Open Induction

Mizuhito Ogawa Christian Sternagel*

April 20, 2020

Abstract

A proof of the open induction schema based on [1].

Contents

1 Binary Predicates Restricted to Elements of a Given Set

theory *Restricted-Predicates*
imports *Main*
begin

A subset C of A is a *chain* on A (w.r.t. P) iff for all pairs of elements of C, one is less than or equal to the other one.

abbreviation *chain-on* $P\ C\ A \equiv$ *pred-on.chain* $A\ P\ C$
lemmas *chain-on-def* $=$ *pred-on.chain-def*

lemma *chain-on-subset*:
$A \subseteq B \Longrightarrow$ *chain-on* $P\ C\ A \Longrightarrow$ *chain-on* $P\ C\ B$

*The research was partly funded by the Austrian Science Fund (FWF): J3202.

⟨*proof*⟩

lemma *chain-on-imp-subset*:
  chain-on P C A $\Longrightarrow$ $C \subseteq A$
⟨*proof*⟩

lemma *subchain-on*:
  assumes $C \subseteq D$ **and** *chain-on P D A*
  shows *chain-on P C A*
⟨*proof*⟩

definition *restrict-to* :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a\ set \Rightarrow ('a \Rightarrow 'a \Rightarrow bool)$ **where**
  restrict-to P A = $(\lambda x\ y.\ x \in A \wedge y \in A \wedge P\ x\ y)$

definition *reflp-on* :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a\ set \Rightarrow bool$ **where**
  reflp-on P A $\longleftrightarrow$ $(\forall\, a{\in}A.\ P\ a\ a)$

definition *transp-on* :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a\ set \Rightarrow bool$ **where**
  transp-on P A $\longleftrightarrow$ $(\forall\, x{\in}A.\ \forall\, y{\in}A.\ \forall\, z{\in}A.\ P\ x\ y \wedge P\ y\ z \longrightarrow P\ x\ z)$

definition *total-on* :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a\ set \Rightarrow bool$ **where**
  total-on P A $\longleftrightarrow$ $(\forall\, x{\in}A.\ \forall\, y{\in}A.\ x = y \vee P\ x\ y \vee P\ y\ x)$

abbreviation *strict* $P \equiv \lambda x\ y.\ P\ x\ y \wedge \neg\ (P\ y\ x)$

abbreviation *incomparable* $P \equiv \lambda x\ y.\ \neg\ P\ x\ y \wedge \neg\ P\ y\ x$

abbreviation *antichain-on* $P\ f\ A \equiv \forall\, (i{::}nat)\ j.\ f\ i \in A \wedge (i < j \longrightarrow$ *incomparable* $P\ (f\ i)\ (f\ j))$

lemma *strict-reflclp-conv* [*simp*]:
  strict $(P^{==})$ = *strict P* ⟨*proof*⟩

lemma *reflp-onI* [*Pure.intro*]:
  $(\bigwedge a.\ a \in A \Longrightarrow P\ a\ a) \Longrightarrow$ *reflp-on P A*
  ⟨*proof*⟩

lemma *transp-onI* [*Pure.intro*]:
  $(\bigwedge x\ y\ z.\ [\![x \in A;\ y \in A;\ z \in A;\ P\ x\ y;\ P\ y\ z]\!] \Longrightarrow P\ x\ z) \Longrightarrow$ *transp-on P A*
  ⟨*proof*⟩

lemma *total-onI* [*Pure.intro*]:
  $(\bigwedge x\ y.\ [\![x \in A;\ y \in A]\!] \Longrightarrow x = y \vee P\ x\ y \vee P\ y\ x) \Longrightarrow$ *total-on P A*
  ⟨*proof*⟩

lemma *reflp-on-reflclp-simp* [*simp*]:
  assumes *reflp-on P A* **and** $a \in A$ **and** $b \in A$
  shows $P^{==}\ a\ b = P\ a\ b$
  ⟨*proof*⟩

lemma *reflp-on-reflclp*:
reflp-on $(P^{==})$ A
⟨*proof*⟩

lemma *reflp-on-converse-simp* [*simp*]:
reflp-on P^{-1-1} $A \longleftrightarrow$ *reflp-on* P A
⟨*proof*⟩

lemma *transp-on-converse*:
transp-on P $A \Longrightarrow$ *transp-on* P^{-1-1} A
⟨*proof*⟩

lemma *transp-on-converse-simp* [*simp*]:
transp-on P^{-1-1} $A \longleftrightarrow$ *transp-on* P A
⟨*proof*⟩

lemma *transp-on-reflclp*:
transp-on P $A \Longrightarrow$ *transp-on* $P^{==}$ A
⟨*proof*⟩

lemma *transp-on-strict*:
transp-on P $A \Longrightarrow$ *transp-on* (*strict* P) A
⟨*proof*⟩

lemma *reflp-on-subset*:
$A \subseteq B \Longrightarrow$ *reflp-on* P $B \Longrightarrow$ *reflp-on* P A
⟨*proof*⟩

lemma *transp-on-subset*:
$A \subseteq B \Longrightarrow$ *transp-on* P $B \Longrightarrow$ *transp-on* P A
⟨*proof*⟩

definition *wfp-on* :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a\ set \Rightarrow bool$
where
wfp-on P $A \longleftrightarrow \neg\ (\exists f.\ \forall i.\ f\ i \in A \wedge P\ (f\ (Suc\ i))\ (f\ i))$

definition *inductive-on* :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a\ set \Rightarrow bool$ **where**
inductive-on P $A \longleftrightarrow (\forall Q.\ (\forall y{\in}A.\ (\forall x{\in}A.\ P\ x\ y \longrightarrow Q\ x) \longrightarrow Q\ y) \longrightarrow (\forall x{\in}A.\ Q\ x))$

lemma *inductive-onI* [*Pure.intro*]:
assumes $\bigwedge Q\ x.\ [\![x \in A;\ (\bigwedge y.\ [\![y \in A;\ \bigwedge x.\ [\![x \in A;\ P\ x\ y]\!] \Longrightarrow Q\ x]\!] \Longrightarrow Q\ y)]\!] \Longrightarrow Q\ x$
shows *inductive-on* P A
⟨*proof*⟩

If P is well-founded on A then every non-empty subset Q of A has a minimal element z w.r.t. P, i.e., all elements that are P-smaller than z are not in Q.

lemma *wfp-on-imp-minimal*:
assumes *wfp-on P A*
shows $\forall\, Q\ x.\ x \in Q \wedge Q \subseteq A \longrightarrow (\exists\, z{\in}Q.\ \forall\, y.\ P\ y\ z \longrightarrow y \notin Q)$
⟨*proof*⟩

lemma *minimal-imp-inductive-on*:
assumes $\forall\, Q\ x.\ x \in Q \wedge Q \subseteq A \longrightarrow (\exists\, z{\in}Q.\ \forall\, y.\ P\ y\ z \longrightarrow y \notin Q)$
shows *inductive-on P A*
⟨*proof*⟩

lemmas *wfp-on-imp-inductive-on* =
wfp-on-imp-minimal [*THEN minimal-imp-inductive-on*]

lemma *inductive-on-induct* [*consumes 2*, *case-names less*, *induct pred*: *inductive-on*]:
assumes *inductive-on P A* **and** $x \in A$
and $\bigwedge y.\ [\![\ y \in A;\ \bigwedge x.\ [\![\ x \in A;\ P\ x\ y\ ]\!] \Longrightarrow Q\ x\ ]\!] \Longrightarrow Q\ y$
shows *Q x*
⟨*proof*⟩

lemma *inductive-on-imp-wfp-on*:
assumes *inductive-on P A*
shows *wfp-on P A*
⟨*proof*⟩

definition *antisymp-on* :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a\ set \Rightarrow bool$ **where**
antisymp-on P A $\longleftrightarrow (\forall\, a{\in}A.\ \forall\, b{\in}A.\ P\ a\ b \wedge P\ b\ a \longrightarrow a = b)$

lemma *antisymp-onI* [*Pure.intro*]:
$(\bigwedge a\ b.\ [\![a \in A;\ b \in A;\ P\ a\ b;\ P\ b\ a]\!] \Longrightarrow a = b) \Longrightarrow$ *antisymp-on P A*
⟨*proof*⟩

lemma *antisymp-on-reflclp* [*simp*]:
antisymp-on $P^{==}$ *A* = *antisymp-on P A*
⟨*proof*⟩

definition *qo-on* :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a\ set \Rightarrow bool$ **where**
qo-on P A $\longleftrightarrow$ *reflp-on P A* $\wedge$ *transp-on P A*

definition *irreflp-on* :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a\ set \Rightarrow bool$ **where**
irreflp-on P A $\longleftrightarrow (\forall\, a{\in}A.\ \neg\ P\ a\ a)$

definition *po-on* :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a\ set \Rightarrow bool$ **where**
po-on P A $\longleftrightarrow$ (*irreflp-on P A* $\wedge$ *transp-on P A*)

lemma *po-onI* [*Pure.intro*]:
$[\![$*irreflp-on P A*; *transp-on P A*$]\!] \Longrightarrow$ *po-on P A*
⟨*proof*⟩

lemma *irreflp-onI* [*Pure.intro*]:

$(\bigwedge a.\ a \in A \Longrightarrow \neg\ P\ a\ a) \Longrightarrow$ *irreflp-on* $P\ A$
⟨*proof*⟩

lemma *irreflp-on-converse*:
irreflp-on $P\ A \Longrightarrow$ *irreflp-on* $P^{-1-1}\ A$
⟨*proof*⟩

lemma *irreflp-on-converse-simp* [*simp*]:
irreflp-on $P^{-1-1}\ A \longleftrightarrow$ *irreflp-on* $P\ A$
⟨*proof*⟩

lemma *po-on-converse-simp* [*simp*]:
po-on $P^{-1-1}\ A \longleftrightarrow$ *po-on* $P\ A$
⟨*proof*⟩

lemma *po-on-imp-qo-on*:
po-on $P\ A \Longrightarrow$ *qo-on* $(P^{==})\ A$
⟨*proof*⟩

lemma *po-on-imp-irreflp-on*:
po-on $P\ A \Longrightarrow$ *irreflp-on* $P\ A$
⟨*proof*⟩

lemma *po-on-imp-transp-on*:
po-on $P\ A \Longrightarrow$ *transp-on* $P\ A$
⟨*proof*⟩

lemma *irreflp-on-subset*:
assumes $A \subseteq B$ **and** *irreflp-on* $P\ B$
shows *irreflp-on* $P\ A$
⟨*proof*⟩

lemma *po-on-subset*:
assumes $A \subseteq B$ **and** *po-on* $P\ B$
shows *po-on* $P\ A$
⟨*proof*⟩

lemma *transp-on-irreflp-on-imp-antisymp-on*:
assumes *transp-on* $P\ A$ **and** *irreflp-on* $P\ A$
shows *antisymp-on* $(P^{==})\ A$
⟨*proof*⟩

lemma *po-on-imp-antisymp-on*:
assumes *po-on* $P\ A$
shows *antisymp-on* $P\ A$
⟨*proof*⟩

lemma *strict-reflclp* [*simp*]:
assumes $x \in A$ **and** $y \in A$

and *transp-on P A* **and** *irreflp-on P A*
shows *strict* $(P^{==})$ *x y* = *P x y*
⟨*proof*⟩

lemma *qo-on-imp-reflp-on*:
qo-on P A ⟹ *reflp-on P A*
⟨*proof*⟩

lemma *qo-on-imp-transp-on*:
qo-on P A ⟹ *transp-on P A*
⟨*proof*⟩

lemma *qo-on-subset*:
A ⊆ *B* ⟹ *qo-on P B* ⟹ *qo-on P A*
⟨*proof*⟩

Quasi-orders are instances of the *preorder* class.

lemma *qo-on-UNIV-conv*:
qo-on P UNIV ⟷ *class.preorder P* (*strict P*) (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *wfp-on-iff-inductive-on*:
wfp-on P A ⟷ *inductive-on P A*
⟨*proof*⟩

lemma *wfp-on-iff-minimal*:
wfp-on P A ⟷ (∀ *Q x*.
x ∈ *Q* ∧ *Q* ⊆ *A* ⟶
(∃ *z*∈*Q*. ∀ *y*. *P y z* ⟶ *y* ∉ *Q*))
⟨*proof*⟩

Every non-empty well-founded set A has a minimal element, i.e., an element that is not greater than any other element.

lemma *wfp-on-imp-has-min-elt*:
assumes *wfp-on P A* **and** *A* ≠ {}
shows ∃ *x*∈*A*. ∀ *y*∈*A*. ¬ *P y x*
⟨*proof*⟩

lemma *wfp-on-induct* [*consumes 2*, *case-names less*, *induct pred*: *wfp-on*]:
assumes *wfp-on P A* **and** *x* ∈ *A*
and ⋀*y*. ⟦ *y* ∈ *A*; ⋀*x*. ⟦ *x* ∈ *A*; *P x y* ⟧ ⟹ *Q x* ⟧ ⟹ *Q y*
shows *Q x*
⟨*proof*⟩

lemma *wfp-on-UNIV* [*simp*]:
wfp-on P UNIV ⟷ *wfP P*
⟨*proof*⟩

1.1 Measures on Sets (Instead of Full Types)

definition
inv-image-betw ::
$('b \Rightarrow 'b \Rightarrow bool) \Rightarrow ('a \Rightarrow 'b) \Rightarrow 'a\ set \Rightarrow 'b\ set \Rightarrow ('a \Rightarrow 'a \Rightarrow bool)$
where
inv-image-betw $P\ f\ A\ B = (\lambda x\ y.\ x \in A \wedge y \in A \wedge f\ x \in B \wedge f\ y \in B \wedge P\ (f\ x)\ (f\ y))$

definition
measure-on :: $('a \Rightarrow nat) \Rightarrow 'a\ set \Rightarrow 'a \Rightarrow 'a \Rightarrow bool$
where
measure-on $f\ A = inv\text{-}image\text{-}betw\ (<)\ f\ A\ UNIV$

lemma *in-inv-image-betw* [*simp*]:
inv-image-betw $P\ f\ A\ B\ x\ y \longleftrightarrow x \in A \wedge y \in A \wedge f\ x \in B \wedge f\ y \in B \wedge P\ (f\ x)\ (f\ y)$
⟨*proof*⟩

lemma *in-measure-on* [*simp*, *code-unfold*]:
measure-on $f\ A\ x\ y \longleftrightarrow x \in A \wedge y \in A \wedge f\ x < f\ y$
⟨*proof*⟩

lemma *wfp-on-inv-image-betw* [*simp*, *intro*!]:
assumes *wfp-on* $P\ B$
shows *wfp-on* $(inv\text{-}image\text{-}betw\ P\ f\ A\ B)\ A$ (**is** *wfp-on* $?P\ A$)
⟨*proof*⟩

lemma *wfp-less*:
wfp-on $(<)$ $(UNIV :: nat\ set)$
⟨*proof*⟩

lemma *wfp-on-measure-on* [*iff*]:
wfp-on $(measure\text{-}on\ f\ A)\ A$
⟨*proof*⟩

lemma *wfp-on-mono*:
$A \subseteq B \Longrightarrow (\bigwedge x\ y.\ x \in A \Longrightarrow y \in A \Longrightarrow P\ x\ y \Longrightarrow Q\ x\ y) \Longrightarrow wfp\text{-}on\ Q\ B \Longrightarrow wfp\text{-}on\ P\ A$
⟨*proof*⟩

lemma *wfp-on-subset*:
$A \subseteq B \Longrightarrow wfp\text{-}on\ P\ B \Longrightarrow wfp\text{-}on\ P\ A$
⟨*proof*⟩

lemma *restrict-to-iff* [*iff*]:
restrict-to $P\ A\ x\ y \longleftrightarrow x \in A \wedge y \in A \wedge P\ x\ y$
⟨*proof*⟩

lemma *wfp-on-restrict-to* [*simp*]:

wfp-on (*restrict-to* *P* *A*) *A* = *wfp-on* *P* *A*
⟨*proof*⟩

lemma *irreflp-on-strict* [*simp*, *intro*]:
irreflp-on (*strict* *P*) *A*
⟨*proof*⟩

lemma *transp-on-map'*:
assumes *transp-on* *Q* *B*
and *g* ‘ *A* ⊆ *B*
and *h* ‘ *A* ⊆ *B*
and ⋀*x*. *x* ∈ *A* ⟹ *Q*== (*h* *x*) (*g* *x*)
shows *transp-on* (λ*x* *y*. *Q* (*g* *x*) (*h* *y*)) *A*
⟨*proof*⟩

lemma *transp-on-map*:
assumes *transp-on* *Q* *B*
and *h* ‘ *A* ⊆ *B*
shows *transp-on* (λ*x* *y*. *Q* (*h* *x*) (*h* *y*)) *A*
⟨*proof*⟩

lemma *irreflp-on-map*:
assumes *irreflp-on* *Q* *B*
and *h* ‘ *A* ⊆ *B*
shows *irreflp-on* (λ*x* *y*. *Q* (*h* *x*) (*h* *y*)) *A*
⟨*proof*⟩

lemma *po-on-map*:
assumes *po-on* *Q* *B*
and *h* ‘ *A* ⊆ *B*
shows *po-on* (λ*x* *y*. *Q* (*h* *x*) (*h* *y*)) *A*
⟨*proof*⟩

lemma *chain-transp-on-less*:
assumes ∀ *i*. *f* *i* ∈ *A* ∧ *P* (*f* *i*) (*f* (*Suc* *i*)) **and** *transp-on* *P* *A* **and** *i* < *j*
shows *P* (*f* *i*) (*f* *j*)
⟨*proof*⟩

lemma *wfp-on-imp-irreflp-on*:
assumes *wfp-on* *P* *A*
shows *irreflp-on* *P* *A*
⟨*proof*⟩

inductive
accessible-on :: ('*a* ⇒ '*a* ⇒ *bool*) ⇒ '*a* *set* ⇒ '*a* ⇒ *bool*
for *P* **and** *A*
where
accessible-onI [*Pure.intro*]:
⟦*x* ∈ *A*; ⋀*y*. ⟦*y* ∈ *A*; *P* *y* *x*⟧ ⟹ *accessible-on* *P* *A* *y*⟧ ⟹ *accessible-on* *P* *A* *x*

lemma *accessible-on-imp-mem*:
assumes *accessible-on P A a*
shows $a \in A$
⟨*proof*⟩

lemma *accessible-on-induct* [*consumes 1*, *induct pred*: *accessible-on*]:
assumes ∗: *accessible-on P A a*
and *IH*: $\bigwedge x.\ [\![\textit{accessible-on } P\ A\ x;\ \bigwedge y.\ [\![y \in A;\ P\ y\ x]\!] \Longrightarrow Q\ y]\!] \Longrightarrow Q\ x$
shows *Q a*
⟨*proof*⟩

lemma *accessible-on-downward*:
accessible-on P A b $\Longrightarrow$ $a \in A$ $\Longrightarrow$ *P a b* $\Longrightarrow$ *accessible-on P A a*
⟨*proof*⟩

lemma *accessible-on-restrict-to-downwards*:
assumes $(\textit{restrict-to } P\ A)^{++}$ *a b* **and** *accessible-on P A b*
shows *accessible-on P A a*
⟨*proof*⟩

lemma *accessible-on-imp-inductive-on*:
assumes $\forall\, x{\in}A.$ *accessible-on P A x*
shows *inductive-on P A*
⟨*proof*⟩

lemmas *accessible-on-imp-wfp-on* = *accessible-on-imp-inductive-on* [*THEN inductive-on-imp-wfp-on*]

lemma *wfp-on-tranclp-imp-wfp-on*:
assumes *wfp-on* (P^{++}) *A*
shows *wfp-on P A*
⟨*proof*⟩

lemma *inductive-on-imp-accessible-on*:
assumes *inductive-on P A*
shows $\forall\, x{\in}A.$ *accessible-on P A x*
⟨*proof*⟩

lemma *inductive-on-accessible-on-conv*:
inductive-on P A $\longleftrightarrow$ $(\forall\, x{\in}A.$ *accessible-on P A x*$)$
⟨*proof*⟩

lemmas *wfp-on-imp-accessible-on* =
wfp-on-imp-inductive-on [*THEN inductive-on-imp-accessible-on*]

lemma *wfp-on-accessible-on-iff*:
wfp-on P A $\longleftrightarrow$ $(\forall\, x{\in}A.$ *accessible-on P A x*$)$
⟨*proof*⟩

lemma *accessible-on-tranclp*:
assumes *accessible-on P A x*
shows *accessible-on* ((*restrict-to P A*)$^{++}$) *A x*
(**is** *accessible-on ?P A x*)
⟨*proof*⟩

lemma *wfp-on-restrict-to-tranclp*:
assumes *wfp-on P A*
shows *wfp-on* ((*restrict-to P A*)$^{++}$) *A*
⟨*proof*⟩

lemma *wfp-on-restrict-to-tranclp'*:
assumes *wfp-on* (*restrict-to P A*)$^{++}$ *A*
shows *wfp-on P A*
⟨*proof*⟩

lemma *wfp-on-restrict-to-tranclp-wfp-on-conv*:
wfp-on (*restrict-to P A*)$^{++}$ *A* ⟷ *wfp-on P A*
⟨*proof*⟩

lemma *tranclp-idemp* [*simp*]:
$(P^{++})^{++} = P^{++}$ (**is** *?l* = *?r*)
⟨*proof*⟩

lemma *stepfun-imp-tranclp*:
assumes *f 0* = *x* **and** *f* (*Suc n*) = *z*
and ∀ *i*≤*n*. *P* (*f i*) (*f* (*Suc i*))
shows P^{++} *x z*
⟨*proof*⟩

lemma *tranclp-imp-stepfun*:
assumes P^{++} *x z*
shows ∃ *f n*. *f 0* = *x* ∧ *f* (*Suc n*) = *z* ∧ (∀ *i*≤*n*. *P* (*f i*) (*f* (*Suc i*)))
(**is** ∃ *f n*. *?P x z f n*)
⟨*proof*⟩

lemma *tranclp-stepfun-conv*:
P^{++} *x y* ⟷ (∃ *f n*. *f 0* = *x* ∧ *f* (*Suc n*) = *y* ∧ (∀ *i*≤*n*. *P* (*f i*) (*f* (*Suc i*))))
⟨*proof*⟩

1.2 Facts About Predecessor Sets

lemma *qo-on-predecessor-subset-conv'*:
assumes *qo-on P A* **and** *B* ⊆ *A* **and** *C* ⊆ *A*
shows {*x*∈*A*. ∃ *y*∈*B*. *P x y*} ⊆ {*x*∈*A*. ∃ *y*∈*C*. *P x y*} ⟷ (∀ *x*∈*B*. ∃ *y*∈*C*. *P x y*)
⟨*proof*⟩

lemma *qo-on-predecessor-subset-conv*:
⟦*qo-on P A*; $x \in A$; $y \in A$⟧ $\Longrightarrow$ $\{z{\in}A.\ P\ z\ x\} \subseteq \{z{\in}A.\ P\ z\ y\} \longleftrightarrow P\ x\ y$
⟨*proof*⟩

lemma *po-on-predecessors-eq-conv*:
assumes *po-on P A* **and** $x \in A$ **and** $y \in A$
shows $\{z{\in}A.\ P^{==}\ z\ x\} = \{z{\in}A.\ P^{==}\ z\ y\} \longleftrightarrow x = y$
⟨*proof*⟩

lemma *restrict-to-rtranclp*:
assumes *transp-on P A*
and $x \in A$ **and** $y \in A$
shows (*restrict-to P A*)** $x\ y \longleftrightarrow P^{==}\ x\ y$
⟨*proof*⟩

lemma *reflp-on-restrict-to-rtranclp*:
assumes *reflp-on P A* **and** *transp-on P A*
and $x \in A$ **and** $y \in A$
shows (*restrict-to P A*)** $x\ y \longleftrightarrow P\ x\ y$
⟨*proof*⟩

end

2 Open Induction

theory *Open-Induction*
imports *Restricted-Predicates*
begin

2.1 (Greatest) Lower Bounds and Chains

A set B has the *lower bound* x iff x is less than or equal to every element of B.

definition *lb P B x* $\longleftrightarrow$ $(\forall\, y{\in}B.\ P^{==}\ x\ y)$

lemma *lbI* [*Pure.intro*]:
$(\bigwedge y.\ y \in B \Longrightarrow P^{==}\ x\ y) \Longrightarrow$ *lb P B x*
⟨*proof*⟩

A set B has the *greatest lower bound* x iff x is a lower bound of B *and* less than or equal to every other lower bound of B.

definition *glb P B x* $\longleftrightarrow$ *lb P B x* $\wedge$ $(\forall\, y.\ lb\ P\ B\ y \longrightarrow P^{==}\ y\ x)$

lemma *glbI* [*Pure.intro*]:
lb P B x $\Longrightarrow$ $(\bigwedge y.\ lb\ P\ B\ y \Longrightarrow P^{==}\ y\ x) \Longrightarrow$ *glb P B x*
⟨*proof*⟩

Antisymmetric relations have unique glbs.

lemma *glb-unique*:
antisymp-on P A $\Longrightarrow$ $x \in A \Longrightarrow y \in A \Longrightarrow$ *glb P B x* $\Longrightarrow$ *glb P B y* $\Longrightarrow x = y$
⟨*proof*⟩

context *pred-on*
begin

lemma *chain-glb*:
assumes *transp-on* $(\sqsubset)$ *A*
shows *chain C* $\Longrightarrow$ *glb* $(\sqsubset)$ *C x* $\Longrightarrow x \in A \Longrightarrow y \in A \Longrightarrow y \sqsubset x \Longrightarrow$ *chain* $(\{y\} \cup C)$
⟨*proof*⟩

2.2 Open Properties

definition *open Q* $\longleftrightarrow$ $(\forall\, C.$ *chain C* $\wedge$ $C \neq \{\} \wedge (\exists\, x{\in}A.$ *glb* $(\sqsubset)$ *C x* $\wedge$ *Q x*$) \longrightarrow (\exists\, y{\in}C.$ *Q y*$))$

lemma *openI* [*Pure.intro*]:
$(\bigwedge C.$ *chain C* $\Longrightarrow$ $C \neq \{\} \Longrightarrow \exists\, x{\in}A.$ *glb* $(\sqsubset)$ *C x* $\wedge$ *Q x* $\Longrightarrow \exists\, y{\in}C.$ *Q y*$) \Longrightarrow$ *open Q*
⟨*proof*⟩

lemma *open-glb*:
$[\![$*chain C*; $C \neq \{\}$; *open Q*; $\forall\, x{\in}C. \neg$ *Q x*; $x \in A$; *glb* $(\sqsubset)$ *C x*$]\!] \Longrightarrow \neg$ *Q x*
⟨*proof*⟩

2.3 Downward Completeness

A relation $\sqsubset$ is *downward-complete* iff every non-empty $\sqsubset$-chain has a greatest lower bound.

definition *downward-complete* $\longleftrightarrow$ $(\forall\, C.$ *chain C* $\wedge$ $C \neq \{\} \longrightarrow (\exists\, x{\in}A.$ *glb* $(\sqsubset)$ *C x*$))$

lemma *downward-completeI* [*Pure.intro*]:
assumes $\bigwedge C.$ *chain C* $\Longrightarrow$ $C \neq \{\} \Longrightarrow \exists\, x{\in}A.$ *glb* $(\sqsubset)$ *C x*
shows *downward-complete*
⟨*proof*⟩

end

abbreviation *open-on P Q A* $\equiv$ *pred-on.open A P Q*
abbreviation *dc-on P A* $\equiv$ *pred-on.downward-complete A P*
lemmas *open-on-def* = *pred-on.open-def*
and *dc-on-def* = *pred-on.downward-complete-def*

lemma *dc-onI* [*Pure.intro*]:
assumes $\bigwedge C.$ *chain-on P C A* $\Longrightarrow$ $C \neq \{\} \Longrightarrow \exists\, x{\in}A.$ *glb P C x*
shows *dc-on P A*

⟨*proof*⟩

lemma *open-onI* [*Pure.intro*]:
(⋀*C*. *chain-on P C A* ⟹ *C* ≠ {} ⟹ ∃ *x*∈*A*. *glb P C x* ∧ *Q x* ⟹ ∃ *y*∈*C*. *Q y*) ⟹ *open-on P Q A*
⟨*proof*⟩

lemma *chain-on-reflclp*:
chain-on $P^{==}$ *A C* ⟷ *chain-on P A C*
⟨*proof*⟩

lemma *lb-reflclp*:
lb $P^{==}$ *B x* ⟷ *lb P B x*
⟨*proof*⟩

lemma *glb-reflclp*:
glb $P^{==}$ *B x* ⟷ *glb P B x*
⟨*proof*⟩

lemma *dc-on-reflclp*:
dc-on $P^{==}$ *A* ⟷ *dc-on P A*
⟨*proof*⟩

2.4 The Open Induction Principle

lemma *open-induct-on* [*consumes 4*, *case-names less*]:
assumes *qo*: *qo-on P A* **and** *dc-on P A* **and** *open-on P Q A*
and *x* ∈ *A*
and *ind*: ⋀*x*. ⟦*x* ∈ *A*; ⋀*y*. ⟦*y* ∈ *A*; *strict P y x*⟧ ⟹ *Q y*⟧ ⟹ *Q x*
shows *Q x*
⟨*proof*⟩

2.5 Open Induction on Universal Domains

Open induction on quasi-orders (i.e., *preorder*).

lemma (**in** *preorder*) *dc-open-induct* [*consumes 2*, *case-names less*]:
assumes *dc-on* (≤) *UNIV*
and *open-on* (≤) *Q UNIV*
and ⋀*x*. (⋀*y*. *y* < *x* ⟹ *Q y*) ⟹ *Q x*
shows *Q x*
⟨*proof*⟩

2.6 Type Class of Downward Complete Orders

class *dcorder* = *preorder* +
assumes *dc-on-UNIV*: *dc-on* (≤) *UNIV*
begin

Open induction on downward-complete orders.

lemmas *open-induct* [*consumes 1*, *case-names less*] = *dc-open-induct* [*OF dc-on-UNIV*]

end

end

References

[1] J.-C. Raoult. Proving open properties by induction. *Information Processing Letters*, 29(1):19–23, 1988. doi:10.1016/0020-0190(88)90126-3.