma 170 (dedekind_universal1) *Let $\alpha : B \rightarrow C$. Then*

$$\nabla_{AC} \cdot \alpha^\# \cdot \alpha = \nabla_{AB} \cdot \alpha.$$

Lemma *dedekind_universal1* {*A B C : eqType*} {*alpha : Rel B C*}:
 (*A C* · *alpha* #) · *alpha* = *A B* · *alpha*.

Proof.

apply *inc_antisym*.
 apply *comp_inc_compat_ab_a'b*.
 apply *inc_alpha_universal*.
 apply (@*inc_trans* _ _ _ (*A B* · ((*alpha* · *alpha* #) · *alpha*))).
 apply *comp_inc_compat_ab_ab'*.
 apply *relation_rel_inv_rel*.
 rewrite -*comp_assoc* -*comp_assoc*.
 apply *comp_inc_compat_ab_a'b*.
 apply *comp_inc_compat_ab_a'b*.
 apply *inc_alpha_universal*.
 Qed.

Lemma 171 (`dedekind_universal2a`, `dedekind_universal2b`, `dedekind_universal2c`) *Let $\alpha : A \rightarrow B$ and $\beta : C \rightarrow B$. Then*

$$\nabla_{IC} \cdot \beta \sqsubseteq \nabla_{IA} \cdot \alpha \Leftrightarrow \nabla_{CC} \cdot \beta \sqsubseteq \nabla_{CA} \cdot \alpha \Leftrightarrow \beta \sqsubseteq \beta \cdot \alpha^\# \cdot \alpha.$$

Lemma `dedekind_universal2a` $\{A\ B\ C : \text{eqType}\} \{\alpha : \text{Rel } A\ B\} \{\beta : \text{Rel } C\ B\} :$
 $(\ i\ C \cdot \beta) \quad (\ i\ A \cdot \alpha) \rightarrow (\ C\ C \cdot \beta) \quad (\ C\ A \cdot \alpha).$

Proof.

`move` $\Rightarrow H$.

`rewrite` `-unit_universal` `-(@lemma_for_tarski2 C A)`.

`rewrite` `comp_assoc comp_assoc`.

`apply` `(comp_inc_compat_ab_ab' H)`.

Qed.

Lemma `dedekind_universal2b` $\{A\ B\ C : \text{eqType}\} \{\alpha : \text{Rel } A\ B\} \{\beta : \text{Rel } C\ B\} :$
 $(\ C\ C \cdot \beta) \quad (\ C\ A \cdot \alpha) \rightarrow \beta \quad ((\beta \cdot \alpha^\#) \cdot \alpha).$

Proof.

`move` $\Rightarrow H$.

`apply` `(@inc_trans _ _ _ (\beta (\ C\ C \cdot \beta)))`.

`apply` `inc_cap`.

`split`.

`apply` `inc_refl`.

`apply` `comp_inc_compat_b_ab`.

`apply` `inc_alpha_universal`.

`apply` `(@inc_trans _ _ _ (\beta (\ C\ A \cdot \alpha)))`.

`apply` `(cap_inc_compat_l H)`.

`rewrite` `cap_comm`.

`apply` `(@inc_trans _ _ _ _ (@dedekind2 _ _ _ _ _))`.

`apply` `comp_inc_compat_ab_a'b`.

`apply` `cap_r`.

Qed.

Lemma `dedekind_universal2c` $\{A\ B\ C : \text{eqType}\} \{\alpha : \text{Rel } A\ B\} \{\beta : \text{Rel } C\ B\} :$
 $\beta \quad ((\beta \cdot \alpha^\#) \cdot \alpha) \rightarrow (\ i\ C \cdot \beta) \quad (\ i\ A \cdot \alpha).$

Proof.

`move` $\Rightarrow H$.

`apply` `(@inc_trans _ _ _ (\ i\ C \cdot ((\beta \cdot \alpha^\#) \cdot \alpha)))`.

`apply` `(comp_inc_compat_ab_ab' H)`.

`rewrite` `-comp_assoc`.

`apply` `comp_inc_compat_ab_a'b`.

`apply` `inc_alpha_universal`.

Qed.

Lemma 172 (dedekind_universal3a, dedekind_universal3b) *Let $\alpha : A \rightarrow B$ and $\beta : A \rightarrow C$. Then*

$$\beta \cdot \nabla_{CI} \sqsubseteq \alpha \cdot \nabla_{BI} \Leftrightarrow \beta \cdot \nabla_{CC} \sqsubseteq \alpha \cdot \nabla_{BC} \Leftrightarrow \beta \sqsubseteq \alpha \cdot \alpha^\# \cdot \beta.$$

Lemma dedekind_universal3a $\{A\ B\ C : \text{eqType}\} \{\alpha : \text{Rel } A\ B\} \{\beta : \text{Rel } A\ C\} :$
 $(\beta \cdot C\ i) \quad (\alpha \cdot B\ i) \Leftrightarrow (\beta \cdot C\ C) \quad (\alpha \cdot B\ C).$

Proof.

split; move $\Rightarrow H$.

apply *inv_inc_invol*.

rewrite *comp_inv comp_inv inv_universal inv_universal*.

apply *dedekind_universal2a*.

apply *inv_inc_invol*.

rewrite *comp_inv comp_inv inv_invol inv_invol inv_universal inv_universal*.

apply *H*.

apply *inv_inc_invol*.

rewrite *comp_inv comp_inv inv_universal inv_universal*.

apply *dedekind_universal2c*.

apply *dedekind_universal2b*.

apply *inv_inc_invol*.

rewrite *comp_inv comp_inv inv_invol inv_invol inv_universal inv_universal*.

apply *H*.

Qed.

Lemma dedekind_universal3b $\{A\ B\ C : \text{eqType}\} \{\alpha : \text{Rel } A\ B\} \{\beta : \text{Rel } A\ C\} :$
 $(\beta \cdot C\ i) \quad (\alpha \cdot B\ i) \Leftrightarrow \beta \quad ((\alpha \cdot \alpha^\#) \cdot \beta).$

Proof.

split; move $\Rightarrow H$.

apply *inv_inc_invol*.

rewrite *comp_inv comp_inv -comp_assoc*.

apply *dedekind_universal2b*.

apply *dedekind_universal2a*.

apply *inv_inc_invol*.

rewrite *comp_inv comp_inv inv_invol inv_invol inv_universal inv_universal*.

apply *H*.

apply *inv_inc_invol*.

rewrite *comp_inv comp_inv inv_universal inv_universal*.

apply *dedekind_universal2c*.

rewrite *-comp_inv -comp_inv -comp_assoc*.

apply *inc_inv*.

apply *H*.

Qed.

Lemma 173 (universal_total) *Let $\alpha : A \rightarrow B$. Then*

$$\alpha \cdot \nabla_{BI} = \nabla_{AI} \Leftrightarrow \text{“}\alpha \text{ is total”}.$$

Lemma *universal_total* { $A B : eqType$ } { $\alpha : Rel A B$ }:
 $\alpha \cdot \nabla_{BI} = \nabla_{AI} \Leftrightarrow total_r \alpha$.

Proof.

```
move : (@dedekind_universal3b _ _ _ alpha (Id A)) => H.
rewrite comp_id_l comp_id_r in H.
rewrite /total_r.
rewrite -H.
split; move => H0.
rewrite H0.
apply inc_refl.
apply inc_antisym.
apply inc_alpha_universal.
apply H0.
Qed.
```

7.3 Dedekind formula と恒等関係

Lemma 174 (dedekind_id1) *Let $\alpha : A \rightarrow A$. Then*

$$\alpha \sqsubseteq id_A \Rightarrow \alpha^\# = \alpha.$$

Lemma *dedekind_id1* { $A : eqType$ } { $\alpha : Rel A A$ }: $\alpha \sqsubseteq Id A \rightarrow \alpha^\# = \alpha$.

Proof.

```
move => H.
assert (alpha # alpha).
move : (@dedekind1 _ _ _ (alpha #) (Id A) (Id A)) => H0.
rewrite comp_id_r comp_id_r inv_invol in H0.
replace (alpha # Id A) with (alpha #) in H0.
replace (Id A alpha) with alpha in H0.
apply (@inc_trans _ _ _ (alpha # • alpha)).
apply H0.
apply comp_inc_compat_ab_b.
rewrite -inv_inc_move.
rewrite inv_id.
apply H.
rewrite cap_comm.
apply inc_def1.
```

```

apply H.
apply inc_def1.
rewrite -inv_inc_move.
rewrite inv_id.
apply H.
apply inc_antisym.
apply H0.
apply inv_inc_move.
apply H0.
Qed.

```

Lemma 175 (dedekind_id2) *Let $\alpha : A \rightarrow A$. Then*

$$\alpha \sqsubseteq id_A \Rightarrow \alpha \cdot \alpha = \alpha.$$

Lemma dedekind_id2 $\{A : eqType\} \{alpha : Rel\ A\ A\}$:
 $alpha \quad Id\ A \rightarrow alpha \cdot alpha = alpha.$

Proof.

```

move  $\Rightarrow$  H.
apply inc_antisym.
apply (comp_inc_compat_ab_a H).
move : (dedekind_id1 H)  $\Rightarrow$  H0.
apply (@inc_trans _ _ _ ((alpha  $\cdot$  Id A) Id A)).
rewrite comp_id_r.
apply inc_cap.
split.
apply inc_refl.
apply H.
apply (@inc_trans _ _ _ _ (@dedekind1 _ _ _ _ _)).
apply comp_inc_compat_ab_ab'.
rewrite H0 comp_id_r.
apply cap_r.
Qed.

```

Lemma 176 (dedekind_id3) *Let $\alpha, \beta : A \rightarrow A$. Then*

$$\alpha \sqsubseteq id_A \wedge \beta \sqsubseteq id_A \Rightarrow \alpha \cdot \beta = \alpha \sqcap \beta.$$

Lemma dedekind_id3 $\{A : eqType\} \{alpha\ beta : Rel\ A\ A\}$:
 $alpha \quad Id\ A \rightarrow beta \quad Id\ A \rightarrow alpha \cdot beta = alpha \sqcap beta.$

Proof.

```

move  $\Rightarrow$  H H0.
apply inc_antisym.

```

```

apply inc_cap.
split.
apply (comp_inc_compat_ab_a H0).
apply (comp_inc_compat_ab_b H).
replace (alpha beta) with ((alpha beta) • (alpha beta)).
apply comp_inc_compat.
apply cap_l.
apply cap_r.
apply dedekind_id2.
apply (fun H' => @inc_trans _ _ _ _ H' H).
apply cap_l.
Qed.

```

Lemma 177 (dedekind_id4) *Let $\alpha, \beta : A \rightarrow A$. Then*

$$\alpha \sqsubseteq id_A \wedge \beta \sqsubseteq id_A \Rightarrow (\alpha \triangleright \beta) \sqcap id_A = (\alpha \Rightarrow \beta) \sqcap id_A.$$

Lemma dedekind_id4 $\{A : eqType\} \{alpha\ beta : Rel\ A\ A\}$:
 $alpha\ Id\ A \rightarrow beta\ Id\ A \rightarrow (alpha\ beta)\ Id\ A = (alpha \gg beta)\ Id\ A.$

Proof.

```

move => H H0.
apply inc_lower.
move => gamma.
rewrite inc_cap inc_cap.
split; elim => H1 H2.
split.
rewrite inc_rpc cap_comm.
rewrite -(@dedekind_id3 _ _ _ H H2).
rewrite -(@dedekind_id1 _ _ H).
apply inc_residual.
apply H1.
apply H2.
split.
rewrite inc_residual (@dedekind_id1 _ _ H) (@dedekind_id3 _ _ _ H H2).
rewrite cap_comm -inc_rpc.
apply H1.
apply H2.
Qed.

```

Chapter 8

Library **Residual**

```
Require Import Basic_Notations.
Require Import Basic_Lemmas.
Require Import Relation_Properties.
Require Import Functions_Mappings.
Require Import Dedekind.
Require Import Logic.FunctionalExtensionality.
```

8.1 剰余合成関係の性質

8.1.1 基本的な性質

Lemma 178 (double_residual) *Let $\alpha : A \rightarrow B$, $\beta : B \rightarrow C$ and $\gamma : C \rightarrow D$. Then*

$$\alpha \triangleright (\beta \triangleright \gamma) = (\alpha \cdot \beta) \triangleright \gamma.$$

Lemma *double_residual*

```
{A B C D : eqType} {alpha : Rel A B} {beta : Rel B C} {gamma : Rel C D}:
alpha (beta gamma) = (alpha • beta) gamma.
```

Proof.

```
apply inc_lower.
move => delta.
split; move => H.
apply inc_residual.
rewrite comp_inv comp_assoc.
rewrite -inc_residual -inc_residual.
apply H.
rewrite inc_residual inc_residual.
rewrite -comp_assoc -comp_inv.
apply inc_residual.
```

apply *H*.

Qed.

Lemma 179 (residual_to_complement) *Let $\alpha : A \rightarrow B$ and $\beta : B \rightarrow C$. Then*

$$\alpha \triangleright \beta = (\alpha \cdot \beta^-)^-.$$

Lemma *residual_to_complement* {*A B C : eqType*} {*alpha : Rel A B*} {*beta : Rel B C*}:
alpha beta = (alpha • beta ^) ^.

Proof.

apply *inc_lower*.

move \Rightarrow *gamma*.

split; move \Rightarrow *H*.

rewrite *bool_lemma2 complement_invol cap_comm*.

apply *inc_antisym*.

apply (@*inc_trans* _ _ _ _ (@*dedekind1* _ _ _ _ _)).

replace (*beta* ^ (alpha # • *gamma*)) with (*B C*).

rewrite *comp_empty_r*.

apply *inc_refl*.

apply *Logic.eq_sym*.

rewrite *cap_comm*.

apply *bool_lemma2*.

apply *inc_residual*.

apply *H*.

apply *inc_empty_alpha*.

apply *inc_residual*.

apply *bool_lemma2*.

apply *inc_antisym*.

apply (@*inc_trans* _ _ _ _ (@*dedekind1* _ _ _ _ _)).

rewrite *inv_invol*.

replace (*gamma* (alpha • *beta* ^)) with (*A C*).

rewrite *comp_empty_r*.

apply *inc_refl*.

apply *Logic.eq_sym*.

rewrite -(@*complement_invol* _ _ (alpha • *beta* ^)).

apply *bool_lemma2*.

apply *H*.

apply *inc_empty_alpha*.

Qed.

CHAPTER 8. LIBRARY RESIDUAL

Lemma 180 (inv_residual_inc) *Let $\alpha : A \rightarrow B$ and $\beta : B \rightarrow C$. Then*

$$\alpha^\# \cdot (\alpha \triangleright \beta) \sqsubseteq \beta.$$

Lemma *inv_residual_inc* { $A\ B\ C : eqType$ } { $\alpha : Rel\ A\ B$ } { $\beta : Rel\ B\ C$ }:
 $\alpha^\# \cdot (\alpha \triangleright \beta) \sqsubseteq \beta$.

Proof.

apply *inc_residual*.

apply *inc_refl*.

Qed.

Lemma 181 (inc_residual_inv) *Let $\alpha : A \rightarrow B$ and $\gamma : A \rightarrow C$. Then*

$$\gamma \sqsubseteq \alpha \triangleright \alpha^\# \cdot \gamma.$$

Lemma *inc_residual_inv* { $A\ B\ C : eqType$ } { $\alpha : Rel\ A\ B$ } { $\gamma : Rel\ A\ C$ }:
 $\gamma \sqsubseteq \alpha \triangleright (\alpha^\# \cdot \gamma)$.

Proof.

apply *inc_residual*.

apply *inc_refl*.

Qed.

Lemma 182 (id_inc_residual) *Let $\alpha : A \rightarrow B$. Then*

$$id_A \sqsubseteq \alpha \triangleright \alpha^\#.$$

Lemma *id_inc_residual* { $A\ B : eqType$ } { $\alpha : Rel\ A\ B$ }: $Id\ A \sqsubseteq \alpha \triangleright (\alpha^\#)$.

Proof.

apply *inc_residual*.

rewrite *comp_id_r*.

apply *inc_refl*.

Qed.

Lemma 183 (residual_universal) *Let $\alpha : A \rightarrow B$. Then*

$$\alpha \triangleright \nabla_{BC} = \nabla_{AC}.$$

Lemma *residual_universal* { $A\ B\ C : eqType$ } { $\alpha : Rel\ A\ B$ }: $\alpha \triangleright \nabla_{BC} = \nabla_{AC}$.

Proof.

apply *inc_antisym*.

apply *inc_alpha_universal*.

apply *inc_residual*.

apply *inc_alpha_universal*.

Qed.

8.1.2 単調性と分配法則

Lemma 184 (residual_inc_compat) *Let $\alpha, \alpha' : A \rightarrow B$ and $\beta, \beta' : B \rightarrow C$. Then*

$$\alpha' \sqsubseteq \alpha \wedge \beta \sqsubseteq \beta' \Rightarrow \alpha \triangleright \beta \sqsubseteq \alpha' \triangleright \beta'.$$

Lemma residual_inc_compat

$\{A\ B\ C : eqType\} \{alpha\ alpha' : Rel\ A\ B\} \{beta\ beta' : Rel\ B\ C\}:$
 $alpha' \quad alpha \rightarrow beta \quad beta' \rightarrow (alpha \quad beta) \quad (alpha' \quad beta').$

Proof.

`move \Rightarrow H H0.`

`apply inc_residual.`

`apply (fun H' \Rightarrow @inc_trans _ _ _ _ H' H0).`

`move : (@inc_refl _ _ (alpha beta)) \Rightarrow H1.`

`apply inc_residual in H1.`

`apply (fun H' \Rightarrow @inc_trans _ _ _ _ H' H1).`

`apply comp_inc_compat_ab_a'b.`

`apply inc_inv.`

`apply H.`

Qed.

Lemma 185 (residual_inc_compat_l) *Let $\alpha : A \rightarrow B$ and $\beta, \beta' : B \rightarrow C$. Then*

$$\beta \sqsubseteq \beta' \Rightarrow \alpha \triangleright \beta \sqsubseteq \alpha \triangleright \beta'.$$

Lemma residual_inc_compat_l

$\{A\ B\ C : eqType\} \{alpha : Rel\ A\ B\} \{beta\ beta' : Rel\ B\ C\}:$
 $beta \quad beta' \rightarrow (alpha \quad beta) \quad (alpha \quad beta').$

Proof.

`move \Rightarrow H.`

`apply (@residual_inc_compat _ _ _ _ _ (@inc_refl _ _ _)) H).`

Qed.

Lemma 186 (residual_inc_compat_r) *Let $\alpha, \alpha' : A \rightarrow B$ and $\beta : B \rightarrow C$. Then*

$$\alpha' \sqsubseteq \alpha \Rightarrow \alpha \triangleright \beta \sqsubseteq \alpha' \triangleright \beta.$$

Lemma residual_inc_compat_r

$\{A\ B\ C : eqType\} \{alpha\ alpha' : Rel\ A\ B\} \{beta : Rel\ B\ C\}:$
 $alpha' \quad alpha \rightarrow (alpha \quad beta) \quad (alpha' \quad beta).$

Proof.

CHAPTER 8. LIBRARY RESIDUAL

move $\Rightarrow H$.
 apply (@residual_inc_compat _ _ _ _ _ H (@inc_refl _ _)).
 Qed.

Lemma 187 (residual_capL_distr, residual_cap_distr) *Let $\alpha : A \rightarrow B$ and $\beta_\lambda : B \rightarrow C$. Then*

$$\alpha \triangleright (\prod_{\lambda \in \Lambda} \beta_\lambda) = \prod_{\lambda \in \Lambda} (\alpha \triangleright \beta_\lambda).$$

Lemma residual_capL_distr

$\{A\ B\ C\ L : \text{eqType}\} \{alpha : \text{Rel } A\ B\} \{beta_L : L \rightarrow \text{Rel } B\ C\}:$
 $alpha \quad (_ \text{beta_} L) = _ (\text{fun } l : L \Rightarrow alpha \quad beta_L\ l).$

Proof.

apply inc_lower.
 move \Rightarrow gamma.
 split; move \Rightarrow H.
 apply inc_capL.
 move \Rightarrow l.
 apply inc_residual.
 move : l.
 apply inc_capL.
 apply inc_residual.
 apply H.
 apply inc_residual.
 apply inc_capL.
 move \Rightarrow l.
 apply inc_residual.
 move : l.
 apply inc_capL.
 apply H.
 Qed.

Lemma residual_cap_distr

$\{A\ B\ C : \text{eqType}\} \{alpha : \text{Rel } A\ B\} \{beta\ gamma : \text{Rel } B\ C\}:$
 $alpha \quad (beta \quad gamma) = (alpha \quad beta) \quad (alpha \quad gamma).$

Proof.

rewrite cap_to_capL cap_to_capL.
 rewrite residual_capL_distr.
 apply f_equal.
 apply functional_extensionality.
 induction x.
 by [].
 by [].
 Qed.

CHAPTER 8. LIBRARY RESIDUAL

Lemma 188 (residual_cupL_distr, residual_cup_distr) *Let $\alpha_\lambda : A \rightarrow B$ and $\beta : B \rightarrow C$. Then*

$$(\sqcup_{\lambda \in \Lambda} \alpha_\lambda) \triangleright \beta = \sqcap_{\lambda \in \Lambda} (\alpha_\lambda \triangleright \beta).$$

Lemma residual_cupL_distr

$\{A\ B\ C\ L : \text{eqType}\} \{\text{beta} : \text{Rel}\ B\ C\} \{\text{alpha_L} : L \rightarrow \text{Rel}\ A\ B\}:$
 $(\text{_} \text{alpha_L}) \quad \text{beta} = \text{_} (\text{fun } l : L \Rightarrow \text{alpha_L } l \quad \text{beta}).$

Proof.

apply *inc_lower*.
 move \Rightarrow *gamma*.
 split; move \Rightarrow *H*.
 apply *inc_capL*.
 move \Rightarrow *l*.
 apply *inc_residual*.
 move : *l*.
 apply *inc_cupL*.
 rewrite *-comp_cupL_distr_r -inv_cupL_distr*.
 apply *inc_residual*.
 apply *H*.
 apply *inc_residual*.
 rewrite *inv_cupL_distr comp_cupL_distr_r*.
 apply *inc_cupL*.
 move \Rightarrow *l*.
 apply *inc_residual*.
 move : *l*.
 apply *inc_capL*.
 apply *H*.

Qed.

Lemma residual_cup_distr

$\{A\ B\ C : \text{eqType}\} \{\text{alpha } \text{beta} : \text{Rel}\ A\ B\} \{\text{gamma} : \text{Rel}\ B\ C\}:$
 $(\text{alpha } \quad \text{beta}) \quad \text{gamma} = (\text{alpha } \quad \text{gamma}) \quad (\text{beta } \quad \text{gamma}).$

Proof.

rewrite *cup_to_cupL cap_to_capL*.
 rewrite *residual_cupL_distr*.
 apply *f_equal*.
 apply *functional_extensionality*.
 induction *x*.

by \square .

by \square .

Qed.

8.1.3 剰余合成と関数

Lemma 189 (total_residual) *Let $\alpha : A \rightarrow B$ be a total relation and $\beta : B \rightarrow C$. Then*

$$\alpha \triangleright \beta \sqsubseteq \alpha \cdot \beta.$$

Lemma *total_residual* {A B C : eqType} {alpha : Rel A B} {beta : Rel B C}:
total_r alpha \rightarrow (alpha beta) (alpha beta).

Proof.

move \Rightarrow H.

apply (@inc_trans _ _ _ ((alpha alpha #) (alpha beta))).

apply (comp_inc_compat_b_ab H).

rewrite comp_assoc.

apply comp_inc_compat_ab_ab'.

apply inv_residual_inc.

Qed.

Lemma 190 (univalent_residual) *Let $\alpha : A \rightarrow B$ be a univalent relation and $\beta : B \rightarrow C$. Then*

$$\alpha \cdot \beta \sqsubseteq \alpha \triangleright \beta.$$

Lemma *univalent_residual* {A B C : eqType} {alpha : Rel A B} {beta : Rel B C}:
univalent_r alpha \rightarrow (alpha beta) (alpha beta).

Proof.

move \Rightarrow H.

apply (@inc_trans _ _ _ _ (@inc_residual_inv _ _ alpha _)).

apply residual_inc_compat_l.

rewrite -comp_assoc.

apply (comp_inc_compat_ab_b H).

Qed.

Lemma 191 (function_residual1) *Let $\alpha : A \rightarrow B$ be a function and $\beta : B \rightarrow C$. Then*

$$\alpha \triangleright \beta = \alpha \cdot \beta.$$

Lemma *function_residual1* {A B C : eqType} {alpha : Rel A B} {beta : Rel B C}:
function_r alpha \rightarrow alpha beta = alpha beta.

Proof.

elim \Rightarrow H H0.

apply inc_antisym.

apply (total_residual H).

apply (univalent_residual H0).

Qed.

CHAPTER 8. LIBRARY RESIDUAL

Lemma 192 (residual_id) *Let $\alpha : A \rightarrow B$. Then*

$$id_A \triangleright \alpha = \alpha.$$

Lemma *residual_id* $\{A\ B : eqType\} \{alpha : Rel\ A\ B\}$:

Id A alpha = alpha.

Proof.

`move : (@function_residual1 _ _ _ (Id A) alpha (@id_function A)) \Rightarrow H.`

`rewrite comp_id_l in H.`

`apply H.`

Qed.

Lemma 193 (universal_residual) *Let $\alpha : A \rightarrow B$. Then*

$$\nabla_{AA} \triangleright \alpha \sqsubseteq \alpha.$$

Lemma *universal_residual* $\{A\ B : eqType\} \{alpha : Rel\ A\ B\}$:

A A alpha alpha.

Proof.

`apply (@inc_trans _ _ _ (Id A alpha)).`

`apply residual_inc_compat_r.`

`apply inc_alpha_universal.`

`rewrite residual_id.`

`apply inc_refl.`

Qed.

Lemma 194 (function_residual2) *Let $\alpha : A \rightarrow B$ be a function, $\beta : B \rightarrow C$ and $\gamma : C \rightarrow D$. Then*

$$\alpha \cdot (\beta \triangleright \gamma) = (\alpha \cdot \beta) \triangleright \gamma.$$

Lemma *function_residual2*

$\{A\ B\ C\ D : eqType\} \{alpha : Rel\ A\ B\} \{beta : Rel\ B\ C\} \{gamma : Rel\ C\ D\}$:

function_r alpha \rightarrow alpha \cdot (beta gamma) = (alpha \cdot beta) gamma.

Proof.

`move \Rightarrow H.`

`rewrite -(@function_residual1 _ _ _ _ H).`

`apply double_residual.`

Qed.

CHAPTER 8. LIBRARY RESIDUAL

Lemma 195 (function_residual3) *Let $\alpha : A \rightarrow B$, $\beta : B \rightarrow C$ be relations and $\gamma : D \rightarrow C$ be a function. Then*

$$(\alpha \triangleright \beta) \cdot \gamma^\# = \alpha \triangleright (\beta \cdot \gamma^\#).$$

Lemma function_residual3

$\{A\ B\ C\ D : \text{eqType}\} \{alpha : \text{Rel}\ A\ B\} \{beta : \text{Rel}\ B\ C\} \{gamma : \text{Rel}\ D\ C\} :$
 $\text{function_r}\ gamma \rightarrow (alpha \quad beta) \cdot gamma \# = alpha \quad (beta \cdot gamma \#).$

Proof.

move $\Rightarrow H$.
 apply inc_lower.
 move $\Rightarrow \text{delta}$.
 split; move $\Rightarrow H0$.
 apply inc_residual.
 rewrite -(@function_move2 _ _ _ _ _ H).
 rewrite comp_assoc.
 apply inc_residual.
 rewrite (@function_move2 _ _ _ _ _ H).
 apply H0.
 rewrite -(@function_move2 _ _ _ _ _ H).
 apply inc_residual.
 rewrite -comp_assoc.
 rewrite (@function_move2 _ _ _ _ _ H).
 apply inc_residual.
 apply H0.
Qed.

Lemma 196 (function_residual4) *Let $\alpha : A \rightarrow B$, $\gamma : C \rightarrow D$ be relations and $\beta : B \rightarrow C$ be a function. Then*

$$\alpha \cdot \beta \triangleright \gamma = \alpha \triangleright \beta \cdot \gamma.$$

Lemma function_residual4

$\{A\ B\ C\ D : \text{eqType}\} \{alpha : \text{Rel}\ A\ B\} \{beta : \text{Rel}\ B\ C\} \{gamma : \text{Rel}\ C\ D\} :$
 $\text{function_r}\ beta \rightarrow (alpha \cdot beta) \quad gamma = alpha \quad (beta \cdot gamma).$

Proof.

move $\Rightarrow H$.
 rewrite -double_residual.
 by [rewrite (function_residual1 H)].
Qed.

8.2 Galois 同値とその系

Lemma 197 (galois) *Let $\alpha : A \rightarrow B$, $\beta : B \rightarrow C$ and $\gamma : A \rightarrow C$. Then*

$$\gamma \sqsubseteq \alpha \triangleright \beta \Leftrightarrow \alpha \sqsubseteq \gamma \triangleright \beta^\#.$$

Lemma galois $\{A\ B\ C : eqType\} \{alpha : Rel\ A\ B\} \{beta : Rel\ B\ C\} \{gamma : Rel\ A\ C\} :$
 $gamma \quad (alpha \quad beta) \leftrightarrow alpha \quad (gamma \quad beta \#).$

Proof.

split; move $\Rightarrow H$.
 apply inc_residual.
 apply inv_inc_move.
 rewrite comp_inv inv_invol.
 apply inc_residual.
 apply H.
 apply inc_residual.
 apply inv_inc_invol.
 rewrite comp_inv inv_invol.
 apply inc_residual.
 apply H.

Qed.

Lemma 198 (galois_corollary1) *Let $\alpha : A \rightarrow B$ and $\beta : B \rightarrow C$. Then*

$$\alpha \sqsubseteq (\alpha \triangleright \beta) \triangleright \beta^\#.$$

Lemma galois_corollary1 $\{A\ B\ C : eqType\} \{alpha : Rel\ A\ B\} \{beta : Rel\ B\ C\} :$
 $alpha \quad ((alpha \quad beta) \quad beta \#).$

Proof.

rewrite -galois.
 apply inc_refl.

Qed.

Lemma 199 (galois_corollary2) *Let $\alpha : A \rightarrow B$ and $\beta : B \rightarrow C$. Then*

$$((\alpha \triangleright \beta) \triangleright \beta^\#) \triangleright \beta = \alpha \triangleright \beta.$$

Lemma galois_corollary2 $\{A\ B\ C : eqType\} \{alpha : Rel\ A\ B\} \{beta : Rel\ B\ C\} :$
 $((alpha \quad beta) \quad beta \#) \quad beta = alpha \quad beta.$

Proof.

apply inc_antisym.
 apply residual_inc_compat_r.

CHAPTER 8. LIBRARY RESIDUAL

```

apply galois_corollary1.
move : (@galois_corollary1 _ _ _ (alpha beta) (beta #)) => H.
rewrite inv_invol in H.
apply H.
Qed.

```

Lemma 200 (galois_corollary3) *Let $\alpha : A \rightarrow B$ and $\beta : B \rightarrow C$. Then*

$$\alpha = (\alpha \triangleright \beta) \triangleright \beta^\# \Leftrightarrow \exists \gamma : A \rightarrow C, \alpha = \gamma \triangleright \beta^\#.$$

Lemma galois_corollary3 $\{A\ B\ C : eqType\} \{alpha : Rel\ A\ B\} \{beta : Rel\ B\ C\}$:
 $alpha = (alpha\ \beta)\ \beta^\# \Leftrightarrow (\exists\ gamma : Rel\ A\ C, alpha = gamma\ \beta^\#)$.
Proof.

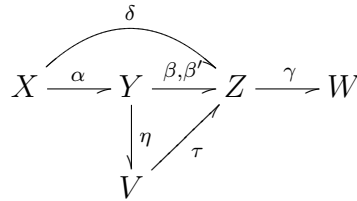
```

split; move => H.
exists (alpha beta).
apply H.
elim H => gamma H0.
rewrite H0.
move : (@galois_corollary2 _ _ _ gamma (beta #)) => H1.
rewrite inv_invol in H1.
by [rewrite H1].
Qed.

```

8.3 その他の性質

この節では、特記が無い限り、記号は以下の図式に従って割り振られるものとする。



Lemma 201 (residual_property1)

$$(\alpha \triangleright \beta) \cdot \gamma \sqsubseteq \alpha \triangleright \beta \cdot \gamma.$$

Lemma residual_property1
 $\{W\ X\ Y\ Z : eqType\} \{alpha : Rel\ X\ Y\} \{beta : Rel\ Y\ Z\} \{gamma : Rel\ Z\ W\}$:
 $((alpha\ \beta) \cdot gamma)\ (alpha\ (\beta \cdot gamma))$.
Proof.

CHAPTER 8. LIBRARY RESIDUAL

```

apply (@inc_trans _ _ _ (alpha (alpha # • ((alpha beta) • gamma)))).
apply inc_residual_inv.
apply residual_inc_compat_l.
rewrite -comp_assoc.
apply comp_inc_compat_ab_a'b.
apply inv_residual_inc.
Qed.

```

Lemma 202 (residual_property2)

$$(\alpha \triangleright \beta) \cdot (\beta^\# \triangleright \eta) \sqsubseteq \alpha \triangleright \eta.$$

Lemma residual_property2

```

{V X Y Z : eqType} {alpha : Rel X Y} {beta : Rel Y Z} {eta : Rel Y V}:
((alpha beta) • (beta # eta)) (alpha eta).

```

Proof.

```

apply (@inc_trans _ _ _ _ (@residual_property1 _ _ _ _ _)).
apply residual_inc_compat_l.
move : (@inv_residual_inc _ _ _ (beta # eta)).
by [rewrite inv_invol].
Qed.

```

Lemma 203 (residual_property3)

$$\alpha \triangleright \beta \sqsubseteq \alpha \cdot \eta \triangleright \eta^\# \cdot \beta.$$

Lemma residual_property3

```

{V X Y Z : eqType} {alpha : Rel X Y} {beta : Rel Y Z} {eta : Rel Y V}:
(alpha beta) ((alpha • eta) (eta # • beta)).

```

Proof.

```

apply (@inc_trans _ _ _ _ (@inc_residual_inv _ _ _ (alpha • eta) (alpha beta)))).
apply residual_inc_compat_l.
rewrite comp_inv comp_assoc.
apply comp_inc_compat_ab_ab'.
apply inv_residual_inc.
Qed.

```

Lemma 204 (residual_property4a, residual_property4b)

$$(\alpha \triangleright \beta) \cdot \gamma \sqsubseteq (\alpha \triangleright \beta \cdot \gamma) \sqcap \nabla_{XZ} \cdot \gamma \sqsubseteq (\alpha \triangleright \beta \cdot \gamma) \cdot \gamma^\# \cdot \gamma.$$

Lemma residual_property4a

```

{W X Y Z : eqType} {alpha : Rel X Y} {beta : Rel Y Z} {gamma : Rel Z W}:

```

CHAPTER 8. LIBRARY RESIDUAL

$((\alpha \quad \beta) \cdot \gamma) \quad ((\alpha \quad (\beta \cdot \gamma)) \quad (X \ Z \cdot \gamma)).$

Proof.

rewrite $-(\text{@cap_universal} _ _ (\alpha \quad \beta)).$
 apply $(\text{@inc_trans} _ _ _ _ (\text{@comp_cap_distr_r} _ _ _ _)).$
 apply $\text{cap_inc_compat_r}.$
 apply $\text{residual_property1}.$
Qed.

Lemma residual_property4b

$\{W \ X \ Y \ Z : \text{eqType}\} \{ \alpha : \text{Rel } X \ Y \} \{ \beta : \text{Rel } Y \ Z \} \{ \gamma : \text{Rel } Z \ W \} :$
 $((\alpha \quad (\beta \cdot \gamma)) \quad (X \ Z \cdot \gamma)) \quad ((\alpha \quad (\beta \cdot \gamma)) \cdot (\gamma \# \cdot \gamma)).$

Proof.

rewrite $\text{cap_comm}.$
 apply $(\text{@inc_trans} _ _ _ _ (\text{@dedekind2} _ _ _ _)).$
 rewrite $\text{cap_comm cap_universal comp_assoc}.$
 apply $\text{inc_refl}.$
Qed.

Lemma 205 (residual_property5) *Let τ be a univalent relation. Then,*

$$(\alpha \triangleright \beta) \cdot \tau^\# = (\alpha \triangleright \beta \cdot \tau^\#) \sqcap \nabla_{XZ} \cdot \tau^\#.$$

Lemma residual_property5

$\{V \ X \ Y \ Z : \text{eqType}\} \{ \alpha : \text{Rel } X \ Y \} \{ \beta : \text{Rel } Y \ Z \} \{ \tau : \text{Rel } V \ Z \} :$
 $\text{univalent_r } \tau \rightarrow$
 $(\alpha \quad \beta) \cdot \tau \# = (\alpha \quad (\beta \cdot \tau \#)) \quad (X \ Z \cdot \tau \#).$

Proof.

move $\Rightarrow H.$
 apply $\text{inc_antisym}.$
 rewrite $-(\text{@cap_universal} _ _ (\alpha \quad \beta)).$
 apply $(\text{@inc_trans} _ _ _ _ (\text{@comp_cap_distr_r} _ _ _ _)).$
 apply $\text{cap_inc_compat_r}.$
 apply $\text{residual_property1}.$
 rewrite $\text{cap_comm}.$
 apply $(\text{@inc_trans} _ _ _ _ (\text{@dedekind2} _ _ _ _)).$
 rewrite $\text{cap_comm cap_universal inv_invol}.$
 apply $\text{comp_inc_compat_ab_a'b}.$
 apply $(\text{@inc_trans} _ _ _ _ (\text{@residual_property1} _ _ _ _)).$
 apply $\text{residual_inc_compat_l}.$
 rewrite $\text{comp_assoc}.$
 apply $(\text{comp_inc_compat_ab_a } H).$
Qed.

Lemma 206 (residual_property6)

$$\alpha \triangleright (\gamma^\# \triangleright \beta^\#)^\# = (\gamma^\# \triangleright (\alpha \triangleright \beta)^\#)^\#.$$

Lemma residual_property6

$\{W\ X\ Y\ Z : eqType\} \{alpha : Rel\ X\ Y\} \{beta : Rel\ Y\ Z\} \{gamma : Rel\ Z\ W\} :$
 $alpha\ (gamma\ \# \ \beta\ \#) \ \# = (gamma\ \# \ (alpha\ \beta) \ \#) \ \#.$

Proof.

apply *inc_lower*.
 move \Rightarrow *delta*.
 split; move \Rightarrow *H*.
 apply *inv_inc_move*.
 apply *inc_residual*.
 apply *inv_inc_move*.
 apply *inc_residual*.
 rewrite *comp_inv comp_assoc*.
 apply *inv_inc_move*.
 apply *inc_residual*.
 apply *inv_inc_invol*.
 rewrite *comp_inv inv_invol*.
 apply *inc_residual*.
 apply *H*.
 apply *inc_residual*.
 apply *inv_inc_move*.
 apply *inc_residual*.
 apply *inv_inc_move*.
 rewrite *comp_inv inv_invol inv_invol comp_assoc*.
 apply *inc_residual*.
 apply *inv_inc_invol*.
 rewrite *comp_inv*.
 apply *inc_residual*.
 apply *inv_inc_move*.
 apply *H*.

Qed.

Lemma 207 (residual_property7a, residual_property7b)

$$\alpha \triangleright (\beta \Rightarrow \beta') \sqsubseteq (\alpha \cdot \beta \Rightarrow \alpha \cdot \beta') \sqsubseteq \alpha \triangleright (\beta \Rightarrow \alpha^\# \cdot \beta').$$

Lemma residual_property7a

$\{X\ Y\ Z : eqType\} \{alpha : Rel\ X\ Y\} \{beta\ beta' : Rel\ Y\ Z\} :$
 $(alpha\ (\beta \gg \beta')) \sqsubseteq ((alpha \cdot \beta) \gg (alpha \cdot \beta')).$

CHAPTER 8. LIBRARY RESIDUAL

Proof.

```

apply inc_rpc.
rewrite cap_comm.
apply (@inc_trans _ _ _ _ (@dedekind1 _ _ _ _ _)).
apply comp_inc_compat_ab_ab'.
rewrite cap_comm.
apply inc_rpc.
apply inv_residual_inc.
Qed.

```

Lemma residual_property7b

$\{X \ Y \ Z : \text{eqType}\} \{alpha : \text{Rel } X \ Y\} \{beta \ beta' : \text{Rel } Y \ Z\}:$
 $((alpha \cdot beta) \gg (alpha \cdot beta')) \quad (alpha \quad (beta \gg (alpha \# \cdot (alpha \cdot beta')))).$

Proof.

```

rewrite inc_residual inc_rpc.
apply (@inc_trans _ _ _ _ (@dedekind1 _ _ _ _ _)).
apply comp_inc_compat_ab_ab'.
rewrite inv_invol -inc_rpc.
apply inc_refl.
Qed.

```

Lemma 208 (residual_property8) *Let α be a univalent relation. Then,*

$$\alpha \triangleright (\beta \Rightarrow \beta') = (\alpha \cdot \beta \Rightarrow \alpha \cdot \beta').$$

Lemma residual_property8

$\{X \ Y \ Z : \text{eqType}\} \{alpha : \text{Rel } X \ Y\} \{beta \ beta' : \text{Rel } Y \ Z\}:$
 $\text{univalent_r } alpha \rightarrow alpha \quad (beta \gg beta') = (alpha \cdot beta) \gg (alpha \cdot beta').$

Proof.

```

move => H.
apply inc_antisym.
apply residual_property7a.
apply (@inc_trans _ _ _ _ residual_property7b).
apply residual_inc_compat_l.
apply rpc_inc_compat_l.
rewrite -comp_assoc.
apply (comp_inc_compat_ab_b H).
Qed.

```

Lemma 209 (residual_property9) *Let α be a univalent relation. Then,*

$$\alpha \triangleright \beta = (\alpha \cdot \nabla_{YZ} \Rightarrow \alpha \cdot \beta).$$

Lemma residual_property9

CHAPTER 8. LIBRARY RESIDUAL

$\{X\ Y\ Z : eqType\} \{alpha : Rel\ X\ Y\} \{beta : Rel\ Y\ Z\} :$
 $univalent_r\ alpha \rightarrow alpha \quad beta = (alpha \cdot \quad Y\ Z) \gg (alpha \cdot beta).$

Proof.

`move` $\Rightarrow H$.

by `[rewrite -(residual_property8 H) rpc_universal_alpha]`.

Qed.

Chapter 9

Library Rationality

```
Require Import Basic_Notations.
Require Import Basic_Lemmas.
Require Import Relation_Properties.
Require Import Functions_Mappings.
```

9.1 有理性から導かれる系

Lemma 210 (rationality_corollary1) *Let $u : A \rightarrow A$ and $u \sqsubseteq id_A$. Then,*

$$\exists R, \exists j : R \rightarrowtail A, u = j^\# \cdot j.$$

Lemma *rationality_corollary1* { $A : eqType$ } { $u : Rel\ A\ A$ }:
 $u \sqsubseteq Id\ A \rightarrow \exists (R : eqType)(j : Rel\ R\ A), injection_r\ j \wedge u = j^\# \cdot j.$

Proof.

```
move : (rationality _ _ u).
elim => R.
elim => f.
elim => g.
elim => H.
elim => H0.
elim => H1 H2 H3.
exists R.
exists f.
assert (g = f).
apply (function_inc H0 H).
apply (@inc_trans _ _ _ ((f · f #) · g)).
apply comp_inc_compat_b_ab.
apply H.
rewrite comp_assoc -H1.
```

CHAPTER 9. LIBRARY RATIONALITY

```

apply (comp_inc_compat_ab_a H3).
rewrite H4 in H1.
rewrite H4 cap_idem in H2.
split.
split.
apply H.
rewrite /univalent_r.
rewrite inv_invol H2.
apply inc_refl.
apply H1.
Qed.

```

Lemma 211 (rationality_corollary2) *Let $f : A \rightarrow B$ be a function. Then,*

$$\exists e : A \rightarrow R, \exists m : R \rightarrow B, f = e \cdot m.$$

Lemma *rationality_corollary2* $\{A\ B : \text{eqType}\} \{f : \text{Rel } A\ B\}$:
 $\text{function_r } f \rightarrow \exists (R : \text{eqType})(e : \text{Rel } A\ R)(m : \text{Rel } R\ B), \text{surjection_r } e \wedge \text{injection_r } m.$

Proof.

```

elim  $\Rightarrow$  H H0.
move : (@rationality_corollary1 - (f # · f) H0).
elim  $\Rightarrow$  R.
elim  $\Rightarrow$  m.
elim  $\Rightarrow$  H1 H2.
 $\exists$  R.
 $\exists$  (f · m #).
 $\exists$  m.
split.
apply (injection_unique2 H1 (conj H H0) H2).
apply H1.
Qed.

```

Chapter 10

Library Conjugate

```
Require Import Basic_Notations.
Require Import Basic_Lemmas.
Require Import Relation_Properties.
Require Import Functions_Mappings.
Require Import Dedekind.
```

10.1 共役性の定義

条件 P を満たす関係 $\alpha : A \rightarrow B$ と条件 Q を満たす関係 $\beta : A' \rightarrow B'$ が変換 $\alpha = \phi(\beta), \beta = \psi(\alpha)$ によって, 1 対 1 (全射的) に対応することを, 図式

$$\frac{\alpha : A \rightarrow B \ \{P\} \quad \alpha = \phi(\beta)}{\beta : A' \rightarrow B' \ \{Q\} \quad \beta = \psi(\alpha)}$$

によって表す. また, Coq では以下のように表すことにする.

Definition *conjugate*

```
(A B C D : eqType) (P : Rel A B → Prop) (Q : Rel C D → Prop)
(phi : Rel C D → Rel A B) (psi : Rel A B → Rel C D) :=
(∀ alpha : Rel A B, P alpha → Q (psi alpha) ∧ phi (psi alpha) = alpha)
∧ (∀ beta : Rel C D, Q beta → P (phi beta) ∧ psi (phi beta) = beta).
```

さらに, 上の図式において条件 P または Q が不要な場合には, 以下の `True_r` を代入する.

Definition *True_r* {A B : eqType} := fun _ : Rel A B ⇒ True.

10.2 共役の例

Lemma 212 (inv_conjugate) *Inverse relation ($\#$) makes conjugate. That is,*

$$\frac{\alpha : A \rightarrow B \quad \alpha = \beta^\#}{\beta : B \rightarrow A \quad \beta = \alpha^\#}.$$

Lemma *inv_conjugate* {A B : eqType}:

conjugate A B B A True_r True_r (@inverse -) (@inverse -).

Proof.

split.

move \Rightarrow *alpha H*.

split.

by [].

apply *inv_invol*.

move \Rightarrow *beta H*.

split.

by [].

apply *inv_invol*.

Qed.

Lemma 213 (injection_conjugate) *Let $j : C \hookrightarrow B$ be an injection. Then,*

$$\frac{f : A \rightarrow B \quad \{f^\# \cdot f \sqsubseteq j^\# \cdot j\}}{h : A \rightarrow C} \quad \frac{f = h \cdot j}{h = f \cdot j^\#}$$

Lemma *injection_conjugate* {A B C : eqType} {j : Rel C B}:

injection_r j \rightarrow

conjugate A B A C (fun f : Rel A B \Rightarrow ((f # \cdot f) (j # \cdot j)) \wedge function_r f)

(fun h : Rel A C \Rightarrow function_r h) (fun h : Rel A C \Rightarrow h \cdot j) (fun f : Rel A B \Rightarrow f \cdot j #).

Proof.

elim.

elim \Rightarrow *H H0 H1*.

split.

move \Rightarrow *alpha*.

elim \Rightarrow *H2*.

elim \Rightarrow *H3 H4*.

assert (function_r (alpha \cdot j #)).

split.

apply (@inc_trans - - - - H3).

rewrite *comp_inv inv_invol comp_assoc* -(@comp_assoc - - - - j).

```
apply (@inc_trans _ _ _ (alpha • ((alpha # • alpha) • alpha #))).
rewrite comp_assoc -comp_assoc.
apply (comp_inc_compat_a_ab H3).
apply comp_inc_compat_ab_ab'.
apply (comp_inc_compat_ab_a'b H2).
apply (fun H' ⇒ @inc_trans _ _ _ _ H' H1).
rewrite comp_inv inv_invol comp_assoc -(@comp_assoc _ _ _ _ alpha).
apply comp_inc_compat_ab_ab'.
apply comp_inc_compat_ab_b.
apply (@inc_trans _ _ _ _ H2).
apply H0.
split.
apply H5.
apply function_inc.
apply function_comp.
apply H5.
split.
apply H.
apply H0.
split.
apply H3.
apply H4.
rewrite comp_assoc.
apply comp_inc_compat_ab_a.
apply H0.
move ⇒ beta.
elim ⇒ H2 H3.
assert (function_r (beta • j)).
split.
apply (@inc_trans _ _ _ _ H2).
rewrite comp_inv comp_assoc -(@comp_assoc _ _ _ _ j).
apply comp_inc_compat_ab_ab'.
apply (comp_inc_compat_b_ab H).
apply (fun H' ⇒ @inc_trans _ _ _ _ H' H0).
rewrite comp_inv comp_assoc -(@comp_assoc _ _ _ _ beta).
apply comp_inc_compat_ab_ab'.
apply (comp_inc_compat_ab_b H3).
split.
split.
rewrite comp_inv comp_assoc -(@comp_assoc _ _ _ _ beta).
apply comp_inc_compat_ab_ab'.
apply (comp_inc_compat_ab_b H3).
```



```

apply H4.
rewrite comp_assoc.
replace (j • j #) with (Id C).
apply comp_id_r.
apply inc_antisym.
apply H.
rewrite /univalent_r in H1.
rewrite inv_invol in H1.
apply H1.
Qed.

```

Lemma 214 (`injection_conjugate_corollary1`, `injection_conjugate_corollary2`)
Let $j : C \rightarrow B$ be an injection and $f : A \rightarrow B$ be a function. Then,

$$f^\# \cdot f \sqsubseteq j^\# \cdot j \Leftrightarrow (\exists! h : A \rightarrow C, f = h \cdot j) \Leftrightarrow (\exists h' : A \rightarrow C, f \sqsubseteq h' \cdot j).$$

Lemma `injection_conjugate_corollary1` { $A B C : eqType$ } { $f : Rel A B$ } { $j : Rel C B$ }:
`injection_r j → function_r f →`
`((f # • f) (j # • j) ↔ ∃! h : Rel A C, function_r h ∧ f = h • j).`

Proof.

```

move ⇒ H H0.
move : (@injection_conjugate A _ _ H).
elim ⇒ H1 H2.
split; move ⇒ H3.
∃ (f • j #).
split.
move : (H1 f (conj H3 H0)).
elim ⇒ H4 H5.
split.
apply H4.
by [rewrite H5].
move ⇒ h.
elim ⇒ H4 H5.
rewrite H5 comp_assoc.
replace (j • j #) with (Id C).
apply comp_id_r.
rewrite /injection_r/function_r/univalent_r in H.
rewrite inv_invol in H.
apply inc_antisym.
apply H.
apply H.
elim H3 ⇒ h.
elim.

```

CHAPTER 10. LIBRARY CONJUGATE

```

elim ⇒ H4 H5 H6.
rewrite H5 comp_inv comp_assoc -(@comp_assoc _ _ _ _ h).
apply comp_inc_compat_ab_ab'.
apply comp_inc_compat_ab_b.
apply H4.
Qed.

Lemma injection_conjugate_corollary2 {A B C : eqType} {f : Rel A B} {j : Rel C B}:
  injection_r j → function_r f →
  ((f # • f) (• j # • j) ↔ ∃ h' : Rel A C, f (• h' • j)).

Proof.
move ⇒ H H0.
split; move ⇒ H1.
apply (injection_conjugate_corollary1 H H0) in H1.
elim H1 ⇒ h.
elim.
elim ⇒ H2 H3 H4.
∃ h.
rewrite H3.
apply inc_refl.
elim H1 ⇒ h' H2.
replace (f # • f) with (f # • (f (• h' • j))).
apply (@inc_trans _ _ ((f # • f) • (j # • j))).
rewrite comp_assoc cap_comm -(@comp_assoc _ _ _ _ f).
apply comp_inc_compat_ab_ab'.
apply (@inc_trans _ _ _ _ (@dedekind2 _ _ _ _ _)).
apply comp_inc_compat_ab_a'b.
apply cap_r.
apply comp_inc_compat_ab_b.
apply H0.
apply f_equal.
apply inc_def1 in H2.
by [rewrite -H2].
Qed.

```

Lemma 215 (surjection_conjugate) *Let $e : A \twoheadrightarrow C$ be a surjection. Then,*

$$\frac{f : A \rightarrow B \quad \{e \cdot e^\# \sqsubseteq f \cdot f^\#\}}{h : C \rightarrow B} \quad \frac{f = e \cdot h}{h = e^\# \cdot f}$$

Lemma *surjection_conjugate* {*A B C* : *eqType*} {*e* : *Rel A C*}:
surjection_r e →
conjugate A B C B (**fun** *f* : *Rel A B* ⇒ ((*e* • *e* #) (• *f* • *f* #)) ∧ *function_r f*)

CHAPTER 10. LIBRARY CONJUGATE

(**fun** $h : \text{Rel } C \ B \Rightarrow \text{function_r } h$) (**fun** $h : \text{Rel } C \ B \Rightarrow e \cdot h$) (**fun** $f : \text{Rel } A \ B \Rightarrow e \# \cdot f$).

Proof.

elim.

elim $\Rightarrow H \ H0 \ H1$.

split.

move $\Rightarrow \text{alpha}$.

elim $\Rightarrow H2$.

elim $\Rightarrow H3 \ H4$.

assert ($\text{function_r } (e \# \cdot \text{alpha})$).

split.

apply ($@inc_trans _ _ _ _ H1$).

rewrite $comp_inv \ inv_invol \ comp_assoc \ -(@comp_assoc _ _ _ _ \text{alpha})$.

apply $comp_inc_compat_ab_ab'$.

apply ($comp_inc_compat_b_ab \ H3$).

apply (**fun** $H' \Rightarrow @inc_trans _ _ _ _ H' \ H4$).

rewrite $comp_inv \ inv_invol \ comp_assoc \ -(@comp_assoc _ _ _ _ e)$.

apply ($@inc_trans _ _ _ (\text{alpha} \# \cdot ((\text{alpha} \cdot \text{alpha} \#) \cdot \text{alpha}))$).

apply $comp_inc_compat_ab_ab'$.

apply ($comp_inc_compat_ab_a'b \ H2$).

rewrite $comp_assoc \ -comp_assoc$.

apply ($comp_inc_compat_ab_a \ H4$).

split.

apply $H5$.

apply Logic.eq_sym .

apply function_inc .

split.

apply $H3$.

apply $H4$.

apply function_comp .

split.

apply H .

apply $H0$.

apply $H5$.

rewrite $-comp_assoc$.

apply $comp_inc_compat_b_ab$.

apply H .

move $\Rightarrow \text{beta}$.

elim $\Rightarrow H2 \ H3$.

assert ($\text{function_r } (e \cdot \text{beta})$).

split.

apply ($@inc_trans _ _ _ _ H$).

CHAPTER 10. LIBRARY CONJUGATE

```

rewrite comp_inv comp_assoc -(@comp_assoc _ _ _ beta).
apply comp_inc_compat_ab_ab'.
apply (comp_inc_compat_b_ab H2).
apply (fun H' => @inc_trans _ _ _ _ H' H3).
rewrite comp_inv comp_assoc -(@comp_assoc _ _ _ _ e).
apply comp_inc_compat_ab_ab'.
apply (comp_inc_compat_ab_b H0).
split.
split.
rewrite comp_inv comp_assoc -(@comp_assoc _ _ _ beta).
apply comp_inc_compat_ab_ab'.
apply (comp_inc_compat_b_ab H2).
apply H4.
rewrite -comp_assoc.
replace (e # • e) with (Id C).
apply comp_id_l.
apply inc_antisym.
rewrite /total_r in H1.
rewrite inv_invol in H1.
apply H1.
apply H0.
Qed.

```

Lemma 216 (surjection_conjugate_corollary) *Let $e : A \twoheadrightarrow C$ be a surjection and $f : A \rightarrow B$ be a function. Then,*

$$e \cdot e^\# \sqsubseteq f \cdot f^\# \Leftrightarrow (\exists! h : C \rightarrow B, f = e \cdot h).$$

Lemma *surjection_conjugate_corollary* {A B C : eqType} {f : Rel A B} {e : Rel A C}:
surjection_r e → function_r f →
((e • e #) (f • f #) ↔ ∃! h : Rel C B, function_r h ∧ f = e • h).

Proof.

```

move => H H0.
move : (@surjection_conjugate _ B _ _ H).
elim => H1 H2.
split; move => H3.
∃ (e # • f).
split.
move : (H1 f (conj H3 H0)).
elim => H4 H5.
split.
apply H4.
by [rewrite H5].

```

CHAPTER 10. LIBRARY CONJUGATE

```

move  $\Rightarrow$   $h$ .
elim  $\Rightarrow$   $H4\ H5$ .
rewrite  $H5$  -comp_assoc.
replace ( $e \# \cdot e$ ) with ( $Id\ C$ ).
apply comp_id_l.
rewrite /surjection_r/function_r/total_r in  $H$ .
rewrite inv_invol in  $H$ .
apply inc_antisym.
apply  $H$ .
apply  $H$ .
elim  $H3 \Rightarrow h$ .
elim.
elim  $\Rightarrow$   $H4\ H5\ H6$ .
rewrite  $H5$  comp_inv comp_assoc -(@comp_assoc _ _ _  $h$ ).
apply comp_inc_compat_ab_ab'.
apply comp_inc_compat_b_ab.
apply  $H4$ .
Qed.

```

Lemma 217 (subid_conjugate) *Subidentity $u \sqsubseteq id_A$ corresponds $\rho : I \rightarrow A$. That is,*

$$\frac{\rho : I \rightarrow A}{u : A \rightarrow A \ \{u \sqsubseteq id_A\}} \quad \frac{\rho = \nabla_{IA} \cdot u}{u = id_A \sqcap \nabla_{AI} \cdot \rho}.$$

Lemma subid_conjugate $\{A : eqType\}$:
 $conjugate\ i\ A\ A\ A\ True_r\ (\text{fun } u : Rel\ A\ A \Rightarrow u \quad Id\ A)$
 $(\text{fun } u : Rel\ A\ A \Rightarrow \quad i\ A \cdot u) (\text{fun } rho : Rel\ i\ A \Rightarrow Id\ A \quad (\quad A\ i \cdot rho)).$

Proof.
split.
move \Rightarrow $alpha\ H$.
split.
apply cap_l.
apply inc_antisym.
apply (@inc_trans _ _ _ ($\quad i\ A \cdot (\quad A\ i \cdot alpha)$)).
apply comp_inc_compat_ab_ab'.
apply cap_r.
rewrite -comp_assoc.
apply comp_inc_compat_ab_b.
rewrite unit_identity_is_universal.
apply inc_alpha_universal.
rewrite -(@inv_universal $i\ A$).
apply (fun $H' \Rightarrow @inc_trans\ _\ _\ _\ H' (@dedekind1\ _\ _\ _\ _)$).
rewrite comp_id_r cap_comm cap_universal.

```

apply inc_refl.
move ⇒ beta H.
split.
by [].
apply inc_antisym.
rewrite cap_comm -comp_assoc lemma_for_tarski2.
apply (@inc_trans _ _ _ _ (@dedekind2 _ _ _ _ _)).
rewrite comp_id_l cap_comm cap_universal.
apply comp_inc_compat_ab_b.
rewrite -inv_inc_move inv_id.
apply H.
apply inc_cap.
split.
apply H.
rewrite -comp_assoc.
apply comp_inc_compat_b_ab.
rewrite lemma_for_tarski2.
apply inc_alpha_universal.
Qed.

```

Lemma 218 (subid_conjugate_corollary1) *Let $u, v : A \rightarrow A$ and $u, v \sqsubseteq id_A$. Then,*

$$\nabla_{IA} \cdot u = \nabla_{IA} \cdot v \Rightarrow u = v.$$

Lemma subid_conjugate_corollary1 $\{A : eqType\} \{u \ v : Rel \ A \ A\}$:
 $u \quad Id \ A \rightarrow v \quad Id \ A \rightarrow \quad i \ A \cdot u = \quad i \ A \cdot v \rightarrow u = v.$

Proof.

```

move ⇒ H H0 H1.
move : (@subid_conjugate A).
elim ⇒ H2 H3.
move : (H3 u H).
elim ⇒ H4 H5.
rewrite -H5.
move : (H3 v H0).
elim ⇒ H6 H7.
rewrite -H7.
apply f_equal.
apply f_equal.
apply H1.
Qed.

```

Lemma 219 (subid_conjugate_corollary2) *Let $\rho, \rho' : I \rightarrow A$. Then,*

$$id_A \sqcap \nabla_{AI} \cdot \rho = id_A \sqcap \nabla_{AI} \cdot \rho' \Rightarrow \rho = \rho'.$$

Lemma *subid_conjugate_corollary2* $\{A : eqType\} \{rho\ rho' : Rel\ i\ A\}$:
 $Id\ A \quad (\quad A\ i \cdot rho) = Id\ A \quad (\quad A\ i \cdot rho') \rightarrow rho = rho'.$

Proof.

move $\Rightarrow H$.

move : (*@subid_conjugate A*).

elim $\Rightarrow H0\ H1$.

move : (*H0 rho I*).

elim $\Rightarrow H2\ H3$.

rewrite -*H3*.

move : (*H0 rho' I*).

elim $\Rightarrow H4\ H5$.

rewrite -*H5*.

apply f_equal.

apply *H*.

Qed.

Chapter 11

Library Domain

```
Require Import Basic_Notations.
Require Import Basic_Lemmas.
Require Import Relation_Properties.
Require Import Functions_Mappings.
Require Import Dedekind.
Require Import Logic.FunctionalExtensionality.
```

11.1 定義域の定義

関係 $\alpha : A \rightarrow B$ に対して, その定義域 (関係) $[\alpha] : A \rightarrow A$ は,

$$[\alpha] = \alpha \cdot \alpha^\# \sqcap id_A$$

で表される. また, Coq では以下のように表すことにする.

Definition *domain* $\{A\ B : eqType\}$ (*alpha* : *Rel A B*) := (*alpha* · *alpha* #) Id *A*.

11.2 定義域の性質

11.2.1 基本的な性質

Lemma 220 (*domain_another_def*) *Let* $\alpha : A \rightarrow B$. *Then*,

$$[\alpha] = \alpha \cdot \nabla_{BA} \sqcap id_A.$$

Lemma *domain_another_def* $\{A\ B : eqType\}$ $\{\alpha : Rel\ A\ B\}$:
domain alpha = (*alpha* · B *A*) Id *A*.

Proof.

CHAPTER 11. LIBRARY DOMAIN

```

apply inc_antisym.
apply cap_inc_compat_r.
apply comp_inc_compat_ab_ab'.
apply inc_alpha_universal.
apply inc_cap.
split.
apply (@inc_trans _ _ _ _ (@dedekind1 _ _ _ _ _)).
apply comp_inc_compat_ab_ab'.
rewrite cap_comm comp_id_r cap_universal.
apply inc_refl.
apply cap_r.
Qed.

```

Lemma 221 (domain_inv) *Let $\alpha : A \rightarrow B$. Then,*

$$\lfloor \alpha \rfloor^\# = \lfloor \alpha \rfloor.$$

Lemma domain_inv $\{A B : eqType\} \{alpha : Rel A B\}$:
 $(domain\ alpha) \# = domain\ alpha$.

Proof.

```

apply dedekind_id1.
apply cap_r.
Qed.

```

Lemma 222 (domain_comp_alpha1, domain_comp_alpha2) *Let $\alpha : A \rightarrow B$. Then,*

$$\lfloor \alpha \rfloor \cdot \alpha = \alpha \wedge \alpha^\# \cdot \lfloor \alpha \rfloor = \alpha^\#.$$

Lemma domain_comp_alpha1 $\{A B : eqType\} \{alpha : Rel A B\}$:
 $(domain\ alpha) \cdot alpha = alpha$.

Proof.

```

apply inc_antisym.
apply comp_inc_compat_ab_b.
apply cap_r.
rewrite /domain.
rewrite cap_comm.
apply (fun H' => @inc_trans _ _ _ _ H' (@dedekind2 _ _ _ _ _)).
rewrite comp_id_l cap_idem.
apply inc_refl.
Qed.

```

Lemma domain_comp_alpha2 $\{A B : eqType\} \{alpha : Rel A B\}$:
 $alpha \# \cdot (domain\ alpha) = alpha \#$.

CHAPTER 11. LIBRARY DOMAIN

Proof.

```
rewrite -domain_inv -comp_inv.
apply f_equal.
apply domain_comp_alpha1.
Qed.
```

Lemma 223 (domain_inc_compat) *Let $\alpha, \alpha' : A \rightarrow B$. Then,*

$$\alpha \sqsubseteq \alpha' \Rightarrow \lfloor \alpha \rfloor \sqsubseteq \lfloor \alpha' \rfloor.$$

Lemma *domain_inc_compat* {A B : eqType} {alpha alpha' : Rel A B}:
 $\text{alpha} \quad \text{alpha}' \rightarrow \text{domain alpha} \quad \text{domain alpha}'.$

Proof.

```
move => H.
apply cap_inc_compat_r.
apply comp_inc_compat.
apply H.
apply (@inc_inv _ _ _ H).
Qed.
```

Lemma 224 (domain_total) *Let $\alpha : A \rightarrow B$. Then,*

$$“\alpha \text{ is total}” \Leftrightarrow \lfloor \alpha \rfloor = id_A.$$

Lemma *domain_total* {A B : eqType} {alpha : Rel A B}:
 $\text{total_r alpha} \leftrightarrow \text{domain alpha} = Id A.$

Proof.

```
split; move => H.
rewrite /domain.
rewrite cap_comm.
apply Logic.eq_sym.
apply inc_def1.
apply H.
apply inc_def1.
rewrite /domain in H.
by [rewrite cap_comm H].
Qed.
```

Lemma 225 (domain_inc_id) *Let $u : A \rightarrow A$. Then,*

$$u \sqsubseteq id_A \Leftrightarrow \lfloor u \rfloor = u.$$

Lemma *domain_inc_id* {A : eqType} {u : Rel A A}: $u \quad Id A \leftrightarrow \text{domain } u = u.$

Proof.

```

split; move ⇒ H.
rewrite /domain.
rewrite (dedekind_id1 H) (dedekind_id2 H).
apply inc_def1 in H.
by [rewrite -H].
rewrite -H.
apply cap_r.
Qed.

```

11.2.2 合成と定義域

Lemma 226 (comp_domain1, comp_domain2) *Let $\alpha : A \rightarrow B$ and $\beta : B \rightarrow C$. Then,*

$$[\alpha \cdot \beta] = [\alpha \cdot [\beta]] \sqsubseteq [\alpha].$$

Lemma comp_domain1 $\{A\ B\ C : eqType\} \{alpha : Rel\ A\ B\} \{beta : Rel\ B\ C\}$:
 $domain\ (alpha \cdot beta) = domain\ alpha.$

Proof.

```

rewrite /domain.
rewrite comp_inv.
apply (@inc_trans _ _ _ ((alpha · ((beta · (beta # · alpha #)) alpha #)) Id A)).
replace (((alpha · beta) · (beta # · alpha #)) Id A) with (((alpha · beta) ·
(beta # · alpha #)) Id A) Id A.
apply cap_inc_compat_r.
rewrite comp_assoc.
apply (@inc_trans _ _ _ _ (@dedekind1 _ _ _ _ _)).
rewrite comp_id_r.
apply inc_refl.
by [rewrite cap_assoc cap_idem].
apply cap_inc_compat_r.
apply comp_inc_compat_ab_ab'.
apply cap_r.
Qed.

```

Lemma comp_domain2 $\{A\ B\ C : eqType\} \{alpha : Rel\ A\ B\} \{beta : Rel\ B\ C\}$:
 $domain\ (alpha \cdot beta) = domain\ (alpha \cdot domain\ beta).$

Proof.

```

apply inc_antisym.
replace (domain (alpha · beta)) with (domain ((alpha · domain beta) · beta)).
apply comp_domain1.
by [rewrite comp_assoc domain_comp_alpha1].
apply (@inc_trans _ _ _ (domain (alpha · (beta · beta #)))).

```

CHAPTER 11. LIBRARY DOMAIN

```

apply domain_inc_compat.
apply comp_inc_compat_ab_ab'.
apply cap_l.
rewrite -comp_assoc.
apply comp_domain1.
Qed.

```

Lemma 227 (comp_domain3) *Let $\alpha : A \rightarrow B$ be a relation and $\beta : B \rightarrow C$ be a total relation. Then,*

$$\lfloor \alpha \cdot \beta \rfloor = \lfloor \alpha \rfloor.$$

Lemma comp_domain3 $\{A\ B\ C : eqType\} \{alpha : Rel\ A\ B\} \{beta : Rel\ B\ C\}$:
 $total_r\ beta \rightarrow domain\ (alpha \cdot beta) = domain\ alpha.$

Proof.

```

move => H.
apply inc_antisym.
apply comp_domain1.
rewrite /domain.
rewrite comp_inv_comp_assoc -(@comp_assoc _ _ _ beta).
apply cap_inc_compat_r.
apply comp_inc_compat_ab_ab'.
apply (comp_inc_compat_b_ab H).
Qed.

```

Lemma 228 (comp_domain4) *Let $\alpha : A \rightarrow B$ and $\beta : B \rightarrow C$. Then,*

$$\lfloor \alpha^\# \rfloor \sqsubseteq \lfloor \beta \rfloor \Rightarrow \lfloor \alpha \cdot \beta \rfloor = \lfloor \alpha \rfloor.$$

Lemma comp_domain4 $\{A\ B\ C : eqType\} \{alpha : Rel\ A\ B\} \{beta : Rel\ B\ C\}$:
 $domain\ (alpha \#) \quad domain\ beta \rightarrow domain\ (alpha \cdot beta) = domain\ alpha.$

Proof.

```

move => H.
apply inc_antisym.
apply comp_domain1.
rewrite /domain.
rewrite -(@domain_comp_alpha1 _ _ (alpha #)) comp_inv_comp_assoc -(@comp_assoc _ _
_ _ beta).
apply cap_inc_compat_r.
apply comp_inc_compat_ab_ab'.
apply comp_inc_compat_ab_a'b.
apply (@inc_trans _ _ _ _ H).
apply cap_l.
Qed.

```

CHAPTER 11. LIBRARY DOMAIN

Lemma 229 (comp_domain5) *Let $\alpha : A \rightarrow B$ be a univalent relation and $\beta : B \rightarrow C$. Then,*

$$\lfloor \alpha^\# \rfloor \sqsubseteq \lfloor \beta \rfloor \Leftrightarrow \lfloor \alpha \cdot \beta \rfloor = \lfloor \alpha \rfloor.$$

Lemma comp_domain5 $\{A\ B\ C : \text{eqType}\} \{ \text{alpha} : \text{Rel } A\ B \} \{ \text{beta} : \text{Rel } B\ C \}$:
 $\text{univalent_r } \text{alpha} \rightarrow$
 $(\text{domain } (\text{alpha } \#) \quad \text{domain } \text{beta} \leftrightarrow \text{domain } (\text{alpha} \cdot \text{beta}) = \text{domain } \text{alpha}).$

Proof.

```
move  $\Rightarrow$  H.
split; move  $\Rightarrow$  H0.
apply (comp_domain4 H0).
rewrite /domain.
rewrite inv_invol.
apply cap_inc_compat_r.
replace (alpha #  $\cdot$  alpha) with (alpha #  $\cdot$  (domain (alpha  $\cdot$  beta)  $\cdot$  alpha)).
rewrite /domain.
rewrite comp_inv.
apply (@inc_trans _ _ _ (alpha #  $\cdot$  (((alpha  $\cdot$  beta)  $\cdot$  (beta #  $\cdot$  alpha #))  $\cdot$  alpha))).
apply comp_inc_compat_ab_ab'.
apply comp_inc_compat_ab_a'b.
apply cap_l.
rewrite comp_assoc comp_assoc comp_assoc -comp_assoc -(@comp_assoc _ _ _ beta).
apply (@inc_trans _ _ _ _ (comp_inc_compat_ab_b H)).
apply (comp_inc_compat_ab_a H).
by [rewrite H0 domain_comp_alpha1].
Qed.
```

Lemma 230 (comp_domain6) *Let $\alpha : A \rightarrow B$ and $\beta : B \rightarrow C$. Then,*

$$\alpha \cdot \lfloor \beta \rfloor \sqsubseteq \lfloor \alpha \cdot \beta \rfloor \cdot \alpha.$$

Lemma comp_domain6 $\{A\ B\ C : \text{eqType}\} \{ \text{alpha} : \text{Rel } A\ B \} \{ \text{beta} : \text{Rel } B\ C \}$:
 $(\text{alpha} \cdot \text{domain } \text{beta}) \quad (\text{domain } (\text{alpha} \cdot \text{beta}) \cdot \text{alpha}).$

Proof.

```
apply (@inc_trans _ _ _ _ (@comp_cap_distr_l _ _ _ _ _)).
rewrite cap_comm.
replace (alpha  $\cdot$  Id B) with (Id A  $\cdot$  alpha).
apply (@inc_trans _ _ _ _ (@dedekind2 _ _ _ _ _)).
rewrite cap_comm -comp_assoc comp_assoc -comp_inv.
apply inc_refl.
by [rewrite comp_id_l comp_id_r].
Qed.
```

CHAPTER 11. LIBRARY DOMAIN

Lemma 231 (comp_domain7) *Let $\alpha : A \rightarrow B$ be a univalent relation and $\beta : B \rightarrow C$. Then,*

$$\alpha \cdot \lfloor \beta \rfloor = \lfloor \alpha \cdot \beta \rfloor \cdot \alpha.$$

Lemma comp_domain7 $\{A\ B\ C : \text{eqType}\} \{alpha : \text{Rel } A\ B\} \{beta : \text{Rel } B\ C\}$:
 $\text{univalent_r } alpha \rightarrow alpha \cdot \text{domain } beta = \text{domain } (alpha \cdot beta) \cdot alpha.$

Proof.

```
move  $\Rightarrow$  H.
apply inc_antisym.
apply comp_domain6.
apply (@inc_trans _ _ _ _ (@comp_cap_distr_r _ _ _ _ _)).
rewrite comp_id_l comp_inv comp_assoc comp_assoc.
apply (@inc_trans _ _ _ _ (@dedekind1 _ _ _ _ _)).
apply comp_inc_compat_ab_ab'.
apply (fun H'  $\Rightarrow$  cap_inc_compat H' H).
rewrite comp_assoc -comp_assoc.
apply (comp_inc_compat_ab_a H).
```

Qed.

Lemma 232 (comp_domain8) *Let $u : A \rightarrow A$, $\alpha : A \rightarrow B$ and $u \sqsubseteq id_A$. Then,*

$$\lfloor u \cdot \alpha \rfloor = u \cdot \lfloor \alpha \rfloor.$$

Lemma comp_domain8 $\{A\ B : \text{eqType}\} \{u : \text{Rel } A\ A\} \{alpha : \text{Rel } A\ B\}$:
 $u \text{ Id } A \rightarrow \text{domain } (u \cdot alpha) = u \cdot \text{domain } alpha.$

Proof.

```
move  $\Rightarrow$  H.
apply inc_antisym.
rewrite -(@cap_idem _ _ (domain (u · alpha))).
rewrite (dedekind_id3 H).
apply cap_inc_compat.
apply (@inc_trans _ _ _ _ (@comp_domain1 _ _ _ _ _)).
apply domain_inc_id in H.
rewrite H.
apply inc_refl.
apply domain_inc_compat.
apply (comp_inc_compat_ab_b H).
apply cap_r.
apply (@inc_trans _ _ _ _ (@comp_domain6 _ _ _ _ _)).
apply (comp_inc_compat_ab_a H).
```

Qed.

11.2.3 その他の性質

Lemma 233 (cap_domain) *Let $\alpha, \alpha' : A \rightarrow B$. Then,*

$$\lfloor \alpha \sqcap \alpha' \rfloor = \alpha \cdot \alpha'^{\#} \sqcap \text{id}_A.$$

Lemma *cap_domain* { $A B : \text{eqType}$ } { $\alpha \alpha' : \text{Rel } A B$ }:
 $\text{domain } (\alpha \sqcap \alpha') = (\alpha \cdot \alpha' \#) \sqcap \text{Id } A.$

Proof.

```

apply inc_antisym.
apply cap_inc_compat_r.
apply comp_inc_compat.
apply cap_l.
apply inc_inv.
apply cap_r.
rewrite (@cap_idem _ _ (Id A)) -cap_assoc.
apply cap_inc_compat_r.
apply (@inc_trans _ _ _ _ (@dedekind _ _ _ _ _)).
rewrite inv_invol comp_id_l comp_id_r -inv_cap_distr (@cap_comm _ _ alpha').
apply inc_refl.

```

Qed.

Lemma 234 (cupL_domain_distr, cupL_domain_distr) *Let $\alpha_\lambda : A \rightarrow B$. Then,*

$$\lfloor \sqcup_{\lambda \in \Lambda} \alpha_\lambda \rfloor = \sqcup_{\lambda \in \Lambda} \lfloor \alpha_\lambda \rfloor.$$

Lemma *cupL_domain_distr* { $A B L : \text{eqType}$ } { $\alpha_L : L \rightarrow \text{Rel } A B$ }:
 $\text{domain } (\sqcup \alpha_L) = \sqcup (\text{fun } l : L \Rightarrow \text{domain } (\alpha_L l)).$

Proof.

```

rewrite /domain.
rewrite inv_cupL_distr comp_cupL_distr_l cap_cupL_distr_r.
apply f_equal.
apply functional_extensionality.
move => l.
rewrite -cap_domain -cap_domain.
apply f_equal.
rewrite cap_idem.
apply inc_antisym.
apply cap_r.
apply inc_cap.
split.
apply inc_cupL.
apply inc_refl.

```

CHAPTER 11. LIBRARY DOMAIN

apply *inc_refl*.

Qed.

Lemma *cup_domain_distr* {*A B* : *eqType*} {*alpha alpha'* : *Rel A B*}:
 $\text{domain } (\text{alpha} \quad \text{alpha}') = \text{domain } \text{alpha} \quad \text{domain } \text{alpha}'.$

Proof.

rewrite *cup_to_cupL cup_to_cupL*.

rewrite *cupL_domain_distr*.

apply *f_equal*.

apply *functional_extensionality*.

induction *x*.

by [].

by [].

Qed.

Lemma 235 (domain_universal1) *Let $\alpha : A \rightarrow B$. Then,*

$$\lfloor \alpha \rfloor \cdot \nabla_{AC} = \alpha \cdot \nabla_{BC}.$$

Lemma *domain_universal1* {*A B C* : *eqType*} {*alpha* : *Rel A B*}:
 $\text{domain } \text{alpha} \cdot \quad A \ C = \text{alpha} \cdot \quad B \ C.$

Proof.

apply *inc_antisym*.

apply (@*inc_trans* _ _ _ ((*alpha* · *alpha* #) · *A C*)).

apply *comp_inc_compat_ab_a'b*.

apply *cap_l*.

rewrite *comp_assoc*.

apply *comp_inc_compat_ab_ab'*.

apply *inc_alpha_universal*.

apply (@*inc_trans* _ _ _ ((*domain alpha* · *alpha*) · *B C*)).

rewrite *domain_comp_alpha1*.

apply *inc_refl*.

rewrite *comp_assoc*.

apply *comp_inc_compat_ab_ab'*.

apply *inc_alpha_universal*.

Qed.

Lemma 236 (domain_universal2) *Let $\alpha : A \rightarrow B$ and $\beta : B \rightarrow C$. Then,*

$$\alpha \cdot \lfloor \beta \rfloor = \alpha \sqcap \nabla_{AC} \cdot \beta^\#.$$

Lemma *domain_universal2* {*A B C* : *eqType*} {*alpha* : *Rel A B*} {*beta* : *Rel B C*}:
 $\text{alpha} \cdot \text{domain } \text{beta} = \text{alpha} \quad (\quad A \ C \cdot \text{beta} \#).$

Proof.

CHAPTER 11. LIBRARY DOMAIN

```

apply inc_antisym.
apply inc_cap.
split.
apply comp_inc_compat_ab_a.
apply cap_r.
apply (@inc_trans _ _ _ _ (@comp_cap_distr_l _ _ _ _)).
apply (@inc_trans _ _ _ _ (@cap_l _ _ _ _)).
rewrite -comp_assoc.
apply comp_inc_compat_ab_a'b.
apply inc_alpha_universal.
rewrite -inv_universal -comp_inv -domain_universal1.
rewrite comp_inv inv_universal domain_inv cap_comm.
apply (@inc_trans _ _ _ _ (@dedekind2 _ _ _ _)).
apply comp_inc_compat_ab_a'b.
rewrite cap_comm cap_universal domain_inv.
apply comp_inc_compat_ab_a.
apply cap_r.
Qed.

```

Lemma 237 (domain_lemma1) *Let $\alpha, \beta : A \rightarrow B$ and β is univalent. Then,*

$$\alpha \sqsubseteq \beta \wedge \lfloor \alpha \rfloor = \lfloor \beta \rfloor \Rightarrow \alpha = \beta.$$

Lemma domain_lemma1 $\{A\ B : \text{eqType}\} \{\text{alpha}\ \text{beta} : \text{Rel}\ A\ B\}$:
 $\text{univalent_r}\ \text{beta} \rightarrow \text{alpha} \rightarrow \text{beta} \rightarrow \text{domain}\ \text{alpha} = \text{domain}\ \text{beta} \rightarrow \text{alpha} = \text{beta}.$

Proof.

```

move => H H0 H1.
apply inc_antisym.
apply H0.
rewrite -(@domain_comp_alpha1 _ _ beta) -H1.
apply (@inc_trans _ _ _ _ (@comp_cap_distr_r _ _ _ _)).
apply (@inc_trans _ _ _ _ (@cap_l _ _ _ _)).
rewrite comp_assoc.
apply comp_inc_compat_ab_a.
apply (fun H' => @inc_trans _ _ _ _ H' H).
apply comp_inc_compat_ab_a'b.
apply (@inc_inv _ _ _ _ H0).
Qed.

```

Lemma 238 (domain_lemma2a, domain_lemma2b) *Let $\alpha : A \rightarrow B$ and $\beta : A \rightarrow C$. Then,*

$$\lfloor \alpha \rfloor \sqsubseteq \lfloor \beta \rfloor \Leftrightarrow \alpha \cdot \nabla_{BB} \sqsubseteq \beta \cdot \nabla_{CB} \Leftrightarrow \alpha \sqsubseteq \beta \cdot \beta^\# \cdot \alpha.$$

CHAPTER 11. LIBRARY DOMAIN

Lemma *domain_lemma2a* $\{A\ B\ C : eqType\} \{alpha : Rel\ A\ B\} \{beta : Rel\ A\ C\}$:
 $domain\ alpha \quad domain\ beta \leftrightarrow (alpha \cdot B\ B) \quad (beta \cdot C\ B).$

Proof.

```
split; move  $\Rightarrow$  H.
rewrite -(@domain_comp_alpha1 _ _ alpha) comp_assoc.
apply (@inc_trans _ _ _ _ (comp_inc_compat_ab_a'b H)).
apply (@inc_trans _ _ _ _ (comp_inc_compat_ab_a'b (@cap_l _ _ _))).
rewrite comp_assoc.
apply comp_inc_compat_ab_ab'.
apply inc_alpha_universal.
apply (@inc_trans _ _ _ (domain ((beta · beta #) · alpha))).
apply domain_inc_compat.
apply (@inc_trans _ _ _ (alpha (beta · C B))).
apply (fun H'  $\Rightarrow$  @inc_trans _ _ _ _ H' (cap_inc_compat_l H)).
replace (alpha (alpha · B B)) with ((alpha · Id B) (alpha · B B)).
apply (fun H'  $\Rightarrow$  @inc_trans _ _ _ _ H' (@comp_cap_distr_l _ _ _ _ _)).
rewrite cap_universal comp_id_r.
apply inc_refl.
by [rewrite comp_id_r].
rewrite cap_comm comp_assoc.
apply (@inc_trans _ _ _ _ (@dedekind1 _ _ _ _ _)).
rewrite cap_comm cap_universal.
apply inc_refl.
rewrite comp_assoc.
apply comp_domain1.
```

Qed.

Lemma *domain_lemma2b* $\{A\ B\ C : eqType\} \{alpha : Rel\ A\ B\} \{beta : Rel\ A\ C\}$:
 $domain\ alpha \quad domain\ beta \leftrightarrow alpha \quad ((beta \cdot beta \#) \cdot alpha).$

Proof.

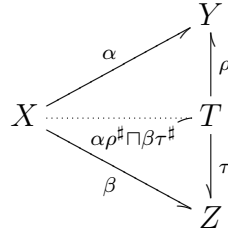
```
split; move  $\Rightarrow$  H.
apply domain_lemma2a in H.
apply (@inc_trans _ _ _ (alpha (beta · C B))).
apply (fun H'  $\Rightarrow$  @inc_trans _ _ _ _ H' (cap_inc_compat_l H)).
replace (alpha (alpha · B B)) with ((alpha · Id B) (alpha · B B)).
apply (fun H'  $\Rightarrow$  @inc_trans _ _ _ _ H' (@comp_cap_distr_l _ _ _ _ _)).
rewrite cap_universal comp_id_r.
apply inc_refl.
by [rewrite comp_id_r].
rewrite cap_comm comp_assoc.
apply (@inc_trans _ _ _ _ (@dedekind1 _ _ _ _ _)).
rewrite cap_comm cap_universal.
apply inc_refl.
```

CHAPTER 11. LIBRARY DOMAIN

apply *domain_inc_compat* in *H*.
 apply (@*inc_trans* _ _ _ _ *H*).
 rewrite *comp_assoc*.
 apply *comp_domain1*.
 Qed.

Lemma 239 (domain_corollary1) *In below figure,*

“ α and β are total” $\wedge \alpha^\# \cdot \beta \sqsubseteq \rho^\# \cdot \tau \Rightarrow$ “ $\alpha \cdot \rho^\# \sqcap \beta \cdot \tau^\#$ is total”.



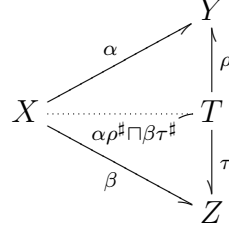
Lemma *domain_corollary1* {*X Y Z T* : *eqType*}
 {*alpha* : *Rel X Y*} {*beta* : *Rel X Z*} {*rho* : *Rel T Y*} {*tau* : *Rel T Z*}:
total_r alpha \rightarrow *total_r beta* \rightarrow (*alpha* # \cdot *beta*) (*rho* # \cdot *tau*) \rightarrow
total_r ((*alpha* \cdot *rho* #) (*beta* \cdot *tau* #)).

Proof.

move \Rightarrow *H H0 H1*.
 move : (*comp_inc_compat H H0*) \Rightarrow *H2*.
 rewrite *comp_id_l* -*comp_assoc* (@*comp_assoc* _ _ _ *alpha*) in *H2*.
 rewrite /*total_r*.
 replace (*Id X*) with (((*alpha* \cdot (*rho* # \cdot *tau*)) \cdot *beta* #) *Id X*).
 rewrite -*comp_assoc comp_assoc*.
 apply (@*inc_trans* _ _ _ _ (@*dedekind* _ _ _ _ _)).
 rewrite *comp_id_l comp_id_r comp_inv comp_inv inv_invol inv_invol*.
 rewrite *inv_cap_distr comp_inv comp_inv inv_invol inv_invol* (@*cap_comm* _ _ (*tau* \cdot *beta* #)).
 apply *inc_refl*.
 apply *Logic.eq_sym*.
 rewrite *cap_comm*.
 apply *inc_def1*.
 apply (@*inc_trans* _ _ _ _ *H2*).
 apply *comp_inc_compat_ab_a'b*.
 apply (*comp_inc_compat_ab_ab' H1*).
 Qed.

Lemma 240 (domain_corollary2) *In below figure,*

“ α and β are univalent” $\wedge \rho \cdot \rho^\sharp \sqcap \tau \cdot \tau^\sharp = \text{id}_T \Rightarrow$ “ $\alpha \cdot \rho^\sharp \sqcap \beta \cdot \tau^\sharp$ is univalent”.



Lemma domain_corollary2 $\{X\ Y\ Z\ T : \text{eqType}\}$
 $\{\text{alpha} : \text{Rel } X\ Y\} \{\text{beta} : \text{Rel } X\ Z\} \{\text{rho} : \text{Rel } T\ Y\} \{\text{tau} : \text{Rel } T\ Z\}$:
 $\text{univalent_r } \text{alpha} \rightarrow \text{univalent_r } \text{beta} \rightarrow (\text{rho} \cdot \text{rho}^\sharp) \quad (\text{tau} \cdot \text{tau}^\sharp) = \text{Id } T \rightarrow$
 $\text{univalent_r } ((\text{alpha} \cdot \text{rho}^\sharp) \quad (\text{beta} \cdot \text{tau}^\sharp)).$

Proof.

`move \Rightarrow H H0 H1.`

`rewrite /univalent_r.`

`rewrite -H1 inv_cap_distr.`

`apply (@inc_trans _ _ _ _ (@comp_cap_distr_l _ _ _ _)).`

`apply cap_inc_compat.`

`apply (@inc_trans _ _ _ _ (@comp_cap_distr_r _ _ _ _)).`

`apply (@inc_trans _ _ _ _ (@cap_l _ _ _ _)).`

`rewrite comp_inv inv_invol -comp_assoc (@comp_assoc _ _ _ _ rho).`

`apply comp_inc_compat_ab_a'b.`

`apply (comp_inc_compat_ab_a H).`

`apply (@inc_trans _ _ _ _ (@comp_cap_distr_r _ _ _ _)).`

`apply (@inc_trans _ _ _ _ (@cap_r _ _ _ _)).`

`rewrite comp_inv inv_invol -comp_assoc (@comp_assoc _ _ _ _ tau).`

`apply comp_inc_compat_ab_a'b.`

`apply (comp_inc_compat_ab_a H0).`

Qed.

11.2.4 矩形関係

$\alpha : A \rightarrow B$ が

$$\alpha \cdot \nabla_{BA} \cdot \alpha \sqsubseteq \alpha$$

を満たすとき, α は 矩形関係 (rectangular relation) であると言われる.

Definition rectangular $\{A\ B : \text{eqType}\} (\text{alpha} : \text{Rel } A\ B) :=$
 $((\text{alpha} \cdot \quad B\ A) \cdot \text{alpha}) \quad \text{alpha}.$

Lemma 241 (rectangular_inv) *Let $\alpha : A \rightarrow B$ is a rectangular relation, then $\alpha^\#$ is also a rectangular relation.*

Lemma *rectangular_inv* $\{A\ B\ : eqType\} \{alpha\ : Rel\ A\ B\}$:
 $rectangular\ alpha \rightarrow rectangular\ (alpha\ \#)$.

Proof.

move $\Rightarrow H$.

apply *inv_inc_move*.

rewrite *comp_inv comp_inv inv_invol inv_universal -comp_assoc*.

apply *H*.

Qed.

Lemma 242 (rectangular_capL, rectangular_cap) *Let $\alpha_\lambda : A \rightarrow B$ are rectangular relations, then $\prod_{\lambda \in \Lambda} \alpha_\lambda$ is also a rectangular relation.*

Lemma *rectangular_capL* $\{A\ B\ L\ : eqType\} \{alpha_L\ : L \rightarrow Rel\ A\ B\}$:
 $(\forall\ l\ : L, rectangular\ (alpha_L\ l)) \rightarrow rectangular\ (_ _ alpha_L)$.

Proof.

move $\Rightarrow H$.

rewrite */rectangular*.

apply $(@inc_trans\ _ _ _ (_ _ (\text{fun } l : L \Rightarrow (alpha_L\ l \cdot _ _ B\ A) \cdot alpha_L\ l)))$.

apply $(@inc_trans\ _ _ _ _ (@comp_capL_distr_l\ _ _ _ _))$.

apply *inc_capL*.

move $\Rightarrow l$.

apply $(@inc_trans\ _ _ _ (((_ _ alpha_L) \cdot _ _ B\ A) \cdot alpha_L\ l))$.

move : *l*.

apply *inc_capL*.

apply *inc_refl*.

apply *comp_inc_compat_ab_a'b*.

apply *comp_inc_compat_ab_a'b*.

apply *inc_capL*.

apply *inc_refl*.

apply *inc_capL*.

move $\Rightarrow l$.

apply $(\text{fun } H' \Rightarrow @inc_trans\ _ _ _ _ H' (H\ l))$.

move : *l*.

apply *inc_capL*.

apply *inc_refl*.

Qed.

Lemma *rectangular_cap* $\{A\ B\ : eqType\} \{alpha\ beta\ : Rel\ A\ B\}$:
 $rectangular\ alpha \rightarrow rectangular\ beta \rightarrow rectangular\ (alpha\ _ _ beta)$.

Proof.

CHAPTER 11. LIBRARY DOMAIN

```

move  $\Rightarrow$   $H$   $H0$ .
rewrite cap_to_capL.
apply rectangular_capL.
induction  $l$ .
apply  $H$ .
apply  $H0$ .
Qed.

```

Lemma 243 (rectangular_comp) *Let $\alpha : A \rightarrow B$, $\beta : B \rightarrow C$ and α or β is a rectangular relation, then $\alpha \cdot \beta$ is also a rectangular relation.*

Lemma *rectangular_comp* $\{A\ B\ C : eqType\} \{alpha : Rel\ A\ B\} \{beta : Rel\ B\ C\}$:
rectangular alpha \vee rectangular beta \rightarrow rectangular (alpha \cdot beta).

Proof.

```

rewrite /rectangular.
case; move  $\Rightarrow$   $H$ .
rewrite -comp_assoc.
apply comp_inc_compat_ab_a'b.
apply (fun  $H' \Rightarrow @inc\_trans \_ \_ \_ \_ H' H$ ).
apply comp_inc_compat_ab_a'b.
rewrite comp_assoc.
apply comp_inc_compat_ab_ab'.
apply inc_alpha_universal.
rewrite comp_assoc comp_assoc.
apply comp_inc_compat_ab_ab'.
apply (fun  $H' \Rightarrow @inc\_trans \_ \_ \_ \_ H' H$ ).
rewrite -comp_assoc -comp_assoc.
apply comp_inc_compat_ab_a'b.
rewrite comp_assoc.
apply comp_inc_compat_ab_ab'.
apply inc_alpha_universal.
Qed.

```

Lemma 244 (rectangular_unit) *Let $\alpha : A \rightarrow B$. Then,*

$$“\alpha \text{ is rectangular}” \Leftrightarrow \exists \mu : I \rightarrow A, \exists \rho : I \rightarrow B, \alpha = \rho^\# \cdot \mu.$$

Lemma *rectangular_unit* $\{A\ B : eqType\} \{alpha : Rel\ A\ B\}$:
rectangular alpha $\leftrightarrow \exists (mu : Rel\ i\ A)(rho : Rel\ i\ B), alpha = mu \# \cdot rho$.

Proof.

```

split; move  $\Rightarrow$   $H$ .
 $\exists ( \_ i\ B \cdot alpha \# )$ .
 $\exists ( \_ i\ A \cdot alpha )$ .

```

CHAPTER 11. LIBRARY DOMAIN

```
rewrite comp_inv inv_invol inv_universal.
rewrite -comp_assoc (@comp_assoc _ _ _ alpha) lemma_for_tarski2.
apply inc_antisym.
apply (@inc_trans _ _ _ _ (@relation_rel_inv_rel _ _ _)).
apply comp_inc_compat_ab_a'b.
apply comp_inc_compat_ab_ab'.
apply inc_alpha_universal.
apply H.
elim H => mu.
elim => rho H0.
rewrite H0.
rewrite /rectangular.
rewrite -comp_assoc.
apply comp_inc_compat_ab_a'b.
rewrite comp_assoc comp_assoc.
apply comp_inc_compat_ab_a.
rewrite unit_identity_is_universal.
apply inc_alpha_universal.
Qed.
```

Chapter 12

Library **Sum_Product**

```
Require Import Basic_Notations.
Require Import Basic_Lemmas.
Require Import Relation_Properties.
Require Import Functions_Mappings.
Require Import Dedekind.
Require Import Conjugate.
Require Import Domain.
Require Import Logic.IndefiniteDescription.
```

12.1 関係の直和

12.1.1 入射対, 関係直和の定義

入射対の存在公理 (Axiom 23) で入射対が存在することまでは仮定済みなので, 実際に入射対 $j : A \rightarrow A + B, k : B \rightarrow A + B$ を定義する関数を定義する.

```
Definition sum_r (A B : eqType):
  {x : (Rel A (sum_eqType A B)) × (Rel B (sum_eqType A B)) |
    (fst x) • (fst x) # = Id A ∧ (snd x) • (snd x) # = Id B ∧
    (fst x) • (snd x) # = A B ∧
    ((fst x) # • (fst x)) ((snd x) # • (snd x)) = Id (sum_eqType A B)}.
apply constructive_indefinite_description.
elim (@pair_of_inclusions A B) ⇒ j.
elim ⇒ k H.
∃ (j,k).
simpl.
apply H.
Defined.
Definition inl_r (A B : eqType):= fst (sval (sum_r A B)).
```


CHAPTER 12. LIBRARY SUM_PRODUCT

Definition $\text{inr_r} (A B : \text{eqType}) := \text{snd} (\text{sval} (\text{sum_r} A B))$.

またこの定義による入射対が, 入射対としての性質 (Axiom 23) $+\alpha$ を満たしていることも事前に証明しておく.

Lemma $\text{inl_id} \{A B : \text{eqType}\} : \text{inl_r} A B \cdot \text{inl_r} A B \# = \text{Id } A$.

Proof.

apply (proj2_sig (sum_r A B)).

Qed.

Lemma $\text{inr_id} \{A B : \text{eqType}\} : \text{inr_r} A B \cdot \text{inr_r} A B \# = \text{Id } B$.

Proof.

apply (proj2_sig (sum_r A B)).

Qed.

Lemma $\text{inl_inr_empty} \{A B : \text{eqType}\} : \text{inl_r} A B \cdot \text{inr_r} A B \# = A B$.

Proof.

apply (proj2_sig (sum_r A B)).

Qed.

Lemma $\text{inr_inl_empty} \{A B : \text{eqType}\} : \text{inr_r} A B \cdot \text{inl_r} A B \# = B A$.

Proof.

apply inv_invol2.

rewrite comp_inv inv_invol inv_empty.

apply inl_inr_empty.

Qed.

Lemma $\text{inl_inr_cup_id} \{A B : \text{eqType}\} :$

$(\text{inl_r} A B \# \cdot \text{inl_r} A B) (\text{inr_r} A B \# \cdot \text{inr_r} A B) = \text{Id} (\text{sum_eqType } A B)$.

Proof.

apply (proj2_sig (sum_r A B)).

Qed.

Lemma $\text{inl_function} \{A B : \text{eqType}\} : \text{function_r} (\text{inl_r} A B)$.

Proof.

move : (proj2_sig (sum_r A B)).

elim $\Rightarrow H$.

elim $\Rightarrow H0$.

elim $\Rightarrow H1 H2$.

split.

rewrite /total_r.

rewrite H.

apply inc_refl.

rewrite /univalent_r.

rewrite -H2.

apply cup_l.

Qed.

CHAPTER 12. LIBRARY SUM_PRODUCT

Lemma *inr_function* $\{A\ B : eqType\} : function_r\ (inr_r\ A\ B).$

Proof.

```
move : (proj2_sig (sum_r A B)).
elim  $\Rightarrow H$ .
elim  $\Rightarrow H0$ .
elim  $\Rightarrow H1\ H2$ .
split.
rewrite /total_r.
rewrite  $H0$ .
apply inc_refl.
rewrite /univalent_r.
rewrite - $H2$ .
apply cup_r.
Qed.
```

さらに $\alpha : A \rightarrow C$ と $\beta : B \rightarrow C$ の関係直和 $\alpha \perp \beta : A + B \rightarrow C$ を, $\alpha \perp \beta := j^\# \cdot \alpha \sqcup k^\# \cdot \beta$ で定義する.

Definition *Rel_sum* $\{A\ B\ C : eqType\} (alpha : Rel\ A\ C) (\mathbf{beta} : Rel\ B\ C) :=$
 $(inl_r\ A\ B \# \cdot alpha) \quad (inr_r\ A\ B \# \cdot \mathbf{beta}).$

12.1.2 関係直和の性質

Lemma 245 (sum_inc_compat) *Let $\alpha, \alpha' : A \rightarrow C$ and $\beta, \beta' : B \rightarrow C$. Then,*

$$\alpha \sqsubseteq \alpha' \wedge \beta \sqsubseteq \beta' \Rightarrow \alpha \perp \beta \sqsubseteq \alpha' \perp \beta'.$$

Lemma *sum_inc_compat*

$\{A\ B\ C : eqType\} \{alpha\ alpha' : Rel\ A\ C\} \{\mathbf{beta}\ beta' : Rel\ B\ C\} :$
 $alpha \quad alpha' \rightarrow \mathbf{beta} \quad beta' \rightarrow Rel_sum\ alpha\ \mathbf{beta} \quad Rel_sum\ alpha'\ beta'.$

Proof.

```
move  $\Rightarrow H\ H0$ .
apply cup_inc_compat.
apply (comp_inc_compat_ab_ab' H).
apply (comp_inc_compat_ab_ab' H0).
Qed.
```

Lemma 246 (sum_inc_compat_l) *Let $\alpha : A \rightarrow C$ and $\beta, \beta' : B \rightarrow C$. Then,*

$$\beta \sqsubseteq \beta' \Rightarrow \alpha \perp \beta \sqsubseteq \alpha \perp \beta'.$$

Lemma *sum_inc_compat_l*

CHAPTER 12. LIBRARY SUM_PRODUCT

$\{A\ B\ C : eqType\} \{alpha : Rel\ A\ C\} \{beta\ beta' : Rel\ B\ C\}:$
 $beta\ beta' \rightarrow Rel_sum\ alpha\ beta\ Rel_sum\ alpha\ beta'.$

Proof.

move $\Rightarrow H$.

apply (sum_inc_compat (@inc_refl _ _ alpha) H).

Qed.

Lemma 247 (sum_inc_compat_r) *Let $\alpha, \alpha' : A \rightarrow C$ and $\beta : B \rightarrow C$. Then,*

$$\alpha \sqsubseteq \alpha' \Rightarrow \alpha \perp \beta \sqsubseteq \alpha' \perp \beta.$$

Lemma sum_inc_compat_r

$\{A\ B\ C : eqType\} \{alpha\ alpha' : Rel\ A\ C\} \{beta : Rel\ B\ C\}:$
 $alpha\ alpha' \rightarrow Rel_sum\ alpha\ beta\ Rel_sum\ alpha'\ beta.$

Proof.

move $\Rightarrow H$.

apply (sum_inc_compat H (@inc_refl _ _ beta)).

Qed.

Lemma 248 (total_sum) *Let $\alpha : A \rightarrow C$ and $\beta : B \rightarrow C$ are total relations, then $\alpha \perp \beta$ is also a total relation.*

Lemma total_sum $\{A\ B\ C : eqType\} \{alpha : Rel\ A\ C\} \{beta : Rel\ B\ C\}:$
 $total_r\ alpha \rightarrow total_r\ beta \rightarrow total_r\ (Rel_sum\ alpha\ beta).$

Proof.

move $\Rightarrow H\ H0$.

rewrite /total_r/Rel_sum.

rewrite -inl_inr_cup_id inv_cup_distr comp_cup_distr_l comp_cup_distr_r comp_cup_distr_r.

rewrite comp_inv comp_inv inv_invol inv_invol.

apply cup_inc_compat.

apply (fun H' \Rightarrow @inc_trans _ _ _ _ H' (@cup_l _ _ _)).

rewrite comp_assoc -(@comp_assoc _ _ _ _ alpha).

apply comp_inc_compat_ab_ab'.

apply (comp_inc_compat_b_ab H).

apply (fun H' \Rightarrow @inc_trans _ _ _ _ H' (@cup_r _ _ _)).

rewrite comp_assoc -(@comp_assoc _ _ _ _ beta).

apply comp_inc_compat_ab_ab'.

apply (comp_inc_compat_b_ab H0).

Qed.

Lemma 249 (univalent_sum) *Let $\alpha : A \rightarrow C$ and $\beta : B \rightarrow C$ are univalent relations, then $\alpha \perp \beta$ is also a univalent relation.*

CHAPTER 12. LIBRARY SUM_PRODUCT

Lemma *univalent_sum* $\{A\ B\ C : \text{eqType}\} \{alpha : \text{Rel}\ A\ C\} \{beta : \text{Rel}\ B\ C\}$:
 $\text{univalent_r}\ alpha \rightarrow \text{univalent_r}\ beta \rightarrow \text{univalent_r}\ (\text{Rel_sum}\ alpha\ beta).$

Proof.

`move \Rightarrow H H0.`

`rewrite /univalent_r/Rel_sum.`

`rewrite inv_cup_distr comp_cup_distr_l comp_cup_distr_r comp_cup_distr_r.`

`rewrite comp_inv comp_inv inv_invol inv_invol.`

`rewrite comp_assoc -(@comp_assoc _ _ _ (inl_r A B)) inl_id comp_id_l.`

`rewrite comp_assoc -(@comp_assoc _ _ _ (inr_r A B)) inr_inl_empty comp_empty_l
 comp_empty_r cup_empty.`

`rewrite -cup_assoc comp_assoc -(@comp_assoc _ _ _ (inl_r A B)) inl_inr_empty comp_empty_l
 comp_empty_r cup_empty.`

`rewrite comp_assoc -(@comp_assoc _ _ _ (inr_r A B)) inr_id comp_id_l.`

`apply inc_cup.`

`split.`

`apply H.`

`apply H0.`

Qed.

Lemma 250 (function_sum) *Let $\alpha : A \rightarrow C$ and $\beta : B \rightarrow C$ be functions, then $\alpha \perp \beta$ is also a function.*

Lemma *function_sum* $\{A\ B\ C : \text{eqType}\} \{alpha : \text{Rel}\ A\ C\} \{beta : \text{Rel}\ B\ C\}$:
 $\text{function_r}\ alpha \rightarrow \text{function_r}\ beta \rightarrow \text{function_r}\ (\text{Rel_sum}\ alpha\ beta).$

Proof.

`elim \Rightarrow H H0.`

`elim \Rightarrow H1 H2.`

`split.`

`apply (total_sum H H1).`

`apply (univalent_sum H0 H2).`

Qed.

Lemma 251 (sum_conjugate) *Let $\alpha : A \rightarrow C$, $\beta : B \rightarrow C$ and $\gamma : A + B \rightarrow C$ be relations, $j : A \rightarrow A + B$ and $k : B \rightarrow A + B$ be inclusions. Then,*

$$j \cdot \gamma = \alpha \wedge k \cdot \gamma = \beta \Leftrightarrow \gamma = \alpha \perp \beta.$$

Lemma *sum_conjugate*

$\{A\ B\ C : \text{eqType}\} \{alpha : \text{Rel}\ A\ C\} \{beta : \text{Rel}\ B\ C\} \{gamma : \text{Rel}\ (\text{sum_eqType}\ A\ B)\ C\}$:

$\text{inl_r}\ A\ B \cdot \text{gamma} = \alpha \wedge \text{inr_r}\ A\ B \cdot \text{gamma} = \text{beta} \Leftrightarrow$
 $\text{gamma} = \text{Rel_sum}\ alpha\ \text{beta}.$

Proof.

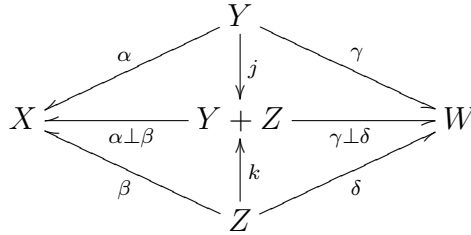
```

split; move => H.
elim H => H0 H1.
rewrite -(@comp_id_l _ _ gamma).
rewrite -inl_inr_cup_id comp_cup_distr_r comp_assoc comp_assoc.
by [rewrite H0 H1].
split.
rewrite H comp_cup_distr_l -comp_assoc -comp_assoc.
rewrite inl_id inl_inr_empty comp_id_l comp_empty_l.
by [rewrite cup_empty].
rewrite H comp_cup_distr_l -comp_assoc -comp_assoc.
rewrite inr_id inr_inl_empty comp_id_l comp_empty_l.
by [rewrite cup_comm cup_empty].
Qed.

```

Lemma 252 (sum_comp) *In below figure,*

$$(\alpha \perp \beta)^\# \cdot (\gamma \perp \delta) = \alpha^\# \cdot \gamma \sqcup \beta^\# \cdot \delta.$$



```

Lemma sum_comp {W X Y Z : eqType}
  {alpha : Rel Y X} {beta : Rel Z X} {gamma : Rel Y W} {delta : Rel Z W}:
  (Rel_sum alpha beta) # • Rel_sum gamma delta =
  (alpha # • gamma)    (beta # • delta).

```

Proof.

```

rewrite /Rel_sum.
rewrite inv_cup_distr comp_cup_distr_l comp_cup_distr_r comp_cup_distr_r.
rewrite comp_inv comp_inv inv_invol inv_invol.
apply f_equal2.
rewrite comp_assoc -(@comp_assoc _ _ _ (inl_r Y Z)) inl_id comp_id_l.
by [rewrite comp_assoc -(@comp_assoc _ _ _ (inr_r Y Z)) inr_inl_empty comp_empty_l
    comp_empty_r cup_empty].
rewrite comp_assoc -(@comp_assoc _ _ _ (inl_r Y Z)) inl_inr_empty comp_empty_l
    comp_empty_r cup_comm cup_empty.
by [rewrite comp_assoc -(@comp_assoc _ _ _ (inr_r Y Z)) inr_id comp_id_l].
Qed.

```

12.1.3 分配法則

Lemma 253 (sum_cap_distr_l) *Let $\alpha : A \rightarrow C$ and $\beta, \beta' : B \rightarrow C$. Then,*

$$\alpha \perp (\beta \sqcap \beta') \sqsubseteq (\alpha \perp \beta) \sqcap (\alpha \perp \beta').$$

Lemma *sum_cap_distr_l*

$\{A\ B\ C : eqType\} \{alpha : Rel\ A\ C\} \{beta\ beta' : Rel\ B\ C\} :$
 $Rel_sum\ alpha\ (beta\ \beta')\ (Rel_sum\ alpha\ beta\ Rel_sum\ alpha\ beta').$

Proof.

rewrite -cup_cap_distr_l.

apply cup_inc_compat_l.

apply comp_cap_distr_l.

Qed.

Lemma 254 (sum_cap_distr_r) *Let $\alpha, \alpha' : A \rightarrow C$ and $\beta : B \rightarrow C$. Then,*

$$(\alpha \sqcap \alpha') \perp \beta \sqsubseteq (\alpha \perp \beta) \sqcap (\alpha' \perp \beta).$$

Lemma *sum_cap_distr_r*

$\{A\ B\ C : eqType\} \{alpha\ alpha' : Rel\ A\ C\} \{beta : Rel\ B\ C\} :$
 $Rel_sum\ (alpha\ alpha')\ beta\ (Rel_sum\ alpha\ beta\ Rel_sum\ alpha'\ beta').$

Proof.

rewrite -cup_cap_distr_r.

apply cup_inc_compat_r.

apply comp_cap_distr_l.

Qed.

Lemma 255 (sum_cup_distr_l) *Let $\alpha : A \rightarrow C$ and $\beta, \beta' : B \rightarrow C$. Then,*

$$\alpha \perp (\beta \sqcup \beta') = (\alpha \perp \beta) \sqcup (\alpha \perp \beta').$$

Lemma *sum_cup_distr_l*

$\{A\ B\ C : eqType\} \{alpha : Rel\ A\ C\} \{beta\ beta' : Rel\ B\ C\} :$
 $Rel_sum\ alpha\ (beta\ \beta') = Rel_sum\ alpha\ beta\ Rel_sum\ alpha\ beta'.$

Proof.

rewrite -cup_assoc (@cup_comm _ _ (Rel_sum alpha beta)) -cup_assoc.

by [rewrite cup_idem cup_assoc -comp_cup_distr_l].

Qed.

CHAPTER 12. LIBRARY SUM_PRODUCT

Lemma 256 (sum_cup_distr_r) *Let $\alpha, \alpha' : A \rightarrow C$ and $\beta : B \rightarrow C$. Then,*

$$(\alpha \sqcup \alpha') \perp \beta = (\alpha \perp \beta) \sqcup (\alpha' \perp \beta).$$

Lemma *sum_cup_distr_r*

$\{A\ B\ C : eqType\} \{alpha\ alpha' : Rel\ A\ C\} \{beta : Rel\ B\ C\} :$
 $Rel_sum\ (alpha\ \ alpha')\ beta = (Rel_sum\ alpha\ beta\ \ Rel_sum\ alpha'\ beta).$

Proof.

`rewrite cup_assoc (@cup_comm _ _ (inr_r A B # • beta)) cup_assoc.`

`by [rewrite cup_idem -cup_assoc -comp_cup_distr_l].`

Qed.

Lemma 257 (comp_sum_distr_r) *Let $\alpha : A \rightarrow C$, $\beta : B \rightarrow C$ and $\gamma : C \rightarrow D$. Then,*

$$(\alpha \perp \beta) \cdot \gamma = \alpha \cdot \gamma \perp \beta \cdot \gamma.$$

Lemma *comp_sum_distr_r*

$\{A\ B\ C\ D : eqType\} \{alpha : Rel\ A\ C\} \{beta : Rel\ B\ C\} \{gamma : Rel\ C\ D\} :$
 $(Rel_sum\ alpha\ beta) \cdot gamma = Rel_sum\ (alpha \cdot gamma)\ (beta \cdot gamma).$

Proof.

`by [rewrite comp_cup_distr_r comp_assoc comp_assoc].`

Qed.

12.2 関係の直積

12.2.1 射影対, 関係直積の定義

射影対の存在公理 (Axiom 24) で射影対が存在することまでは仮定済みなので, 実際に射影対 $p : A \times B \rightarrow A, k : A \times B \rightarrow B$ を定義する関数を定義する.

Definition *prod_r* ($A\ B : eqType$):

$\{x : (Rel\ (prod_eqType\ A\ B)\ A) \times (Rel\ (prod_eqType\ A\ B)\ B) \mid$
 $(fst\ x) \# \cdot (snd\ x) = A\ B \wedge$
 $((fst\ x) \cdot (fst\ x) \#) \cdot ((snd\ x) \cdot (snd\ x) \#) = Id\ (prod_eqType\ A\ B) \wedge$
 $univalent_r\ (fst\ x) \wedge univalent_r\ (snd\ x)\}.$

`apply constructive_indefinite_description.`

`elim (@pair_of_projections A B) => p.`

`elim => q H.`

`∃ (p,q).`

`simpl.`

`apply H.`

Defined.

Definition $\text{fst}_r (A B : \text{eqType}) := \text{fst} (\text{sva} (\text{prod}_r A B))$.

Definition $\text{snd}_r (A B : \text{eqType}) := \text{snd} (\text{sva} (\text{prod}_r A B))$.

またこの定義による射影対が, 射影対としての性質 (Axiom 24) $+\alpha$ を満たしていることも事前に証明しておく.

Lemma $\text{fst_snd_universal} \{A B : \text{eqType}\} : \text{fst}_r A B \# \cdot \text{snd}_r A B = A B$.

Proof.

`apply (proj2_sig (prod_r A B)).`

Qed.

Lemma $\text{snd_fst_universal} \{A B : \text{eqType}\} : \text{snd}_r A B \# \cdot \text{fst}_r A B = B A$.

Proof.

`apply inv_invol2.`

`rewrite comp_inv inv_invol inv_universal.`

`apply fst_snd_universal.`

Qed.

Lemma $\text{fst_snd_cap_id} \{A B : \text{eqType}\} :$

$(\text{fst}_r A B \cdot \text{fst}_r A B \#) \quad (\text{snd}_r A B \cdot \text{snd}_r A B \#) = \text{Id} (\text{prod_eqType} A B)$.

Proof.

`apply (proj2_sig (prod_r A B)).`

Qed.

Lemma $\text{fst_function} \{A B : \text{eqType}\} : \text{function}_r (\text{fst}_r A B)$.

Proof.

`move : (proj2_sig (prod_r A B)).`

`elim $\Rightarrow H$.`

`elim $\Rightarrow H0 H1$.`

`split.`

`rewrite /total_r.`

`rewrite -H0.`

`apply cap_l.`

`apply H1.`

Qed.

Lemma $\text{snd_function} \{A B : \text{eqType}\} : \text{function}_r (\text{snd}_r A B)$.

Proof.

`move : (proj2_sig (prod_r A B)).`

`elim $\Rightarrow H$.`

`elim $\Rightarrow H0 H1$.`

`split.`

`rewrite /total_r.`

`rewrite -H0.`

`apply cap_r.`

CHAPTER 12. LIBRARY SUM_PRODUCT

alpha *alpha'* \rightarrow *Rel_prod* *alpha* **beta** *Rel_prod* *alpha'* **beta**.

Proof.

move \Rightarrow *H*.

apply (*prod_inc_compat* *H* (@*inc_refl* _ _ **beta**)).

Qed.

Lemma 261 (total_prod) *Let $\alpha : A \rightarrow B$ and $\beta : A \rightarrow C$ be total relations, then $\alpha \top \beta$ is also a total relation.*

Lemma *total_prod* {*A B C* : *eqType*} {*alpha* : *Rel A B*} {**beta** : *Rel A C*}:
total_r alpha \rightarrow *total_r* **beta** \rightarrow *total_r* (*Rel_prod* *alpha* **beta**).

Proof.

move \Rightarrow *H H0*.

rewrite *domain_total cap_domain cap_comm*.

apply *Logic.eq_sym*.

apply *inc_def1*.

apply (@*inc_trans* _ _ _ _ *H*).

rewrite *comp_inv inv_invol comp_assoc*.

apply *comp_inc_compat_ab_ab'*.

apply (@*inc_trans* _ _ _ (alpha # • (b beta • b beta #))).

apply (*comp_inc_compat_a_ab H0*).

rewrite -*comp_assoc* -*comp_assoc fst_snd_universal*.

apply *comp_inc_compat_ab_a'b*.

apply *inc_alpha_universal*.

Qed.

Lemma 262 (univalent_prod) *Let $\alpha : A \rightarrow B$ and $\beta : A \rightarrow C$ be univalent relations, then $\alpha \top \beta$ is also a univalent relation.*

Lemma *univalent_prod* {*A B C* : *eqType*} {*alpha* : *Rel A B*} {**beta** : *Rel A C*}:
univalent_r alpha \rightarrow *univalent_r* **beta** \rightarrow *univalent_r* (*Rel_prod* *alpha* **beta**).

Proof.

move \Rightarrow *H H0*.

rewrite /*univalent_r*/ *Rel_prod*.

rewrite *inv_cap_distr comp_inv inv_invol comp_inv inv_invol*.

apply (@*inc_trans* _ _ _ _ (@*comp_cap_distr_l* _ _ _ _ _)).

rewrite -*fst_snd_cap_id*.

apply *cap_inc_compat*.

apply (@*inc_trans* _ _ _ _ (@*comp_cap_distr_r* _ _ _ _ _)).

apply (@*inc_trans* _ _ _ _ (@*cap_l* _ _ _ _)).

rewrite *comp_assoc* -(@*comp_assoc* _ _ _ _ *alpha*).

apply *comp_inc_compat_ab_ab'*.

apply (*comp_inc_compat_ab_b H*).

CHAPTER 12. LIBRARY SUM_PRODUCT

```

apply (@inc_trans _ _ _ _ (@comp_cap_distr_r _ _ _ _ _)).
apply (@inc_trans _ _ _ _ (@cap_r _ _ _ _ _)).
rewrite comp_assoc -(@comp_assoc _ _ _ _ beta).
apply comp_inc_compat_ab_ab'.
apply (comp_inc_compat_ab_b H0).
Qed.

```

Lemma 263 (function_prod) *Let $\alpha : A \rightarrow B$ and $\beta : A \rightarrow C$ be functions, then $\alpha \top \beta$ is also a function.*

Lemma function_prod $\{A B C : eqType\} \{alpha : Rel A B\} \{beta : Rel A C\}$:
 $function_r\ alpha \rightarrow function_r\ beta \rightarrow function_r\ (Rel_prod\ alpha\ beta)$.

Proof.

```

elim  $\Rightarrow$  H H0.
elim  $\Rightarrow$  H1 H2.
split.
apply (total_prod H H1).
apply (univalent_prod H0 H2).
Qed.

```

Lemma 264 (prod_fst_surjection) *Let $p : B \times C \rightarrow B$ be a projection. Then,*

$$“p \text{ is a surjection}” \Leftrightarrow \forall D, \nabla_{BD} = \nabla_{BC} \cdot \nabla_{CD}.$$

Lemma prod_fst_surjection $\{B C : eqType\}$:
 $surjection_r\ (fst_r\ B\ C) \Leftrightarrow \forall D : eqType, \quad B\ D = \quad B\ C \cdot \quad C\ D$.

Proof.

```

split; move  $\Rightarrow$  H.
move  $\Rightarrow$  D.
elim H  $\Rightarrow$  H0 H1.
apply inc_antisym.
apply (@inc_trans _ _ _ ((fst_r B C #  $\cdot$  (fst_r B C #) #)  $\cdot$  B D)).
apply (comp_inc_compat_b_ab H1).
rewrite inv_invol.
apply (@inc_trans _ _ _ (((fst_r B C #  $\cdot$  snd_r B C)  $\cdot$  (snd_r B C #  $\cdot$  fst_r B C))  $\cdot$  B D)).
apply comp_inc_compat_ab_a'b.
rewrite comp_assoc -(@comp_assoc _ _ _ _ (snd_r B C)).
apply comp_inc_compat_ab_ab'.
apply comp_inc_compat_b_ab.
apply snd_function.
rewrite (@comp_assoc _ _ _ _ _ (B D)).
apply comp_inc_compat.

```

```

apply inc_alpha_universal.
apply inc_alpha_universal.
apply inc_alpha_universal.
split.
apply fst_function.
rewrite /total_r.
rewrite -(@cap_universal _ _ (Id B)) (H B) -(@fst_snd_universal B C) cap_comm comp_assoc.
apply (@inc_trans _ _ _ _ (@dedekind1 _ _ _ _ _)).
apply comp_inc_compat_ab_ab'.
rewrite comp_id_r.
apply cap_r.
Qed.

```

Lemma 265 (prod_snd_surjection) *Let $q : B \times C \rightarrow C$ be a projection. Then,*

$$“q \text{ is a surjection}” \Leftrightarrow \forall D, \nabla_{CD} = \nabla_{CB} \cdot \nabla_{BD}.$$

Lemma *prod_snd_surjection* { $B \ C : eqType$ }:
 $surjection_r \ (snd_r \ B \ C) \leftrightarrow \forall \ D : eqType, \quad C \ D = \quad C \ B \cdot \quad B \ D.$

Proof.

```

split; move => H.
move => D.
elim H => H0 H1.
apply inc_antisym.
apply (@inc_trans _ _ _ ((snd_r B C # · (snd_r B C #) #) · C D)).
apply (comp_inc_compat_b_ab H1).
rewrite inv_invol.
apply (@inc_trans _ _ _ (((snd_r B C # · fst_r B C) · (fst_r B C # · snd_r B C)) · C D)).
apply comp_inc_compat_ab_a'b.
rewrite comp_assoc -(@comp_assoc _ _ _ (fst_r B C)).
apply comp_inc_compat_ab_ab'.
apply comp_inc_compat_b_ab.
apply fst_function.
rewrite (@comp_assoc _ _ _ _ _ (C D)).
apply comp_inc_compat.
apply inc_alpha_universal.
apply inc_alpha_universal.
apply inc_alpha_universal.
split.
apply snd_function.
rewrite /total_r.
rewrite -(@cap_universal _ _ (Id C)) (H C) -(@snd_fst_universal B C) cap_comm comp_assoc.

```

CHAPTER 12. LIBRARY SUM_PRODUCT

```

apply (@inc_trans _ _ _ _ (@dedekind1 _ _ _ _ _)).
apply comp_inc_compat_ab_ab'.
rewrite comp_id_r.
apply cap_r.
Qed.

```

Lemma 266 (prod_fst_domain1) *Let $p : B \times C \rightarrow B$ be a projection, $\alpha : A \rightarrow B$ and $\beta : A \rightarrow C$. Then,*

$$(\alpha \top \beta) \cdot p = \lfloor \beta \rfloor \cdot \alpha.$$

Lemma prod_fst_domain1 $\{A\ B\ C : eqType\} \{alpha : Rel\ A\ B\} \{beta : Rel\ A\ C\}$:
 $(Rel_prod\ alpha\ beta) \cdot fst_r\ B\ C = domain\ beta \cdot alpha$.

Proof.

```

rewrite (@comp_inv_inv A A) domain_inv.
rewrite domain_universal2 inv_cap_distr comp_inv inv_invol inv_invol inv_universal.
rewrite -snd_fst_universal.
apply inc_antisym.
apply (@inc_trans _ _ _ _ (@comp_cap_distr_r _ _ _ _ _)).
rewrite comp_assoc comp_assoc.
apply cap_inc_compat_r.
apply comp_inc_compat_ab_a.
apply fst_function.
rewrite cap_comm -comp_assoc.
apply (@inc_trans _ _ _ _ (@dedekind2 _ _ _ _ _)).
rewrite cap_comm.
apply inc_refl.
Qed.

```

Lemma 267 (prod_fst_domain2) *Let $p : B \times C \rightarrow B$ be a projection, $\alpha : A \rightarrow B$ and $\beta : A \rightarrow C$. Then,*

$$(\alpha \top \beta) \cdot p = \alpha \Leftrightarrow \lfloor \alpha \rfloor \sqsubseteq \lfloor \beta \rfloor.$$

Lemma prod_fst_domain2 $\{A\ B\ C : eqType\} \{alpha : Rel\ A\ B\} \{beta : Rel\ A\ C\}$:
 $(Rel_prod\ alpha\ beta) \cdot fst_r\ B\ C = alpha \leftrightarrow domain\ alpha \sqsubseteq domain\ beta$.

Proof.

```

rewrite prod_fst_domain1.
split; move => H.
apply domain_lemma2b.
assert ((domain beta \cdot alpha) ((beta \cdot beta #) \cdot alpha)).
apply comp_inc_compat_ab_a'b.
apply cap_l.
rewrite H in H0.
apply H0.

```

```

apply inc_antisym.
apply comp_inc_compat_ab_b.
apply cap_r.
apply (@inc_trans _ _ _ (domain alpha · alpha)).
rewrite domain_comp_alpha1.
apply inc_refl.
apply (comp_inc_compat_ab_a'b H).
Qed.

```

Lemma 268 (prod_snd_domain1) *Let $q : B \times C \rightarrow C$ be a projection, $\alpha : A \rightarrow B$ and $\beta : A \rightarrow C$. Then,*

$$(\alpha \top \beta) \cdot q = \lfloor \alpha \rfloor \cdot \beta.$$

Lemma prod_snd_domain1 $\{A B C : eqType\} \{alpha : Rel A B\} \{beta : Rel A C\}$:
 $(Rel_prod\ alpha\ beta) \cdot snd_r\ B\ C = domain\ alpha \cdot beta$.

Proof.

```

rewrite (@comp_inv_inv A A) domain_inv.
rewrite domain_universal2 inv_cap_distr comp_inv inv_invol inv_invol inv_universal.
rewrite fst_snd_universal.
apply inc_antisym.
apply (@inc_trans _ _ _ _ (@comp_cap_distr_r _ _ _ _ _)).
rewrite comp_assoc comp_assoc cap_comm.
apply cap_inc_compat_r.
apply comp_inc_compat_ab_a.
apply snd_function.
rewrite cap_comm -comp_assoc.
apply dedekind2.
Qed.

```

Lemma 269 (prod_snd_domain2) *Let $q : B \times C \rightarrow C$ be a projection, $\alpha : A \rightarrow B$ and $\beta : A \rightarrow C$. Then,*

$$(\alpha \top \beta) \cdot q = \beta \Leftrightarrow \lfloor \beta \rfloor \sqsubseteq \lfloor \alpha \rfloor.$$

Lemma prod_snd_domain2 $\{A B C : eqType\} \{alpha : Rel A B\} \{beta : Rel A C\}$:
 $(Rel_prod\ alpha\ beta) \cdot snd_r\ B\ C = beta \leftrightarrow domain\ beta \sqsubseteq domain\ alpha$.

Proof.

```

rewrite prod_snd_domain1.
split; move => H.
apply domain_lemma2b.
assert ((domain alpha · beta) ((alpha · alpha #) · beta)).
apply comp_inc_compat_ab_a'b.
apply cap_l.
rewrite H in H0.

```

```

apply H0.
apply inc_antisym.
apply comp_inc_compat_ab_b.
apply cap_r.
apply (@inc_trans _ _ _ (domain beta • beta)).
rewrite domain_comp_alpha1.
apply inc_refl.
apply (comp_inc_compat_ab_a'b H).
Qed.

```

Lemma 270 (prod_to_cap) *Let $\alpha : A \rightarrow B$ and $\beta : A \rightarrow C$. Then,*

$$\lfloor \alpha \top \beta \rfloor = \lfloor \alpha \rfloor \sqcap \lfloor \beta \rfloor.$$

Lemma prod_to_cap $\{A\ B\ C : \text{eqType}\} \{\alpha : \text{Rel } A\ B\} \{\beta : \text{Rel } A\ C\}$:
 $\text{domain } (\text{Rel_prod } \alpha \ \beta) = \text{domain } \alpha \quad \text{domain } \beta.$

Proof.

```

replace (domain (Rel_prod alpha beta)) with (domain (Rel_prod alpha beta • snd_r B C)).
rewrite prod_snd_domain1 comp_domain8.
apply dedekind_id3.
apply cap_r.
apply cap_r.
apply cap_r.
apply comp_domain3.
apply snd_function.
Qed.

```

Lemma 271 (prod_conjugate1) *Let $\alpha : A \rightarrow B$ and $\beta : A \rightarrow C$ be functions, $p : B \times C \rightarrow B$ and $q : B \times C \rightarrow C$ be projections. Then,*

$$(\alpha \top \beta) \cdot p = \alpha \wedge (\alpha \top \beta) \cdot q = \beta.$$

Lemma prod_conjugate1 $\{A\ B\ C : \text{eqType}\} \{\alpha : \text{Rel } A\ B\} \{\beta : \text{Rel } A\ C\}$:
 $\text{function_r } \alpha \rightarrow \text{function_r } \beta \rightarrow$
 $\text{Rel_prod } \alpha \ \beta \cdot \text{fst_r } B\ C = \alpha \wedge \text{Rel_prod } \alpha \ \beta \cdot \text{snd_r } B\ C = \beta.$

Proof.

```

move => H H0.
split.
rewrite prod_fst_domain1.
elim H0 => H1 H2.
apply inc_def1 in H1.
rewrite /domain.

```

CHAPTER 12. LIBRARY SUM_PRODUCT

```
by [rewrite cap_comm -H1 comp_id_l].
rewrite prod_snd_domain1.
elim H ⇒ H1 H2.
apply inc_def1 in H1.
rewrite /domain.
by [rewrite cap_comm -H1 comp_id_l].
Qed.
```

Lemma 272 (prod_conjugate2) *Let $\gamma : A \rightarrow B \times C$ be a function, $p : B \times C \rightarrow B$ and $q : B \times C \rightarrow C$ be projections. Then,*

$$(\gamma \cdot p)^\top (\gamma \cdot q) = \gamma.$$

Lemma prod_conjugate2 $\{A\ B\ C : \text{eqType}\} \{ \text{gamma} : \text{Rel } A\ (\text{prod_eqType } B\ C) \}$:
 $\text{function_r gamma} \rightarrow \text{Rel_prod } (\text{gamma} \cdot \text{fst_r } B\ C) (\text{gamma} \cdot \text{snd_r } B\ C) = \text{gamma}.$

Proof.

```
move ⇒ H.
rewrite /Rel_prod.
rewrite comp_assoc comp_assoc -(function_cap_distr_l H).
by [rewrite fst_snd_cap_id comp_id_r].
Qed.
```

Lemma 273 (diagonal_conjugate) *Let $p : B \times C \rightarrow B$ and $q : B \times C \rightarrow C$ be projections. Then,*

$$\frac{\alpha : A \rightarrow B \quad \alpha = p^\sharp \cdot u \cdot q}{u \sqsubseteq \text{id}_{A \times B} \quad u = [p \cdot \alpha \sqcap q]}.$$

Lemma diagonal_conjugate $\{A\ B : \text{eqType}\} \{ \text{alpha} : \text{Rel } A\ B \}$:
 $\text{conjugate } A\ B\ (\text{prod_eqType } A\ B) (\text{prod_eqType } A\ B)$
 $\text{True_r } (\text{fun } u \Rightarrow u \quad \text{Id } (\text{prod_eqType } A\ B))$
 $(\text{fun } u \Rightarrow (\text{fst_r } A\ B \# \cdot u) \cdot \text{snd_r } A\ B)$
 $(\text{fun } \text{alpha} \Rightarrow \text{domain } ((\text{fst_r } A\ B \cdot \text{alpha}) \quad \text{snd_r } A\ B)).$

Proof.

```
split.
move ⇒ alpha0 H.
split.
apply cap_r.
rewrite cap_domain.
apply inc_antisym.
apply (@inc_trans _ _ ((fst_r A B # · ((fst_r A B · alpha0) · snd_r A B #)) · snd_r
A B)).
apply comp_inc_compat_ab_a'b.
apply comp_inc_compat_ab_ab'.
```



```
apply cap_l.
rewrite comp_assoc comp_assoc -comp_assoc -(@comp_assoc _ _ _ (fst_r A B #)).
apply (@inc_trans _ _ _ ((fst_r A B # • fst_r A B) • alpha0)).
apply comp_inc_compat_ab_a.
apply snd_function.
apply comp_inc_compat_ab_b.
apply fst_function.
apply (@inc_trans _ _ _ (alpha0 ((fst_r A B # • Id (prod_eqType A B)) • snd_r A B))).
rewrite comp_id_r fst_snd_universal cap_universal.
apply inc_refl.
rewrite cap_comm.
apply (@inc_trans _ _ _ _ (@dedekind2 _ _ _ _ _)).
apply comp_inc_compat_ab_a'b.
apply (@inc_trans _ _ _ _ (@dedekind1 _ _ _ _ _)).
apply comp_inc_compat_ab_ab'.
rewrite cap_comm inv_invol comp_assoc.
apply inc_refl.
move ⇒ u H.
split.
by [].
replace ((fst_r A B • ((fst_r A B # • u) • snd_r A B)) snd_r A B) with (u • snd_r A B).
apply domain_inc_id in H.
move : (@snd_function A B) ⇒ H0.
elim H0 ⇒ H1 H2.
by [rewrite (comp_domain3 H1) H].
rewrite comp_assoc -comp_assoc.
apply inc_antisym.
apply (@inc_trans _ _ _ ((u • snd_r A B) snd_r A B)).
apply inc_cap.
split.
apply inc_refl.
apply (comp_inc_compat_ab_b H).
apply cap_inc_compat_r.
apply comp_inc_compat_b_ab.
apply fst_function.
apply (@inc_trans _ _ _ _ (@dedekind2 _ _ _ _ _)).
apply comp_inc_compat_ab_b.
rewrite -fst_snd_cap_id.
apply cap_inc_compat_l.
apply comp_inc_compat_ab_ab'.
```

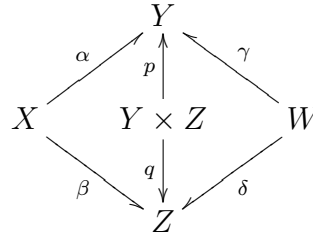
apply *inc_inv*.
 apply (*comp_inc_compat_ab_b* H).
 Qed.

12.2.3 鋭敏性

この節の補題は以下の 1 つのみだが、証明が異様に長いため単独の節を設ける。

Lemma 274 (sharpness) *In below figure,*

$$\alpha \cdot \gamma^\# \sqcap \beta \cdot \delta^\# = (\alpha \cdot p^\# \sqcap \beta \cdot q^\#) \cdot (p \cdot \gamma^\# \sqcap q \cdot \delta^\#).$$



Lemma sharpness {W X Y Z : eqType}
 {alpha : Rel X Y} {beta : Rel X Z} {gamma : Rel W Y} {delta : Rel W Z} :
 (alpha · gamma #) (beta · delta #) =
 ((alpha · fst_r Y Z #) (beta · snd_r Y Z #))
 · ((fst_r Y Z · gamma #) (snd_r Y Z · delta #)).

Proof.

apply *inc_antisym*.
 move : (rationality _ _ alpha) ⇒ H.
 move : (rationality _ _ beta) ⇒ H0.
 move : (rationality _ _ (gamma #)) ⇒ H1.
 move : (rationality _ _ (delta #)) ⇒ H2.
 elim H ⇒ R.
 elim ⇒ f0.
 elim ⇒ g0 H3.
 elim H0 ⇒ R0.
 elim ⇒ f1.
 elim ⇒ g1 H4.
 elim H1 ⇒ R1.
 elim ⇒ h0.
 elim ⇒ k0 H5.
 elim H2 ⇒ R2.
 elim ⇒ h1.
 elim ⇒ k1 H6.

CHAPTER 12. LIBRARY SUM_PRODUCT

```

move : (rationality _ _ (g0 · h0 #)) ⇒ H7.
move : (rationality _ _ (g1 · h1 #)) ⇒ H8.
move : (rationality _ _ ((alpha · gamma #) (beta · delta #))) ⇒ H9.
elim H7 ⇒ R3.
elim ⇒ s0.
elim ⇒ t0 H10.
elim H8 ⇒ R4.
elim ⇒ s1.
elim ⇒ t1 H11.
elim H9 ⇒ R5.
elim ⇒ x.
elim ⇒ z H12.
assert (alpha · gamma # = (f0 # · (s0 # · t0)) · k0).
replace alpha with (f0 # · g0).
replace (gamma #) with (h0 # · k0).
rewrite -comp_assoc (@comp_assoc _ _ _ (f0 #)).
apply f_equal2.
apply f_equal.
apply H10.
by [].
apply Logic.eq_sym.
apply H5.
apply Logic.eq_sym.
apply H3.
assert (beta · delta # = (f1 # · (s1 # · t1)) · k1).
replace beta with (f1 # · g1).
replace (delta #) with (h1 # · k1).
rewrite -comp_assoc (@comp_assoc _ _ _ (f1 #)).
apply f_equal2.
apply f_equal.
apply H11.
by [].
apply Logic.eq_sym.
apply H6.
apply Logic.eq_sym.
apply H4.
assert (t0 · h0 = s0 · g0).
apply function_inc.
apply function_comp.
apply H10.
apply H5.
apply function_comp.

```

```
apply H10.
apply H3.
apply (@inc_trans _ _ _ (s0 · ((s0 # · t0) · h0))).
rewrite comp_assoc -comp_assoc.
apply comp_inc_compat_b_ab.
apply H10.
apply comp_inc_compat_ab_ab'.
replace (s0 # · t0) with (g0 · h0 #).
rewrite comp_assoc.
apply comp_inc_compat_ab_a.
apply H5.
apply H10.
assert (t1 · h1 = s1 · g1).
apply function_inc.
apply function_comp.
apply H11.
apply H6.
apply function_comp.
apply H11.
apply H4.
apply (@inc_trans _ _ _ (s1 · ((s1 # · t1) · h1))).
rewrite comp_assoc -comp_assoc.
apply comp_inc_compat_b_ab.
apply H11.
apply comp_inc_compat_ab_ab'.
replace (s1 # · t1) with (g1 · h1 #).
rewrite comp_assoc.
apply comp_inc_compat_ab_a.
apply H6.
apply H11.
remember ((x · (s0 · f0) #) (z · (t0 · k0) #)) as m0.
remember ((x · (s1 · f1) #) (z · (t1 · k1) #)) as m1.
assert (total_r m0).
rewrite Heqm0.
apply domain_corollary1.
apply H12.
apply H12.
replace (x # · z) with ((alpha · gamma #) (beta · delta #)).
apply (@inc_trans _ _ _ _ (@cap_l _ _ _)).
rewrite comp_inv H13 -comp_assoc comp_assoc.
apply inc_refl.
apply H12.
```

```
assert (total_r m1).
rewrite Heqm1.
apply domain_corollary1.
apply H12.
apply H12.
replace (x # • z) with ((alpha • gamma #) (beta • delta #)).
apply (@inc_trans _ _ _ _ (@cap_r _ _ _ _)).
rewrite comp_inv H14 -comp_assoc comp_assoc.
apply inc_refl.
apply H12.
remember (m0 • (s0 • g0)) as n0.
remember (m1 • (s1 • g1)) as n1.
assert (total_r n0).
rewrite Heqn0.
apply (total_comp H17).
apply total_comp.
apply H10.
apply H3.
assert (total_r n1).
rewrite Heqn1.
apply (total_comp H18).
apply total_comp.
apply H11.
apply H4.
assert (total_r ((n0 • fst_r Y Z #) (n1 • snd_r Y Z #))).
apply (domain_corollary1 H19 H20).
rewrite fst_snd_universal.
apply inc_alpha_universal.
assert ((x # • n0) alpha).
replace alpha with (f0 # • g0).
rewrite Heqn0 Heqm0.
apply (@inc_trans _ _ _ (((x # • x) • f0 #) • ((s0 # • s0) • g0))).
rewrite comp_assoc comp_assoc.
apply comp_inc_compat_ab_ab'.
rewrite -comp_assoc -comp_assoc -comp_assoc -comp_assoc.
apply comp_inc_compat_ab_a'b.
apply comp_inc_compat_ab_a'b.
rewrite comp_assoc -comp_inv.
apply cap_l.
apply comp_inc_compat.
apply comp_inc_compat_ab_b.
apply H12.
```

CHAPTER 12. LIBRARY SUM_PRODUCT

```

apply comp_inc_compat_ab_b.
apply H10.
apply Logic.eq_sym.
apply H3.
assert ((x # · n1) beta).
replace beta with (f1 # · g1).
rewrite Heqn1 Heqm1.
apply (@inc_trans _ _ _ (((x # · x) · f1 #) · ((s1 # · s1) · g1))).
rewrite comp_assoc comp_assoc.
apply comp_inc_compat_ab_ab'.
rewrite -comp_assoc -comp_assoc -comp_assoc -comp_assoc.
apply comp_inc_compat_ab_a'b.
apply comp_inc_compat_ab_a'b.
rewrite comp_assoc -comp_inv.
apply cap_l.
apply comp_inc_compat.
apply comp_inc_compat_ab_b.
apply H12.
apply comp_inc_compat_ab_b.
apply H11.
apply Logic.eq_sym.
apply H4.
assert ((n0 # · z) gamma #).
replace (gamma #) with (h0 # · k0).
rewrite Heqn0 Heqm0 -H15 comp_inv comp_inv inv_cap_distr.
apply (@inc_trans _ _ _ ((h0 # · (t0 # · t0)) · (k0 · (z # · z)))).
rewrite -comp_assoc -comp_assoc -comp_assoc.
apply comp_inc_compat_ab_a'b.
rewrite comp_assoc comp_assoc comp_assoc comp_assoc.
apply comp_inc_compat_ab_ab'.
apply comp_inc_compat_ab_ab'.
rewrite -comp_assoc (@comp_inv _ _ _ z) inv_invol.
apply cap_r.
apply comp_inc_compat.
apply comp_inc_compat_ab_a.
apply H10.
apply comp_inc_compat_ab_a.
apply H12.
apply Logic.eq_sym.
apply H5.
assert ((n1 # · z) delta #).
replace (delta #) with (h1 # · k1).

```

CHAPTER 12. LIBRARY SUM_PRODUCT

```

rewrite Heqn1 Heqm1 -H16 comp_inv comp_inv inv_cap_distr.
apply (@inc_trans _ _ _ ((h1 # · (t1 # · t1)) · (k1 · (z # · z)))).
rewrite -comp_assoc -comp_assoc -comp_assoc.
apply comp_inc_compat_ab_a'b.
rewrite comp_assoc comp_assoc comp_assoc comp_assoc.
apply comp_inc_compat_ab_ab'.
apply comp_inc_compat_ab_ab'.
rewrite -comp_assoc (@comp_inv _ _ _ z) inv_invol.
apply cap_r.
apply comp_inc_compat.
apply comp_inc_compat_ab_a.
apply H11.
apply comp_inc_compat_ab_a.
apply H12.
apply Logic.eq_sym.
apply H6.
replace ((alpha · gamma #) (beta · delta #)) with (x # · z).
apply (@inc_trans _ _ _ ((x # · (((n0 · fst_r Y Z #) (n1 · snd_r Y Z #)) · (((n0
· fst_r Y Z #) (n1 · snd_r Y Z #))) #)) · z)).
apply comp_inc_compat_ab_a'b.
apply (comp_inc_compat_a_ab H21).
rewrite -comp_assoc comp_assoc.
apply comp_inc_compat.
apply (@inc_trans _ _ _ _ (@comp_cap_distr_l _ _ _ _ _)).
apply cap_inc_compat.
rewrite -comp_assoc.
apply (comp_inc_compat_ab_a'b H22).
rewrite -comp_assoc.
apply (comp_inc_compat_ab_a'b H23).
rewrite inv_cap_distr comp_inv comp_inv inv_invol inv_invol.
apply (@inc_trans _ _ _ _ (@comp_cap_distr_r _ _ _ _ _)).
apply cap_inc_compat.
rewrite comp_assoc.
apply (comp_inc_compat_ab_ab' H24).
rewrite comp_assoc.
apply (comp_inc_compat_ab_ab' H25).
apply Logic.eq_sym.
apply H12.
apply (@inc_trans _ _ _ _ (@comp_cap_distr_l _ _ _ _ _)).
apply cap_inc_compat.
apply (@inc_trans _ _ _ _ (@comp_cap_distr_r _ _ _ _ _)).
apply (@inc_trans _ _ _ _ (@cap_l _ _ _ _ _)).

```

CHAPTER 12. LIBRARY SUM_PRODUCT

```

rewrite -comp_assoc (@comp_assoc _ _ _ alpha).
apply comp_inc_compat_ab_a'b.
apply comp_inc_compat_ab_a.
apply fst_function.
apply (@inc_trans _ _ _ (@comp_cap_distr_r _ _ _ _)).
apply (@inc_trans _ _ _ (@cap_r _ _ _ _)).
rewrite -comp_assoc (@comp_assoc _ _ _ beta).
apply comp_inc_compat_ab_a'b.
apply comp_inc_compat_ab_a.
apply snd_function.
Qed.

```

12.2.4 分配法則

Lemma 275 (prod_cap_distr_l) *Let $\alpha : A \rightarrow B$ and $\beta, \beta' : A \rightarrow C$. Then,*

$$\alpha \top (\beta \sqcap \beta') = (\alpha \top \beta) \sqcap (\alpha \top \beta').$$

Lemma prod_cap_distr_l $\{A B C : eqType\} \{alpha : Rel A B\} \{beta beta' : Rel A C\} :$
 $Rel_prod\ alpha\ (beta\ \beta') = Rel_prod\ alpha\ beta\ Rel_prod\ alpha\ beta'.$

Proof.

```

rewrite /Rel_prod.
rewrite -cap_assoc (@cap_comm _ _ (alpha • fst_r B C #)) -cap_assoc cap_idem
cap_assoc.
apply f_equal.
apply function_cap_distr_r.
apply snd_function.
Qed.

```

Lemma 276 (prod_cap_distr_r) *Let $\alpha, \alpha' : A \rightarrow B$ and $\beta : A \rightarrow C$. Then,*

$$(\alpha \sqcap \alpha') \top \beta = (\alpha \top \beta) \sqcap (\alpha' \top \beta).$$

Lemma prod_cap_distr_r $\{A B C : eqType\} \{alpha alpha' : Rel A B\} \{beta : Rel A C\} :$
 $Rel_prod\ (alpha\ alpha')\ beta = Rel_prod\ alpha\ beta\ Rel_prod\ alpha'\ beta.$

Proof.

```

rewrite /Rel_prod.
rewrite cap_assoc (@cap_comm _ _ (beta • snd_r B C #)) cap_assoc cap_idem -cap_assoc.
apply (@f_equal _ _ (fun x => @cap _ _ x (beta • snd_r B C #))).
apply function_cap_distr_r.
apply fst_function.
Qed.

```


CHAPTER 12. LIBRARY SUM_PRODUCT

Lemma 277 (prod_cup_distr_l) *Let $\alpha : A \rightarrow B$ and $\beta, \beta' : A \rightarrow C$. Then,*

$$\alpha \top (\beta \sqcup \beta') = (\alpha \top \beta) \sqcup (\alpha \top \beta').$$

Lemma *prod_cup_distr_l* {A B C : eqType} {alpha : Rel A B} {beta beta' : Rel A C}:
 $\text{Rel_prod } \alpha \text{ (beta beta')} = \text{Rel_prod } \alpha \text{ beta } \text{Rel_prod } \alpha \text{ beta'}$.

Proof.

by [rewrite -cap_cup_distr_l -comp_cup_distr_r].

Qed.

Lemma 278 (prod_cup_distr_r) *Let $\alpha, \alpha' : A \rightarrow B$ and $\beta : A \rightarrow C$. Then,*

$$(\alpha \sqcup \alpha') \top \beta = (\alpha \top \beta) \sqcup (\alpha' \top \beta).$$

Lemma *prod_cup_distr_r* {A B C : eqType} {alpha alpha' : Rel A B} {beta : Rel A C}:
 $\text{Rel_prod } (\alpha \sqcup \alpha') \text{ beta} = \text{Rel_prod } \alpha \text{ beta } \text{Rel_prod } \alpha' \text{ beta}$.

Proof.

by [rewrite -cap_cup_distr_r -comp_cup_distr_r].

Qed.

Lemma 279 (comp_prod_distr_l) *Let $\alpha : A \rightarrow B$, $\beta : B \rightarrow C$ and $\gamma : B \rightarrow D$. Then,*

$$\alpha \cdot (\beta \top \gamma) \sqsubseteq \alpha \cdot \beta \top \alpha \cdot \gamma.$$

Lemma *comp_prod_distr_l*

{A B C D : eqType} {alpha : Rel A B} {beta : Rel B C} {gamma : Rel B D}:
 $\alpha \cdot \text{Rel_prod } \beta \text{ gamma } \text{Rel_prod } (\alpha \cdot \beta) (\alpha \cdot \text{gamma})$.

Proof.

rewrite /Rel_prod.

rewrite comp_assoc comp_assoc.

apply comp_cap_distr_l.

Qed.

Lemma 280 (function_prod_distr_l) *Let $\alpha : A \rightarrow B$ be a function, $\beta : B \rightarrow C$ and $\gamma : B \rightarrow D$. Then,*

$$\alpha \cdot (\beta \top \gamma) = \alpha \cdot \beta \top \alpha \cdot \gamma.$$

Lemma *function_prod_distr_l*

{A B C D : eqType} {alpha : Rel A B} {beta : Rel B C} {gamma : Rel B D}:
 $\text{function_r } \alpha \rightarrow \alpha \cdot \text{Rel_prod } \beta \text{ gamma} = \text{Rel_prod } (\alpha \cdot \beta) (\alpha \cdot \text{gamma})$.

Proof.

CHAPTER 12. LIBRARY SUM_PRODUCT

`move ⇒ H.`
`rewrite /Rel_prod.`
`rewrite comp_assoc comp_assoc.`
`apply (function_cap_distr_l H).`
`Qed.`

Lemma 281 (comp_prod_universal) *Let $\alpha : A \rightarrow B$, $\beta : B \rightarrow C$ and $\gamma : D \rightarrow E$. Then,*

$$\alpha \cdot (\beta \top \nabla_{BD} \cdot \gamma) = \alpha \cdot \beta \top \nabla_{AD} \cdot \gamma.$$

Lemma comp_prod_universal

$\{A\ B\ C\ D\ E : \text{eqType}\} \{alpha : \text{Rel } A\ B\} \{beta : \text{Rel } B\ C\} \{gamma : \text{Rel } D\ E\} :$
 $alpha \cdot \text{Rel_prod } beta \ (\quad B\ D \cdot gamma) = \text{Rel_prod } (alpha \cdot beta) \ (\quad A\ D \cdot gamma).$

Proof.

`apply inc_antisym.`
`apply (@inc_trans _ _ _ _ (@comp_prod_distr_l _ _ _ _ _)).`
`apply prod_inc_compat_l.`
`rewrite -comp_assoc.`
`apply comp_inc_compat_ab_a'b.`
`apply inc_alpha_universal.`
`rewrite /Rel_prod.`
`rewrite comp_assoc.`
`apply (@inc_trans _ _ _ _ (@dedekind1 _ _ _ _ _)).`
`apply comp_inc_compat_ab_ab'.`
`apply cap_inc_compat_l.`
`rewrite comp_assoc comp_assoc -comp_assoc.`
`apply comp_inc_compat_ab_a'b.`
`apply inc_alpha_universal.`
`Qed.`

Lemma 282 (fst_cap_snd_distr) *Let $u, v : A \times B \rightarrow A \times B$ and $u, v \sqsubseteq id_{A \times B}$, $p : B \times C \rightarrow B$ and $q : B \times C \rightarrow C$ be projections. Then,*

$$p^\sharp \cdot (u \sqcap v) \cdot q = p^\sharp \cdot u \cdot q \sqcap p^\sharp \cdot v \cdot q.$$

Lemma fst_cap_snd_distr

$\{A\ B : \text{eqType}\} \{u\ v : \text{Rel } (\text{prod_eqType } A\ B) (\text{prod_eqType } A\ B)\} :$
 $u \quad Id (\text{prod_eqType } A\ B) \rightarrow v \quad Id (\text{prod_eqType } A\ B) \rightarrow$
 $\text{fst_r } A\ B \ \# \cdot (u \quad v) \cdot \text{snd_r } A\ B =$
 $((\text{fst_r } A\ B \ \# \cdot u) \cdot \text{snd_r } A\ B) \quad ((\text{fst_r } A\ B \ \# \cdot v) \cdot \text{snd_r } A\ B).$

Proof.

`move ⇒ H H0.`
`apply inc_antisym.`

CHAPTER 12. LIBRARY SUM_PRODUCT

```

apply (fun H' ⇒ @inc_trans _ _ _ _ H' (@comp_cap_distr_r _ _ _ _ _)).
apply comp_inc_compat_ab_a'b.
apply comp_cap_distr_l.
apply (@inc_trans _ _ _ _ (@dedekind1 _ _ _ _ _)).
rewrite -(dedekind_id3 H H0) -(@comp_assoc _ _ _ _ u) (@comp_assoc _ _ _ _ (fst_r A
B # • u) v).
apply comp_inc_compat_ab_ab'.
rewrite cap_comm comp_assoc -comp_assoc.
apply (@inc_trans _ _ _ _ (@dedekind2 _ _ _ _ _)).
apply comp_inc_compat_ab_b.
rewrite comp_inv comp_inv inv_invol -fst_snd_cap_id.
apply cap_inc_compat.
rewrite comp_assoc (dedekind_id1 H).
apply (comp_inc_compat_ab_b H).
rewrite -comp_assoc (dedekind_id1 H0).
apply (comp_inc_compat_ab_a H0).
Qed.

```

Bibliography

- [1] R. Affeldt and M. Hagiwara. Formalization of Shannon 's Theorems in SSReflect-Coq. In 3rd Conference on Interactive Theorem Proving, LNCS 7406, 233–249, 2012.