

Open Induction

Mizuhito Ogawa Christian Sternagel*

May 14, 2024

Abstract

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

Contents

1	Binary Predicates Restricted to Elements of a Given Set	1
1.1	Measures on Sets (Instead of Full Types)	5
1.2	Facts About Predecessor Sets	9
2	Open Induction	9
2.1	(Greatest) Lower Bounds and Chains	10
2.2	Open Properties	10
2.3	Downward Completeness	11
2.4	The Open Induction Principle	11
2.5	Open Induction on Universal Domains	12
2.6	Type Class of Downward Complete Orders	12

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 $\text{chain-on } P \ C \ A \equiv \text{pred-on.chain } A \ P \ C$

lemmas $\text{chain-on-def} = \text{pred-on.chain-def}$

lemma *chain-on-subset*:

$A \subseteq B \implies \text{chain-on } P \ C \ A \implies \text{chain-on } P \ C \ B$

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

<proof>

lemma *chain-on-imp-subset*:
chain-on P C A $\implies C \subseteq A$
<proof>

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

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

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* $\equiv \forall (i::\text{nat}) j. f i \in A \wedge (i < j \longrightarrow \text{incomparable } P (f i) (f j))$

lemma *strict-reflclp-conv* [*simp*]:
strict (P⁼⁼) = *strict P* *<proof>*

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

lemmas *reflp-on-converse-simp* = *reflp-on-conversp*
lemmas *irreflp-on-converse-simp* = *irreflp-on-converse*
lemmas *transp-on-converse-simp* = *transp-on-conversep*

lemma *transp-on-strict*:
transp-on A P $\implies \text{transp-on } A (\text{strict } P)$
<proof>

definition *wfp-on* :: $('a \Rightarrow 'a \Rightarrow \text{bool}) \Rightarrow 'a \text{ set} \Rightarrow \text{bool}$
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* :: $('a \Rightarrow 'a \Rightarrow \text{bool}) \Rightarrow 'a \text{ set} \Rightarrow \text{bool}$ **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 $\bigwedge Q x. \llbracket x \in A; (\bigwedge y. \llbracket y \in A; \bigwedge x. \llbracket x \in A; P x y \rrbracket \implies Q x \rrbracket \implies Q y) \rrbracket$
 $\implies Q x$
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 $\bigwedge y. \llbracket y \in A; \bigwedge x. \llbracket x \in A; P x y \rrbracket \Longrightarrow Q x \rrbracket \Longrightarrow Q y$

shows $Q x$

<proof>

lemma *inductive-on-imp-wfp-on*:

assumes *inductive-on* $P A$

shows *wfp-on* $P A$

<proof>

definition *qo-on* :: $('a \Rightarrow 'a \Rightarrow \text{bool}) \Rightarrow 'a \text{ set} \Rightarrow \text{bool}$ **where**

qo-on $P A \longleftrightarrow \text{reflp-on } A P \wedge \text{transp-on } A P$

definition *po-on* :: $('a \Rightarrow 'a \Rightarrow \text{bool}) \Rightarrow 'a \text{ set} \Rightarrow \text{bool}$ **where**

po-on $P A \longleftrightarrow (\text{irreflp-on } A P \wedge \text{transp-on } A P)$

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

$\llbracket \text{irreflp-on } A P; \text{transp-on } A P \rrbracket \Longrightarrow \text{po-on } P A$

<proof>

lemma *po-on-converse-simp* [*simp*]:

po-on $P^{-1-1} A \longleftrightarrow \text{po-on } P A$

<proof>

lemma *po-on-imp-qo-on*:

po-on $P A \Longrightarrow \text{qo-on } (P^{==}) A$

<proof>

lemma *po-on-imp-irreflp-on*:

po-on $P A \Longrightarrow \text{irreflp-on } A P$

<proof>

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

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

lemma *strict-reflclp* [*simp*]:
 assumes $x \in A$ and $y \in A$
 and *transp-on* $A\ P$ and *irreflp-on* $A\ P$
 shows *strict* $(P^{==})\ x\ y = P\ x\ y$
 ⟨proof⟩

lemma *qo-on-imp-reflp-on*:
 $qo-on\ P\ A \implies reflp-on\ A\ P$
 ⟨proof⟩

lemma *qo-on-imp-transp-on*:
 $qo-on\ P\ A \implies transp-on\ A\ P$
 ⟨proof⟩

lemma *qo-on-subset*:
 $A \subseteq B \implies qo-on\ P\ B \implies qo-on\ P\ A$
 ⟨proof⟩

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

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

lemma *wfp-on-iff-inductive-on*:
 $wfp-on\ P\ A \longleftrightarrow inductive-on\ P\ A$
 ⟨proof⟩

lemma *wfp-on-iff-minimal*:
 $wfp-on\ P\ A \longleftrightarrow (\forall Q\ x.$

$x \in Q \wedge Q \subseteq A \longrightarrow$
 $(\exists z \in Q. \forall y. P y z \longrightarrow y \notin Q)$
 <proof>

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 $\exists x \in A. \forall y \in A. \neg P y x$
 <proof>

lemma *wfp-on-induct* [*consumes 2, case-names less, induct pred: wfp-on*]:
assumes *wfp-on P A and x ∈ A*
and $\bigwedge y. \llbracket y \in A; \bigwedge x. \llbracket x \in A; P x y \rrbracket \Longrightarrow Q x \rrbracket \Longrightarrow Q y$
shows $Q x$
 <proof>

lemma *wfp-on-UNIV* [*simp*]:
 $wfp-on P UNIV \longleftrightarrow wfP P$
 <proof>

1.1 Measures on Sets (Instead of Full Types)

definition

inv-image-betw ::
 $('b \Rightarrow 'a \Rightarrow bool) \Rightarrow ('a \Rightarrow 'b) \Rightarrow 'a \text{ set} \Rightarrow 'b \text{ set} \Rightarrow ('a \Rightarrow 'a \Rightarrow bool)$

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 :: $('a \Rightarrow nat) \Rightarrow 'a \text{ set} \Rightarrow 'a \Rightarrow 'a \Rightarrow bool$

where

$measure-on f A = inv-image-betw (<) f A UNIV$

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-image-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-on* *f* *A*) *A*

<proof>

lemma *wfp-on-mono*:

$A \subseteq B \implies (\bigwedge x y. x \in A \implies y \in A \implies P x y \implies Q x y) \implies \text{wfp-on } Q B \implies \text{wfp-on } P A$

<proof>

lemma *wfp-on-subset*:

$A \subseteq B \implies \text{wfp-on } P B \implies \text{wfp-on } P A$

<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 *A* (*strict* *P*)

<proof>

lemma *transp-on-map'*:

assumes *transp-on* *B* *Q*

and $g \text{ ' } A \subseteq B$

and $h \text{ ' } A \subseteq B$

and $\bigwedge x. x \in A \implies Q = (h x) (g x)$

shows *transp-on* *A* ($\lambda x y. Q (g x) (h y)$)

<proof>

lemma *transp-on-map*:

assumes *transp-on* *B* *Q*

and $h \text{ ' } A \subseteq B$

shows *transp-on* *A* ($\lambda x y. Q (h x) (h y)$)

<proof>

lemma *irreflp-on-map*:

assumes *irreflp-on* *B* *Q*

and $h \text{ ' } A \subseteq B$

shows *irreflp-on* *A* ($\lambda x y. Q (h x) (h y)$)

<proof>

lemma *po-on-map*:

assumes *po-on* $Q B$

and $h : A \subseteq B$

shows *po-on* $(\lambda x y. Q (h x) (h y)) A$

<proof>

lemma *chain-transp-on-less*:

assumes $\forall i. f i \in A \wedge P (f i) (f (Suc i))$ **and** *transp-on* $A P$ **and** $i < j$

shows $P (f i) (f j)$

<proof>

lemma *wfp-on-imp-irreflp-on*:

assumes *wfp-on* $P A$

shows *irreflp-on* $A P$

<proof>

inductive

accessible-on :: $('a \Rightarrow 'a \Rightarrow \text{bool}) \Rightarrow 'a \text{ set} \Rightarrow 'a \Rightarrow \text{bool}$

for P **and** A

where

accessible-onI [*Pure.intro*]:

$\llbracket x \in A; \bigwedge y. \llbracket y \in A; P y x \rrbracket \implies \text{accessible-on } P A y \rrbracket \implies \text{accessible-on } P A x$

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*: $\bigwedge x. \llbracket \text{accessible-on } P A x; \bigwedge y. \llbracket y \in A; P y x \rrbracket \implies Q y \rrbracket \implies Q x$

shows $Q a$

<proof>

lemma *accessible-on-downward*:

accessible-on $P A b \implies a \in A \implies P a b \implies \text{accessible-on } P A a$

<proof>

lemma *accessible-on-restrict-to-downwards*:

assumes $(\text{restrict-to } P A)^{++} a b$ **and** *accessible-on* $P A b$

shows *accessible-on* $P A a$

<proof>

lemma *accessible-on-imp-inductive-on*:

assumes $\forall x \in A. \text{accessible-on } P A x$

shows *inductive-on* $P A$

<proof>

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

tive-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 $\forall x \in A. \text{accessible-on } P A x$

<proof>

lemma *inductive-on-accessible-on-conv:*

inductive-on $P A \longleftrightarrow (\forall x \in A. \text{accessible-on } P A x)$

<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 $P A \longleftrightarrow (\forall x \in A. \text{accessible-on } P A x)$

<proof>

lemma *accessible-on-tranclp:*

assumes *accessible-on* $P A x$

shows *accessible-on* $((\text{restrict-to } P A)^{++}) A x$

(**is** *accessible-on* $?P A x$)

<proof>

lemma *wfp-on-restrict-to-tranclp:*

assumes *wfp-on* $P A$

shows *wfp-on* $((\text{restrict-to } P A)^{++}) A$

<proof>

lemma *wfp-on-restrict-to-tranclp':*

assumes *wfp-on* $(\text{restrict-to } P A)^{++} A$

shows *wfp-on* $P A$

<proof>

lemma *wfp-on-restrict-to-tranclp-wfp-on-conv:*

wfp-on $(\text{restrict-to } P A)^{++} A \longleftrightarrow \text{wfp-on } P A$

<proof>

lemma *tranclp-idemp [simp]:*

$(P^{++})^{++} = P^{++}$ (**is** $?l = ?r$)

<proof>

lemma *stepfun-imp-tranclp:*

assumes $f\ 0 = x$ **and** $f\ (\text{Suc } n) = z$
and $\forall i \leq n. P\ (f\ i)\ (f\ (\text{Suc } i))$
shows $P^{++}\ x\ z$
 $\langle \text{proof} \rangle$

lemma *tranclp-imp-stepfun*:

assumes $P^{++}\ x\ z$
shows $\exists f\ n. f\ 0 = x \wedge f\ (\text{Suc } n) = z \wedge (\forall i \leq n. P\ (f\ i)\ (f\ (\text{Suc } i)))$
(is $\exists f\ n. ?P\ x\ z\ f\ n$ **)**
 $\langle \text{proof} \rangle$

lemma *tranclp-stepfun-conv*:

$P^{++}\ x\ y \longleftrightarrow (\exists f\ n. f\ 0 = x \wedge f\ (\text{Suc } n) = y \wedge (\forall i \leq n. P\ (f\ i)\ (f\ (\text{Suc } i))))$
 $\langle \text{proof} \rangle$

1.2 Facts About Predecessor Sets

lemma *qo-on-predecessor-subset-conv'*:

assumes *qo-on* $P\ A$ **and** $B \subseteq A$ **and** $C \subseteq A$
shows $\{x \in A. \exists y \in B. P\ x\ y\} \subseteq \{x \in A. \exists y \in C. P\ x\ y\} \longleftrightarrow (\forall x \in B. \exists y \in C. P\ x\ y)$
 $\langle \text{proof} \rangle$

lemma *qo-on-predecessor-subset-conv*:

$\llbracket \text{qo-on } P\ A; x \in A; y \in A \rrbracket \implies \{z \in A. P\ z\ x\} \subseteq \{z \in A. P\ z\ y\} \longleftrightarrow P\ x\ y$
 $\langle \text{proof} \rangle$

lemma *po-on-predecessors-eq-conv*:

assumes *po-on* $P\ A$ **and** $x \in A$ **and** $y \in A$
shows $\{z \in A. P^{==}\ z\ x\} = \{z \in A. P^{==}\ z\ y\} \longleftrightarrow x = y$
 $\langle \text{proof} \rangle$

lemma *restrict-to-rtranclp*:

assumes *transp-on* $A\ P$
and $x \in A$ **and** $y \in A$
shows $(\text{restrict-to } P\ A)^{**}\ x\ y \longleftrightarrow P^{==}\ x\ y$
 $\langle \text{proof} \rangle$

lemma *reflp-on-restrict-to-rtranclp*:

assumes *reflp-on* $A\ P$ **and** *transp-on* $A\ P$
and $x \in A$ **and** $y \in A$
shows $(\text{restrict-to } P\ A)^{**}\ x\ y \longleftrightarrow P\ x\ y$
 $\langle \text{proof} \rangle$

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*]:

$(\bigwedge y. y \in B \implies P = x\ y) \implies lb\ P\ B\ x$
<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*]:

$lb\ P\ B\ x \implies (\bigwedge y. lb\ P\ B\ y \implies P = y\ x) \implies glb\ P\ B\ x$
<proof>

Antisymmetric relations have unique glbs.

lemma glb -unique:

$antisym\text{-}on\ A\ P \implies x \in A \implies y \in A \implies glb\ P\ B\ x \implies glb\ P\ B\ y \implies x = y$
<proof>

context $pred$ -on

begin

lemma $chain$ -glb:

assumes $transp$ -on A (\sqsubset)
shows $chain\ C \implies glb\ (\sqsubset)\ C\ x \implies x \in A \implies y \in A \implies y \sqsubset x \implies chain\ (\{y\} \cup C)$
<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*]:

$(\bigwedge C. chain\ C \implies C \neq \{\} \implies \exists x \in A. glb\ (\sqsubset)\ C\ x \wedge Q\ x \implies \exists y \in C. Q\ y) \implies open\ Q$
<proof>

lemma $open$ -glb:

$[[chain\ C; C \neq \{\}]; open\ Q; \forall x \in C. \neg Q\ x; x \in A; glb\ (\sqsubset)\ C\ x] \implies \neg Q\ x$
<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* $\longleftrightarrow (\forall C. \text{chain } C \wedge C \neq \{\} \longrightarrow (\exists x \in A. \text{glb } (\sqsubset) C x))$

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

end

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

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

lemma *open-onI* [*Pure.intro*]:
 $(\bigwedge C. \text{chain-on } P C A \Longrightarrow C \neq \{\} \Longrightarrow \exists x \in A. \text{glb } P C x \wedge Q x \Longrightarrow \exists y \in C. Q y) \Longrightarrow \text{open-on } P Q A$
 $\langle \text{proof} \rangle$

lemma *chain-on-reflclp*:
 $\text{chain-on } P^{==} A C \longleftrightarrow \text{chain-on } P A C$
 $\langle \text{proof} \rangle$

lemma *lb-reflclp*:
 $\text{lb } P^{==} B x \longleftrightarrow \text{lb } P B x$
 $\langle \text{proof} \rangle$

lemma *glb-reflclp*:
 $\text{glb } P^{==} B x \longleftrightarrow \text{glb } P B x$
 $\langle \text{proof} \rangle$

lemma *dc-on-reflclp*:
 $\text{dc-on } P^{==} A \longleftrightarrow \text{dc-on } P A$
 $\langle \text{proof} \rangle$

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  $x \in A$ 
and  $ind: \bigwedge x. [x \in A; \bigwedge y. [y \in A; strict\ P\ y\ x]] \implies Q\ y] \implies Q\ x$ 
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 ( $\leq$ ) UNIV
and open-on ( $\leq$ )  $Q\ UNIV$ 
and  $\bigwedge x. (\bigwedge y. y < x \implies Q\ y) \implies Q\ x$ 
shows  $Q\ x$ 
⟨proof⟩

```

2.6 Type Class of Downward Complete Orders

```

class dcorder = preorder +
assumes dc-on-UNIV: dc-on ( $\leq$ ) 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.