Well-Quasi-Orders

Christian Sternagel*

March 17, 2025

Abstract

Based on Isabelle/HOL's type class for preorders, we introduce a type class for well-quasi-orders (wqo) which is characterized by the absence of "bad" sequences (our proofs are along the lines of the proof of Nash-Williams [1], from which we also borrow terminology). Our main results are instantiations for the product type, the list type, and a type of finite trees, which (almost) directly follow from our proofs of (1) Dickson's Lemma, (2) Higman's Lemma, and (3) Kruskal's Tree Theorem. More concretely:

1. If the sets A and B are wqo then their Cartesian product is wqo.
2. If the set A is wqo then the set of finite lists over A is wqo.
3. If the set A is wqo then the set of finite trees over A is wqo.

Contents

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

1 Infinite Sequences

Some useful constructions on and facts about infinite sequences.

theory *Infinite-Sequences*
imports *Main*
begin

The set of all infinite sequences over elements from A.

definition $SEQ\ A = \{f{::}nat \Rightarrow 'a.\ \forall i.\ f\ i \in A\}$

lemma *SEQ-iff* [*iff*]:
$f \in SEQ\ A \longleftrightarrow (\forall i.\ f\ i \in A)$
⟨*proof*⟩

The i-th "column" of a set B of infinite sequences.

definition *ith* $B\ i = \{f\ i \mid f.\ f \in B\}$

lemma *ithI* [*intro*]:
$f \in B \Longrightarrow f\ i = x \Longrightarrow x \in ith\ B\ i$
⟨*proof*⟩

lemma *ithE* [*elim*]:
$[\![x \in ith\ B\ i;\ \bigwedge f.\ [\![f \in B;\ f\ i = x]\!] \Longrightarrow Q]\!] \Longrightarrow Q$
⟨*proof*⟩

lemma *ith-conv*:
$x \in ith\ B\ i \longleftrightarrow (\exists f \in B.\ x = f\ i)$
⟨*proof*⟩

The restriction of a set B of sequences to sequences that are equal to a given sequence f up to position i.

definition *eq-upto* :: $(nat \Rightarrow 'a)\ set \Rightarrow (nat \Rightarrow 'a) \Rightarrow nat \Rightarrow (nat \Rightarrow 'a)\ set$
where
$eq\text{-}upto\ B\ f\ i = \{g \in B.\ \forall j < i.\ f\ j = g\ j\}$

lemma *eq-uptoI* [*intro*]:
$[\![g \in B;\ \bigwedge j.\ j < i \Longrightarrow f\ j = g\ j]\!] \Longrightarrow g \in eq\text{-}upto\ B\ f\ i$
⟨*proof*⟩

lemma *eq-uptoE* [*elim*]:
$[\![g \in eq\text{-}upto\ B\ f\ i;\ [\![g \in B;\ \bigwedge j.\ j < i \Longrightarrow f\ j = g\ j]\!] \Longrightarrow Q]\!] \Longrightarrow Q$
⟨*proof*⟩

lemma *eq-upto-Suc*:
$[\![g \in eq\text{-}upto\ B\ f\ i;\ g\ i = f\ i]\!] \Longrightarrow g \in eq\text{-}upto\ B\ f\ (Suc\ i)$
⟨*proof*⟩

lemma *eq-upto-0* [*simp*]:
$eq\text{-}upto\ B\ f\ 0 = B$
⟨*proof*⟩

lemma *eq-upto-cong* [*fundef-cong*]:
assumes $\bigwedge j.\ j < i \Longrightarrow f\ j = g\ j$ **and** $B = C$
shows $eq\text{-}upto\ B\ f\ i = eq\text{-}upto\ C\ g\ i$
⟨*proof*⟩

1.1 Lexicographic Order on Infinite Sequences

definition $LEX\ P\ f\ g \longleftrightarrow (\exists i{::}nat.\ P\ (f\ i)\ (g\ i) \wedge (\forall j{<}i.\ f\ j = g\ j))$
abbreviation $LEXEQ\ P \equiv (LEX\ P)^{==}$

lemma *LEX-imp-not-LEX*:
assumes $LEX\ P\ f\ g$

and [*dest*]: $\bigwedge x\ y\ z.\ P\ x\ y \Longrightarrow P\ y\ z \Longrightarrow P\ x\ z$
and [*simp*]: $\bigwedge x.\ \neg\ P\ x\ x$
shows $\neg$ *LEX P g f*
⟨*proof*⟩

lemma *LEX-cases*:
assumes *LEX P f g*
obtains (*eq*) $f = g$ | (*neq*) k **where** $\forall\, i{<}k.\ f\ i = g\ i$ **and** $P\ (f\ k)\ (g\ k)$
⟨*proof*⟩

lemma *LEX-imp-less*:
assumes $\forall\, x{\in}A.\ \neg\ P\ x\ x$ **and** $f \in SEQ\ A \vee g \in SEQ\ A$
and *LEX P f g* **and** $\forall\, i{<}k.\ f\ i = g\ i$ **and** $f\ k \neq g\ k$
shows $P\ (f\ k)\ (g\ k)$
⟨*proof*⟩

end

2 Minimal elements of sets w.r.t. a well-founded and transitive relation

theory *Minimal-Elements*
imports
Infinite-Sequences
Open-Induction.Restricted-Predicates
begin

locale *minimal-element* =
fixes *P A*
assumes *po*: *po-on P A*
and *wf*: *wfp-on P A*
begin

definition *min-elt B* = $(SOME\ x.\ x \in B \wedge (\forall\, y \in A.\ P\ y\ x \longrightarrow y \notin B))$

lemma *minimal*:
assumes $x \in A$ **and** $Q\ x$
shows $\exists\, y \in A.\ P^{==}\ y\ x \wedge Q\ y \wedge (\forall\, z \in A.\ P\ z\ y \longrightarrow \neg\ Q\ z)$
⟨*proof*⟩

lemma *min-elt-ex*:
assumes $B \subseteq A$ **and** $B \neq \{\}$
shows $\exists\, x.\ x \in B \wedge (\forall\, y \in A.\ P\ y\ x \longrightarrow y \notin B)$
⟨*proof*⟩

lemma *min-elt-mem*:
assumes $B \subseteq A$ **and** $B \neq \{\}$
shows *min-elt B* $\in B$

⟨*proof*⟩

lemma *min-elt-minimal*:
assumes ∗: $B \subseteq A$ $B \neq \{\}$
assumes $y \in A$ **and** $P\ y$ (*min-elt* B)
shows $y \notin B$
⟨*proof*⟩

A lexicographically minimal sequence w.r.t. a given set of sequences C

fun *lexmin*
where
lexmin: *lexmin* $C\ i$ = *min-elt* (*ith* (*eq-upto* C (*lexmin* C) i) i)
declare *lexmin* [*simp del*]

lemma *eq-upto-lexmin-non-empty*:
assumes $C \subseteq SEQ\ A$ **and** $C \neq \{\}$
shows *eq-upto* C (*lexmin* C) $i \neq \{\}$
⟨*proof*⟩

lemma *lexmin-SEQ-mem*:
assumes $C \subseteq SEQ\ A$ **and** $C \neq \{\}$
shows *lexmin* $C \in SEQ\ A$
⟨*proof*⟩

lemma *non-empty-ith*:
assumes $C \subseteq SEQ\ A$ **and** $C \neq \{\}$
shows *ith* (*eq-upto* C (*lexmin* C) i) $i \subseteq A$
and *ith* (*eq-upto* C (*lexmin* C) i) $i \neq \{\}$
⟨*proof*⟩

lemma *lexmin-minimal*:
$C \subseteq SEQ\ A \Longrightarrow C \neq \{\} \Longrightarrow y \in A \Longrightarrow P\ y$ (*lexmin* $C\ i$) $\Longrightarrow y \notin$ *ith* (*eq-upto* C (*lexmin* C) i) i
⟨*proof*⟩

lemma *lexmin-mem*:
$C \subseteq SEQ\ A \Longrightarrow C \neq \{\} \Longrightarrow$ *lexmin* $C\ i \in$ *ith* (*eq-upto* C (*lexmin* C) i) i
⟨*proof*⟩

lemma *LEX-chain-on-eq-upto-imp-ith-chain-on*:
assumes *chain-on* (*LEX* P) (*eq-upto* $C\ f\ i$) (*SEQ* A)
shows *chain-on* P (*ith* (*eq-upto* $C\ f\ i$) i) A
⟨*proof*⟩

end

end

3 Enumerations of Well-Ordered Sets in Increasing Order

theory *Least-Enum*
imports *Main*
begin

locale *infinitely-many1* =
  fixes $P :: 'a :: wellorder \Rightarrow bool$
  assumes *infm*: $\forall i.\ \exists j{>}i.\ P\ j$
begin

Enumerate the elements of a well-ordered infinite set in increasing order.

fun *enum* :: $nat \Rightarrow 'a$ **where**
  $enum\ 0 = (LEAST\ n.\ P\ n)$ |
  $enum\ (Suc\ i) = (LEAST\ n.\ n > enum\ i \wedge P\ n)$

lemma *enum-mono*:
  shows $enum\ i < enum\ (Suc\ i)$
  ⟨*proof*⟩

lemma *enum-less*:
  $i < j \Longrightarrow enum\ i < enum\ j$
  ⟨*proof*⟩

lemma *enum-P*:
  shows $P\ (enum\ i)$
  ⟨*proof*⟩

end

locale *infinitely-many2* =
  fixes $P :: 'a :: wellorder \Rightarrow 'a \Rightarrow bool$
    and $N :: 'a$
  assumes *infm*: $\forall i{\geq}N.\ \exists j{>}i.\ P\ i\ j$
begin

Enumerate the elements of a well-ordered infinite set that form a chain w.r.t. a given predicate P starting from a given index N in increasing order.

fun *enumchain* :: $nat \Rightarrow 'a$ **where**
  $enumchain\ 0 = N$ |
  $enumchain\ (Suc\ n) = (LEAST\ m.\ m > enumchain\ n \wedge P\ (enumchain\ n)\ m)$

lemma *enumchain-mono*:
  shows $N \leq enumchain\ i \wedge enumchain\ i < enumchain\ (Suc\ i)$
⟨*proof*⟩

lemma *enumchain-chain*:
  shows $P\ (enumchain\ i)\ (enumchain\ (Suc\ i))$

⟨*proof*⟩

end

end

4 The Almost-Full Property

theory *Almost-Full*
imports
HOL−Library.Sublist
HOL−Library.Ramsey
Regular−Sets.Regexp-Method
Abstract−Rewriting.Seq
Least-Enum
Infinite-Sequences
Open-Induction.Restricted-Predicates
begin

lemma *le-Suc-eq′*:
$x \leq Suc\ y \longleftrightarrow x = 0 \vee (\exists\, x'.\ x = Suc\ x' \wedge x' \leq y)$
⟨*proof*⟩

lemma *ex-leq-Suc*:
$(\exists\, i{\leq}Suc\ j.\ P\ i) \longleftrightarrow P\ 0 \vee (\exists\, i{\leq}j.\ P\ (Suc\ i))$
⟨*proof*⟩

lemma *ex-less-Suc*:
$(\exists\, i{<}Suc\ j.\ P\ i) \longleftrightarrow P\ 0 \vee (\exists\, i{<}j.\ P\ (Suc\ i))$
⟨*proof*⟩

4.1 Basic Definitions and Facts

An infinite sequence is *good* whenever there are indices $i < j$ such that $P\ (f\ i)\ (f\ j)$.

definition *good* :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow (nat \Rightarrow 'a) \Rightarrow bool$
where
$good\ P\ f \longleftrightarrow (\exists\, i\ j.\ i < j \wedge P\ (f\ i)\ (f\ j))$

A sequence that is not good is called *bad*.

abbreviation $bad\ P\ f \equiv \neg\ good\ P\ f$

lemma *goodI*:
$[\![i < j;\ P\ (f\ i)\ (f\ j)]\!] \Longrightarrow good\ P\ f$
⟨*proof*⟩

lemma *goodE* [*elim*]:

good P f $\Longrightarrow$ $(\bigwedge i\ j.\ [\![i < j;\ P\ (f\ i)\ (f\ j)]\!] \Longrightarrow Q) \Longrightarrow Q$
⟨*proof*⟩

lemma *badE* [*elim*]:
bad P f $\Longrightarrow$ $((\bigwedge i\ j.\ i < j \Longrightarrow \neg\ P\ (f\ i)\ (f\ j)) \Longrightarrow Q) \Longrightarrow Q$
⟨*proof*⟩

definition *almost-full-on* :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a\ set \Rightarrow bool$
where
almost-full-on P A $\longleftrightarrow$ $(\forall f \in SEQ\ A.\ good\ P\ f)$

lemma *almost-full-onI* [*Pure.intro*]:
$(\bigwedge f.\ \forall i.\ f\ i \in A \Longrightarrow good\ P\ f) \Longrightarrow$ *almost-full-on P A*
⟨*proof*⟩

lemma *almost-full-onD*:
fixes $f :: nat \Rightarrow 'a$ **and** $A :: 'a\ set$
assumes *almost-full-on P A* **and** $\bigwedge i.\ f\ i \in A$
obtains $i\ j$ **where** $i < j$ **and** $P\ (f\ i)\ (f\ j)$
⟨*proof*⟩

4.2 An equivalent inductive definition

inductive *af* **for** *A*
where
now: $(\bigwedge x\ y.\ x \in A \Longrightarrow y \in A \Longrightarrow P\ x\ y) \Longrightarrow af\ A\ P$
| *later*: $(\bigwedge x.\ x \in A \Longrightarrow af\ A\ (\lambda y\ z.\ P\ y\ z \vee P\ x\ y)) \Longrightarrow af\ A\ P$

lemma *af-imp-almost-full-on*:
assumes *af A P*
shows *almost-full-on P A*
⟨*proof*⟩

lemma *af-mono*:
assumes *af A P*
and $\forall x\ y.\ x \in A \wedge y \in A \wedge P\ x\ y \longrightarrow Q\ x\ y$
shows *af A Q*
⟨*proof*⟩

lemma *accessible-on-imp-af*:
assumes *accessible-on P A x*
shows *af A* $(\lambda u\ v.\ \neg\ P\ v\ u \vee \neg\ P\ u\ x)$
⟨*proof*⟩

lemma *wfp-on-imp-af*:
assumes *wfp-on P A*
shows *af A* $(\lambda x\ y.\ \neg\ P\ y\ x)$
⟨*proof*⟩

lemma *af-leq*:
af UNIV $((\leq) :: nat \Rightarrow nat \Rightarrow bool)$
⟨*proof*⟩

definition $NOTAF\ A\ P = (SOME\ x.\ x \in A \wedge \neg\ af\ A\ (\lambda y\ z.\ P\ y\ z \vee P\ x\ y))$

lemma *not-af*:
$\neg\ af\ A\ P \Longrightarrow (\exists\, x\ y.\ x \in A \wedge y \in A \wedge \neg\ P\ x\ y) \wedge (\exists\, x{\in}A.\ \neg\ af\ A\ (\lambda y\ z.\ P\ y\ z \vee P\ x\ y))$
⟨*proof*⟩

fun *F*
where
$F\ A\ P\ 0 = NOTAF\ A\ P$
| $F\ A\ P\ (Suc\ i) = (let\ x = NOTAF\ A\ P\ in\ F\ A\ (\lambda y\ z.\ P\ y\ z \vee P\ x\ y)\ i)$

lemma *almost-full-on-imp-af*:
assumes *af*: *almost-full-on P A*
shows *af A P*
⟨*proof*⟩

hide-const *NOTAF F*

lemma *almost-full-on-UNIV*:
almost-full-on $(\lambda\text{-}\ \text{-}.\ True)$ *UNIV*
⟨*proof*⟩

lemma *almost-full-on-imp-reflp-on*:
assumes *almost-full-on P A*
shows *reflp-on A P*
⟨*proof*⟩

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

lemma *almost-full-on-mono*:
assumes $A \subseteq B$ **and** $\bigwedge x\ y.\ Q\ x\ y \Longrightarrow P\ x\ y$
and *almost-full-on Q B*
shows *almost-full-on P A*
⟨*proof*⟩

Every sequence over elements of an almost-full set has a homogeneous subsequence.

lemma *almost-full-on-imp-homogeneous-subseq*:
assumes *almost-full-on P A*
and $\forall\, i{::}nat.\ f\ i \in A$
shows $\exists\, \varphi{::}nat \Rightarrow nat.\ \forall\, i\ j.\ i < j \longrightarrow \varphi\ i < \varphi\ j \wedge P\ (f\ (\varphi\ i))\ (f\ (\varphi\ j))$

⟨*proof*⟩

Almost full relations do not admit infinite antichains.

lemma *almost-full-on-imp-no-antichain-on*:
  assumes *almost-full-on P A*
  shows ¬ *antichain-on P f A*
⟨*proof*⟩

If the image of a function is almost-full then also its preimage is almost-full.

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

The homomorphic image of an almost-full set is almost-full.

lemma *almost-full-on-hom*:
  fixes *h* :: *'a* ⇒ *'b*
  assumes *hom*: ⋀*x y*. ⟦*x* ∈ *A*; *y* ∈ *A*; *P x y*⟧ ⟹ *Q* (*h x*) (*h y*)
    and *af*: *almost-full-on P A*
  shows *almost-full-on Q* (*h ‘ A*)
⟨*proof*⟩

The monomorphic preimage of an almost-full set is almost-full.

lemma *almost-full-on-mon*:
  assumes *mon*: ⋀*x y*. ⟦*x* ∈ *A*; *y* ∈ *A*⟧ ⟹ *P x y* = *Q* (*h x*) (*h y*) *bij-betw h A B*
    and *af*: *almost-full-on Q B*
  shows *almost-full-on P A*
⟨*proof*⟩

Every total and well-founded relation is almost-full.

lemma *total-on-and-wfp-on-imp-almost-full-on*:
  assumes *totalp-on A P* **and** *wfp-on P A*
  shows *almost-full-on P*$^{==}$ *A*
⟨*proof*⟩

lemma *Nil-imp-good-list-emb* [*simp*]:
  assumes *f i* = []
  shows *good* (*list-emb P*) *f*
⟨*proof*⟩

lemma *ne-lists*:
  assumes *xs* ≠ [] **and** *xs* ∈ *lists A*
  shows *hd xs* ∈ *A* **and** *tl xs* ∈ *lists A*
  ⟨*proof*⟩

lemma *list-emb-eq-length-induct* [*consumes 2*, *case-names Nil Cons*]:
  assumes *length xs* = *length ys*

and *list-emb P xs ys*
and *Q* [] []
and $\bigwedge x\ y\ xs\ ys.$ $[\![P\ x\ y;$ *list-emb P xs ys*; $Q\ xs\ ys]\!] \Longrightarrow Q\ (x\#xs)\ (y\#ys)$
shows *Q xs ys*
⟨*proof*⟩

lemma *list-emb-eq-length-P*:
assumes *length xs* = *length ys*
and *list-emb P xs ys*
shows $\forall i{<}$*length xs*. *P* (*xs* ! *i*) (*ys* ! *i*)
⟨*proof*⟩

4.3 Special Case: Finite Sets

Every reflexive relation on a finite set is almost-full.

lemma *finite-almost-full-on*:
assumes *finite*: *finite A*
and *refl*: *reflp-on A P*
shows *almost-full-on P A*
⟨*proof*⟩

lemma *eq-almost-full-on-finite-set*:
assumes *finite A*
shows *almost-full-on* (=) *A*
⟨*proof*⟩

4.4 Further Results

lemma *af-trans-extension-imp-wf*:
assumes *subrel*: $\bigwedge x\ y.\ P\ x\ y \Longrightarrow Q\ x\ y$
and *af*: *almost-full-on P A*
and *trans*: *transp-on A Q*
shows *wfp-on* (*strict Q*) *A*
⟨*proof*⟩

lemma *af-trans-imp-wf*:
assumes *almost-full-on P A*
and *transp-on A P*
shows *wfp-on* (*strict P*) *A*
⟨*proof*⟩

lemma *wf-and-no-antichain-imp-qo-extension-wf*:
assumes *wf*: *wfp-on* (*strict P*) *A*
and *anti*: $\neg$ ($\exists f.$ *antichain-on P f A*)
and *subrel*: $\forall x{\in}A.\ \forall y{\in}A.\ P\ x\ y \longrightarrow Q\ x\ y$
and *qo*: *qo-on Q A*
shows *wfp-on* (*strict Q*) *A*
⟨*proof*⟩

lemma *every-qo-extension-wf-imp-af*:
assumes *ext*: $\forall\, Q.\ (\forall\, x{\in}A.\ \forall\, y{\in}A.\ P\ x\ y \longrightarrow Q\ x\ y) \wedge$
qo-on $Q\ A \longrightarrow$ *wfp-on* (*strict* Q) A
and *qo-on* $P\ A$
shows *almost-full-on* $P\ A$
⟨*proof*⟩

end

5 Constructing Minimal Bad Sequences

theory *Minimal-Bad-Sequences*
imports
Almost-Full
Minimal-Elements
begin

A locale capturing the construction of minimal bad sequences over values from A. Where minimality is to be understood w.r.t. *size* of an element.

locale *mbs* =
fixes $A :: ('a :: size)\ set$
begin

Since the *size* is a well-founded measure, whenever some element satisfies a property P, then there is a size-minimal such element.

lemma *minimal*:
assumes $x \in A$ **and** $P\ x$
shows $\exists\, y \in A.\ size\ y \leq size\ x \wedge P\ y \wedge (\forall\, z \in A.\ size\ z < size\ y \longrightarrow \neg\ P\ z)$
⟨*proof*⟩

lemma *less-not-eq* [*simp*]:
$x \in A \Longrightarrow size\ x < size\ y \Longrightarrow x = y \Longrightarrow False$
⟨*proof*⟩

The set of all bad sequences over A.

definition $BAD\ P = \{f \in SEQ\ A.\ bad\ P\ f\}$

lemma *BAD-iff* [*iff*]:
$f \in BAD\ P \longleftrightarrow (\forall\, i.\ f\ i \in A) \wedge bad\ P\ f$
⟨*proof*⟩

A partial order on infinite bad sequences.

definition *geseq* :: $((nat \Rightarrow 'a) \times (nat \Rightarrow 'a))\ set$
where
geseq =
$\{(f, g).\ f \in SEQ\ A \wedge g \in SEQ\ A \wedge (f = g \vee (\exists\, i.\ size\ (g\ i) < size\ (f\ i) \wedge (\forall\, j < i.\ f\ j = g\ j)))\}$

The strict part of the above order.

definition *gseq* :: $((nat \Rightarrow 'a) \times (nat \Rightarrow 'a))$ *set* **where**
$gseq = \{(f, g).\ f \in SEQ\ A \wedge g \in SEQ\ A \wedge (\exists i.\ size\ (g\ i) < size\ (f\ i) \wedge (\forall j < i.\ f\ j = g\ j))\}$

lemma *geseq-iff*:
$(f, g) \in geseq \longleftrightarrow$
$f \in SEQ\ A \wedge g \in SEQ\ A \wedge (f = g \vee (\exists i.\ size\ (g\ i) < size\ (f\ i) \wedge (\forall j < i.\ f\ j = g\ j)))$
⟨*proof*⟩

lemma *gseq-iff*:
$(f, g) \in gseq \longleftrightarrow f \in SEQ\ A \wedge g \in SEQ\ A \wedge (\exists i.\ size\ (g\ i) < size\ (f\ i) \wedge (\forall j < i.\ f\ j = g\ j))$
⟨*proof*⟩

lemma *geseqE*:
assumes $(f, g) \in geseq$
and $[\![\forall i.\ f\ i \in A; \forall i.\ g\ i \in A; f = g]\!] \Longrightarrow Q$
and $\bigwedge i.\ [\![\forall i.\ f\ i \in A; \forall i.\ g\ i \in A; size\ (g\ i) < size\ (f\ i); \forall j < i.\ f\ j = g\ j]\!] \Longrightarrow Q$
shows Q
⟨*proof*⟩

lemma *gseqE*:
assumes $(f, g) \in gseq$
and $\bigwedge i.\ [\![\forall i.\ f\ i \in A; \forall i.\ g\ i \in A; size\ (g\ i) < size\ (f\ i); \forall j < i.\ f\ j = g\ j]\!] \Longrightarrow Q$
shows Q
⟨*proof*⟩

sublocale *min-elt-size?*: *minimal-element measure-on size UNIV A*
rewrites *measure-on size UNIV* $\equiv \lambda x\ y.\ size\ x < size\ y$
⟨*proof*⟩

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

A lower bound to all sequences in a set of sequences B.

abbreviation $lb \equiv lexmin\ (BAD\ P)$

lemma *eq-upto-BAD-mem*:
assumes $f \in eq\text{-}upto\ (BAD\ P)\ g\ i$
shows $f\ j \in A$
⟨*proof*⟩

Assume that there is some infinite bad sequence h.

context

fixes $h :: nat \Rightarrow 'a$
assumes *BAD-ex*: $h \in BAD\ P$
begin

When there is a bad sequence, then filtering $BAD\ P$ w.r.t. positions in lb never yields an empty set of sequences.

lemma *eq-upto-BAD-non-empty*:
$eq\text{-}upto\ (BAD\ P)\ lb\ i \neq \{\}$
⟨*proof*⟩

lemma *non-empty-ith*:
shows $ith\ (eq\text{-}upto\ (BAD\ P)\ lb\ i)\ i \subseteq A$
and $ith\ (eq\text{-}upto\ (BAD\ P)\ lb\ i)\ i \neq \{\}$
⟨*proof*⟩

lemmas
$lb\text{-}minimal = min\text{-}elt\text{-}minimal\ [OF\ non\text{-}empty\text{-}ith, folded\ lexmin]$ **and**
$lb\text{-}mem = min\text{-}elt\text{-}mem\ [OF\ non\text{-}empty\text{-}ith, folded\ lexmin]$

lb is a infinite bad sequence.

lemma *lb-BAD*:
$lb \in BAD\ P$
⟨*proof*⟩

There is no infinite bad sequence that is strictly smaller than lb.

lemma *lb-lower-bound*:
$\forall g.\ (lb,\ g) \in gseq \longrightarrow g \notin BAD\ P$
⟨*proof*⟩

If there is at least one bad sequence, then there is also a minimal one.

lemma *lower-bound-ex*:
$\exists f \in BAD\ P.\ \forall g.\ (f,\ g) \in gseq \longrightarrow g \notin BAD\ P$
⟨*proof*⟩

lemma *gseq-conv*:
$(f,\ g) \in gseq \longleftrightarrow f \neq g \wedge (f,\ g) \in geseq$
⟨*proof*⟩

There is a minimal bad sequence.

lemma *mbs*:
$\exists f \in BAD\ P.\ \forall g.\ (f,\ g) \in gseq \longrightarrow good\ P\ g$
⟨*proof*⟩

end

end

end

end

6 A Proof of Higman's Lemma via Open Induction

theory *Higman-OI*
imports
Open-Induction.Open-Induction
Minimal-Elements
Almost-Full
begin

6.1 Some facts about the suffix relation

lemma *wfp-on-strict-suffix*:
wfp-on strict-suffix A
⟨*proof*⟩

lemma *po-on-strict-suffix*:
po-on strict-suffix A
⟨*proof*⟩

6.2 Lexicographic Order on Infinite Sequences

lemma *antisymp-on-LEX*:
assumes *irreflp-on A P* **and** *antisymp-on A P*
shows *antisymp-on* (*SEQ A*) (*LEX P*)
⟨*proof*⟩

lemma *LEX-trans*:
assumes *transp-on A P* **and** $f \in SEQ\ A$ **and** $g \in SEQ\ A$ **and** $h \in SEQ\ A$
and *LEX P f g* **and** *LEX P g h*
shows *LEX P f h*
⟨*proof*⟩

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

context *minimal-element*
begin

lemma *glb-LEX-lexmin*:
assumes *chain-on* (*LEX P*) *C* (*SEQ A*) **and** $C \neq \{\}$
shows *glb* (*LEX P*) *C* (*lexmin C*)
⟨*proof*⟩

lemma *dc-on-LEXEQ*:
dc-on (*LEXEQ P*) (*SEQ A*)

⟨*proof*⟩

end

Properties that only depend on finite initial segments of a sequence (i.e., which are open with respect to the product topology).

definition *pt-open-on Q A* ⟷ (∀ *f*∈*A*. *Q f* ⟷ (∃ *n*. (∀ *g*∈*A*. (∀ *i*<*n*. *g i* = *f i*) ⟶ *Q g*)))

lemma *pt-open-onD*:
pt-open-on Q A ⟹ *Q f* ⟹ *f* ∈ *A* ⟹ (∃ *n*. (∀ *g*∈*A*. (∀ *i*<*n*. *g i* = *f i*) ⟶ *Q g*))
⟨*proof*⟩

lemma *pt-open-on-good*:
pt-open-on (*good Q*) (*SEQ A*)
⟨*proof*⟩

context *minimal-element*
begin

lemma *pt-open-on-imp-open-on-LEXEQ*:
assumes *pt-open-on Q* (*SEQ A*)
shows *open-on* (*LEXEQ P*) *Q* (*SEQ A*)
⟨*proof*⟩

lemma *open-on-good*:
open-on (*LEXEQ P*) (*good Q*) (*SEQ A*)
⟨*proof*⟩

end

lemma *open-on-LEXEQ-imp-pt-open-on-counterexample*:
fixes *a b* :: *'a*
defines *A* ≡ {*a*, *b*} **and** *P* ≡ (λ*x y*. *False*) **and** *Q* ≡ (λ*f*. ∀ *i*. *f i* = *b*)
assumes [*simp*]: *a* ≠ *b*
shows *minimal-element P A* **and** *open-on* (*LEXEQ P*) *Q* (*SEQ A*)
and ¬ *pt-open-on Q* (*SEQ A*)
⟨*proof*⟩

lemma *higman*:
assumes *almost-full-on P A*
shows *almost-full-on* (*list-emb P*) (*lists A*)
⟨*proof*⟩

end

7 Almost-Full Relations

theory *Almost-Full-Relations*
imports *Minimal-Bad-Sequences*
begin

lemma (**in** *mbs*) *mbs'*:
  assumes ¬ *almost-full-on P A*
  shows ∃ *m* ∈ *BAD P*. ∀ *g*. (*m*, *g*) ∈ *gseq* ⟶ *good P g*
⟨*proof*⟩

7.1 Adding a Bottom Element to a Set

definition *with-bot* :: *'a set* ⇒ *'a option set* (‹-⊥› [*1000*] *1000*)
where
  $A_\perp$ = {*None*} ∪ *Some* ‘ *A*

lemma *with-bot-iff* [*iff*]:
  Some x ∈ $A_\perp$ ⟷ *x* ∈ *A*
  ⟨*proof*⟩

lemma *NoneI* [*simp*, *intro*]:
  None ∈ $A_\perp$
  ⟨*proof*⟩

lemma *not-None-the-mem* [*simp*]:
  x ≠ *None* ⟹ *the x* ∈ *A* ⟷ *x* ∈ $A_\perp$
  ⟨*proof*⟩

lemma *with-bot-cases*:
  u ∈ $A_\perp$ ⟹ (⋀*x*. *x* ∈ *A* ⟹ *u* = *Some x* ⟹ *P*) ⟹ (*u* = *None* ⟹ *P*) ⟹ *P*
  ⟨*proof*⟩

lemma *with-bot-empty-conv* [*iff*]:
  $A_\perp$ = {*None*} ⟷ *A* = {}
  ⟨*proof*⟩

lemma *with-bot-UNIV* [*simp*]:
  $UNIV_\perp$ = *UNIV*
⟨*proof*⟩

7.2 Adding a Bottom Element to an Almost-Full Set

fun
  option-le :: (*'a* ⇒ *'a* ⇒ *bool*) ⇒ *'a option* ⇒ *'a option* ⇒ *bool*
where
  option-le P None y = *True* |
  option-le P (*Some x*) *None* = *False* |
  option-le P (*Some x*) (*Some y*) = *P x y*

lemma *None-imp-good-option-le* [*simp*]:
assumes $f\ i = None$
shows *good* (*option-le P*) *f*
⟨*proof*⟩

lemma *almost-full-on-with-bot*:
assumes *almost-full-on P A*
shows *almost-full-on* (*option-le P*) $A_\perp$ (**is** *almost-full-on ?P ?A*)
⟨*proof*⟩

7.3 Disjoint Union of Almost-Full Sets

fun
sum-le :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow ('b \Rightarrow 'b \Rightarrow bool) \Rightarrow 'a + 'b \Rightarrow 'a + 'b \Rightarrow bool$
where
sum-le P Q (*Inl x*) (*Inl y*) = *P x y* |
sum-le P Q (*Inr x*) (*Inr y*) = *Q x y* |
sum-le P Q x y = *False*

lemma *not-sum-le-cases*:
assumes ¬ *sum-le P Q a b*
and $\bigwedge x\ y.\ [\![a = Inl\ x;\ b = Inl\ y;\ \neg\ P\ x\ y]\!] \Longrightarrow thesis$
and $\bigwedge x\ y.\ [\![a = Inr\ x;\ b = Inr\ y;\ \neg\ Q\ x\ y]\!] \Longrightarrow thesis$
and $\bigwedge x\ y.\ [\![a = Inl\ x;\ b = Inr\ y]\!] \Longrightarrow thesis$
and $\bigwedge x\ y.\ [\![a = Inr\ x;\ b = Inl\ y]\!] \Longrightarrow thesis$
shows *thesis*
⟨*proof*⟩

When two sets are almost-full, then their disjoint sum is almost-full.

lemma *almost-full-on-Plus*:
assumes *almost-full-on P A* **and** *almost-full-on Q B*
shows *almost-full-on* (*sum-le P Q*) (*A <+> B*) (**is** *almost-full-on ?P ?A*)
⟨*proof*⟩

7.4 Dickson's Lemma for Almost-Full Relations

When two sets are almost-full, then their Cartesian product is almost-full.

definition
prod-le :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow ('b \Rightarrow 'b \Rightarrow bool) \Rightarrow 'a \times 'b \Rightarrow 'a \times 'b \Rightarrow bool$
where
prod-le P1 P2 = $(\lambda(p1,\ p2)\ (q1,\ q2).\ P1\ p1\ q1 \wedge P2\ p2\ q2)$

lemma *prod-le-True* [*simp*]:
prod-le P (λ- -. *True*) *a b* = *P* (*fst a*) (*fst b*)
⟨*proof*⟩

lemma *almost-full-on-Sigma*:
assumes *almost-full-on P1 A1* **and** *almost-full-on P2 A2*
shows *almost-full-on* (*prod-le P1 P2*) (*A1 × A2*) (**is** *almost-full-on ?P ?A*)

⟨*proof*⟩

7.5 Higman's Lemma for Almost-Full Relations

lemma *almost-full-on-lists*:
  assumes *almost-full-on P A*
  shows *almost-full-on* (*list-emb P*) (*lists A*) (**is** *almost-full-on ?P ?A*)
⟨*proof*⟩

7.6 Natural Numbers

lemma *almost-full-on-UNIV-nat*:
  almost-full-on (≤) (*UNIV* :: *nat set*)
⟨*proof*⟩

end

8 Well-Quasi-Orders

theory *Well-Quasi-Orders*
imports *Almost-Full-Relations*
begin

8.1 Basic Definitions

definition *wqo-on* :: ('*a* ⇒ '*a* ⇒ *bool*) ⇒ '*a set* ⇒ *bool* **where**
  wqo-on P A ⟷ *transp-on A P* ∧ *almost-full-on P A*

lemma *wqo-on-UNIV*:
  wqo-on (λ- -. *True*) *UNIV*
  ⟨*proof*⟩

lemma *wqo-onI* [*Pure.intro*]:
  ⟦*transp-on A P*; *almost-full-on P A*⟧ ⟹ *wqo-on P A*
  ⟨*proof*⟩

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

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

lemma *wqo-on-imp-almost-full-on*:
  wqo-on P A ⟹ *almost-full-on P A*
  ⟨*proof*⟩

lemma *wqo-on-imp-qo-on*:

wqo-on P A $\Longrightarrow$ *qo-on P A*
⟨*proof*⟩

lemma *wqo-on-imp-good*:
wqo-on P A $\Longrightarrow$ $\forall\, i.\ f\ i \in A \Longrightarrow$ *good P f*
⟨*proof*⟩

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

8.2 Equivalent Definitions

Given a quasi-order P, the following statements are equivalent:

1. P is a almost-full.
2. P does neither allow decreasing chains nor antichains.
3. Every quasi-order extending P is well-founded.

lemma *wqo-af-conv*:
assumes *qo-on P A*
shows *wqo-on P A* $\longleftrightarrow$ *almost-full-on P A*
⟨*proof*⟩

lemma *wqo-wf-and-no-antichain-conv*:
assumes *qo-on P A*
shows *wqo-on P A* $\longleftrightarrow$ *wfp-on* (*strict P*) *A* $\wedge$ $\neg$ ($\exists f.$ *antichain-on P f A*)
⟨*proof*⟩

lemma *wqo-extensions-wf-conv*:
assumes *qo-on P A*
shows *wqo-on P A* $\longleftrightarrow$ ($\forall\, Q.$ ($\forall\, x{\in}A.\ \forall\, y{\in}A.$ *P x y* $\longrightarrow$ *Q x y*) $\wedge$ *qo-on Q A* $\longrightarrow$ *wfp-on* (*strict Q*) *A*)
⟨*proof*⟩

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

The homomorphic image of a wqo set is wqo.

lemma *wqo-on-hom*:
assumes *transp-on* (*h* ‘ *A*) *Q*
and $\forall\, x{\in}A.\ \forall\, y{\in}A.$ *P x y* $\longrightarrow$ *Q* (*h x*) (*h y*)
and *wqo-on P A*
shows *wqo-on Q* (*h* ‘ *A*)
⟨*proof*⟩

The monomorphic preimage of a wqo set is wqo.

lemma *wqo-on-mon*:
assumes ∗: ∀ *x*∈*A*. ∀ *y*∈*A*. *P x y* ⟷ *Q* (*h x*) (*h y*)
and *bij*: *bij-betw h A B*
and *wqo*: *wqo-on Q B*
shows *wqo-on P A*
⟨*proof*⟩

8.3 A Type Class for Well-Quasi-Orders

In a well-quasi-order (wqo) every infinite sequence is good.

class *wqo* = *preorder* +
assumes *good*: *good* (≤) *f*

lemma *wqo-on-class* [*simp*, *intro*]:
wqo-on (≤) (*UNIV* :: (′*a* :: *wqo*) *set*)
⟨*proof*⟩

lemma *wqo-on-UNIV-class-wqo* [*intro*!]:
wqo-on P UNIV ⟹ *class.wqo P* (*strict P*)
⟨*proof*⟩

The following lemma converts between *wqo-on* (for the special case that the domain is the universe of a type) and the class predicate *class.wqo*.

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

The strict part of a wqo is well-founded.

lemma (**in** *wqo*) *wfP* (<)
⟨*proof*⟩

lemma *wqo-on-with-bot*:
assumes *wqo-on P A*
shows *wqo-on* (*option-le P*) $A_\perp$ (**is** *wqo-on ?P ?A*)
⟨*proof*⟩

lemma *wqo-on-option-UNIV* [*intro*]:
wqo-on P UNIV ⟹ *wqo-on* (*option-le P*) *UNIV*
⟨*proof*⟩

When two sets are wqo, then their disjoint sum is wqo.

lemma *wqo-on-Plus*:
assumes *wqo-on P A* **and** *wqo-on Q B*
shows *wqo-on* (*sum-le P Q*) (*A* <+> *B*) (**is** *wqo-on ?P ?A*)
⟨*proof*⟩

lemma *wqo-on-sum-UNIV* [*intro*]:
wqo-on P UNIV ⟹ *wqo-on Q UNIV* ⟹ *wqo-on* (*sum-le P Q*) *UNIV*
⟨*proof*⟩

8.4 Dickson's Lemma

lemma *wqo-on-Sigma*:
fixes *A1* :: *'a set* **and** *A2* :: *'b set*
assumes *wqo-on P1 A1* **and** *wqo-on P2 A2*
shows *wqo-on* (*prod-le P1 P2*) (*A1* × *A2*) (**is** *wqo-on ?P ?A*)
⟨*proof*⟩

lemmas *dickson* = *wqo-on-Sigma*

lemma *wqo-on-prod-UNIV* [*intro*]:
wqo-on P UNIV ⟹ *wqo-on Q UNIV* ⟹ *wqo-on* (*prod-le P Q*) *UNIV*
⟨*proof*⟩

8.5 Higman's Lemma

lemma *transp-on-list-emb*:
assumes *transp-on A P*
shows *transp-on* (*lists A*) (*list-emb P*)
⟨*proof*⟩

lemma *wqo-on-lists*:
assumes *wqo-on P A* **shows** *wqo-on* (*list-emb P*) (*lists A*)
⟨*proof*⟩

lemmas *higman* = *wqo-on-lists*

lemma *wqo-on-list-UNIV* [*intro*]:
wqo-on P UNIV ⟹ *wqo-on* (*list-emb P*) *UNIV*
⟨*proof*⟩

Every reflexive and transitive relation on a finite set is a wqo.

lemma *finite-wqo-on*:
assumes *finite A* **and** *refl*: *reflp-on A P* **and** *transp-on A P*
shows *wqo-on P A*
⟨*proof*⟩

lemma *finite-eq-wqo-on*:
assumes *finite A*
shows *wqo-on* (=) *A*
⟨*proof*⟩

lemma *wqo-on-lists-over-finite-sets*:
wqo-on (*list-emb* (=)) (*UNIV*::(*'a*::*finite*) *list set*)
⟨*proof*⟩

lemma *wqo-on-map*:
fixes *P* **and** *Q* **and** *h*
defines *P'* ≡ λ*x y*. *P x y* ∧ *Q* (*h x*) (*h y*)
assumes *wqo-on P A*

and *wqo-on Q B*
and *subset*: h ‘ $A \subseteq B$
shows *wqo-on P′ A*
⟨*proof*⟩

lemma *wqo-on-UNIV-nat*:
wqo-on (≤) (*UNIV* :: *nat set*)
⟨*proof*⟩

end

9 Kruskal's Tree Theorem

theory *Kruskal*
imports *Well-Quasi-Orders*
begin

locale *kruskal-tree* =
fixes F :: $('b \times nat)$ *set*
and *mk* :: $'b \Rightarrow 'a$ *list* $\Rightarrow$ ($'a$::*size*)
and *root* :: $'a \Rightarrow 'b \times nat$
and *args* :: $'a \Rightarrow 'a$ *list*
and *trees* :: $'a$ *set*
assumes *size-arg*: $t \in trees \Longrightarrow s \in set\ (args\ t) \Longrightarrow size\ s < size\ t$
and *root-mk*: $(f, length\ ts) \in F \Longrightarrow root\ (mk\ f\ ts) = (f, length\ ts)$
and *args-mk*: $(f, length\ ts) \in F \Longrightarrow args\ (mk\ f\ ts) = ts$
and *mk-root-args*: $t \in trees \Longrightarrow mk\ (fst\ (root\ t))\ (args\ t) = t$
and *trees-root*: $t \in trees \Longrightarrow root\ t \in F$
and *trees-arity*: $t \in trees \Longrightarrow length\ (args\ t) = snd\ (root\ t)$
and *trees-args*: $\bigwedge s.\ t \in trees \Longrightarrow s \in set\ (args\ t) \Longrightarrow s \in trees$
begin

lemma *mk-inject* [*iff*]:
assumes $(f, length\ ss) \in F$ **and** $(g, length\ ts) \in F$
shows $mk\ f\ ss = mk\ g\ ts \longleftrightarrow f = g \wedge ss = ts$
⟨*proof*⟩

inductive *emb* **for** P
where
arg: $[\![(f, m) \in F; length\ ts = m; \forall\, t{\in}set\ ts.\ t \in trees;$
$t \in set\ ts; emb\ P\ s\ t]\!] \Longrightarrow emb\ P\ s\ (mk\ f\ ts)$ |
list-emb: $[\![(f, m) \in F; (g, n) \in F; length\ ss = m; length\ ts = n;$
$\forall\, s \in set\ ss.\ s \in trees; \forall\, t \in set\ ts.\ t \in trees;$
$P\ (f, m)\ (g, n); list\text{-}emb\ (emb\ P)\ ss\ ts]\!] \Longrightarrow emb\ P\ (mk\ f\ ss)\ (mk\ g\ ts)$
monos *list-emb-mono*

lemma *almost-full-on-trees*:
assumes *almost-full-on P F*
shows *almost-full-on* (*emb P*) *trees* (**is** *almost-full-on ?P ?A*)

⟨*proof*⟩

inductive-cases
emb-mk2 [*consumes 1*, *case-names arg list-emb*]: *emb P s* (*mk g ts*)

inductive-cases
list-emb-Nil2-cases: *list-emb P xs* [] **and**
list-emb-Cons-cases: *list-emb P xs* (*y*#*ys*)

lemma *list-emb-trans-right*:
assumes *list-emb P xs ys* **and** *list-emb* (λ*y z*. *P y z* ∧ (∀ *x*. *P x y* ⟶ *P x z*)) *ys zs*
shows *list-emb P xs zs*
⟨*proof*⟩

lemma *emb-trans*:
assumes *trans*: ⋀*f g h*. *f* ∈ *F* ⟹ *g* ∈ *F* ⟹ *h* ∈ *F* ⟹ *P f g* ⟹ *P g h* ⟹ *P f h*
assumes *emb P s t* **and** *emb P t u*
shows *emb P s u*
⟨*proof*⟩

lemma *transp-on-emb*:
assumes *transp-on F P*
shows *transp-on trees* (*emb P*)
⟨*proof*⟩

lemma *kruskal*:
assumes *wqo-on P F*
shows *wqo-on* (*emb P*) *trees*
⟨*proof*⟩

end

end
theory *Kruskal-Examples*
imports *Kruskal*
begin

datatype *'a tree* = *Node 'a 'a tree list*

fun *node*
where
node (*Node f ts*) = (*f*, *length ts*)

fun *succs*
where
succs (*Node f ts*) = *ts*

inductive-set *trees* **for** A
where
$f \in A \Longrightarrow \forall\, t \in set\ ts.\ t \in trees\ A \Longrightarrow Node\ f\ ts \in trees\ A$

lemma [*simp*]:
$trees\ UNIV = UNIV$
⟨*proof*⟩

interpretation *kruskal-tree-tree*: *kruskal-tree* $A \times$ *UNIV Node node succs trees* A
for A
⟨*proof*⟩

thm *kruskal-tree-tree.almost-full-on-trees*
thm *kruskal-tree-tree.kruskal*

definition *tree-emb* $A\ P = kruskal\text{-}tree\text{-}tree.emb\ A\ (prod\text{-}le\ P\ (\lambda\text{- -}.\ True))$

lemma *wqo-on-trees*:
assumes *wqo-on* $P\ A$
shows *wqo-on* (*tree-emb* $A\ P$) (*trees* A)
⟨*proof*⟩

If the type $'a$ is well-quasi-ordered by P, then trees of type $'a$ *tree* are well-quasi-ordered by the homeomorphic embedding relation.

instantiation *tree* :: (*wqo*) *wqo*
begin
definition $s \leq t \longleftrightarrow tree\text{-}emb\ UNIV\ (\leq)\ s\ t$
definition $(s :: 'a\ tree) < t \longleftrightarrow s \leq t \wedge \neg\ (t \leq s)$

instance
⟨*proof*⟩
end

datatype $('f, 'v)\ term = Var\ 'v \mid Fun\ 'f\ ('f, 'v)\ term\ list$

fun *root*
where
$root\ (Fun\ f\ ts) = (f, length\ ts)$

fun *args*
where
$args\ (Fun\ f\ ts) = ts$

inductive-set *gterms* **for** F
where
$(f, n) \in F \Longrightarrow length\ ts = n \Longrightarrow \forall\, s \in set\ ts.\ s \in gterms\ F \Longrightarrow Fun\ f\ ts \in gterms\ F$

interpretation *kruskal-term*: *kruskal-tree* F *Fun root args gterms* F **for** F

⟨*proof*⟩

thm *kruskal-term.almost-full-on-trees*

inductive-set *terms*
where
∀ *t* ∈ *set ts*. *t* ∈ *terms* ⟹ *Fun f ts* ∈ *terms*

interpretation *kruskal-variadic*: *kruskal-tree UNIV Fun root args terms*
⟨*proof*⟩

thm *kruskal-variadic.almost-full-on-trees*

datatype *'a exp* = *V 'a* | *C nat* | *Plus 'a exp 'a exp*

datatype *'a symb* = *v 'a* | *c nat* | *p*

fun *mk*
where
mk (*v x*) [] = *V x* |
mk (*c n*) [] = *C n* |
mk p [*a*, *b*] = *Plus a b*

fun *rt*
where
rt (*V x*) = (*v x*, *0*::*nat*) |
rt (*C n*) = (*c n*, *0*) |
rt (*Plus a b*) = (*p*, *2*)

fun *ags*
where
ags (*V x*) = [] |
ags (*C n*) = [] |
ags (*Plus a b*) = [*a*, *b*]

inductive-set *exps*
where
V x ∈ *exps* |
C n ∈ *exps* |
a ∈ *exps* ⟹ *b* ∈ *exps* ⟹ *Plus a b* ∈ *exps*

lemma [*simp*]:
assumes *length ts* = *2*
shows *rt* (*mk p ts*) = (*p*, *2*)
⟨*proof*⟩

lemma [*simp*]:
assumes *length ts* = *2*
shows *ags* (*mk p ts*) = *ts*

⟨*proof*⟩

interpretation *kruskal-exp*: *kruskal-tree*
{(*v x*, *0*) | *x*. *True*} ∪ {(*c n*, *0*) | *n*. *True*} ∪ {(*p*, *2*)}
mk rt ags exps
⟨*proof*⟩

thm *kruskal-exp.almost-full-on-trees*

hide-const (**open**) *tree-emb V C Plus v c p*

end

10 Instances of Well-Quasi-Orders

theory *Wqo-Instances*
imports *Kruskal*
begin

10.1 The Option Type is Well-Quasi-Ordered

instantiation *option* :: (*wqo*) *wqo*
begin
definition $x \leq y \longleftrightarrow$ *option-le* $(\leq)$ *x y*
definition $(x :: 'a\ option) < y \longleftrightarrow x \leq y \wedge \neg\ (y \leq x)$

instance
⟨*proof*⟩
end

10.2 The Sum Type is Well-Quasi-Ordered

instantiation *sum* :: (*wqo*, *wqo*) *wqo*
begin
definition $x \leq y \longleftrightarrow$ *sum-le* $(\leq)$ $(\leq)$ *x y*
definition $(x :: 'a + 'b) < y \longleftrightarrow x \leq y \wedge \neg\ (y \leq x)$

instance
⟨*proof*⟩
end

10.3 Pairs are Well-Quasi-Ordered

If types $'a$ and $'b$ are well-quasi-ordered by P and Q, then pairs of type $'a \times 'b$ are well-quasi-ordered by the pointwise combination of P and Q.

instantiation *prod* :: (*wqo*, *wqo*) *wqo*
begin
definition $p \leq q \longleftrightarrow$ *prod-le* $(\leq)$ $(\leq)$ *p q*

definition $(p :: 'a \times 'b) < q \longleftrightarrow p \leq q \wedge \neg\ (q \leq p)$

instance
⟨*proof*⟩
end

10.4 Lists are Well-Quasi-Ordered

If the type $'a$ is well-quasi-ordered by P, then lists of type $'a$ *list* are well-quasi-ordered by the homeomorphic embedding relation.

instantiation *list* :: (*wqo*) *wqo*
begin
definition $xs \leq ys \longleftrightarrow$ *list-emb* $(\leq)$ *xs ys*
definition $(xs :: 'a\ list) < ys \longleftrightarrow xs \leq ys \wedge \neg\ (ys \leq xs)$

instance
⟨*proof*⟩
end

end

11 Multiset Extension of Orders (as Binary Predicates)

theory *Multiset-Extension*
imports
Open-Induction.Restricted-Predicates
HOL−Library.Multiset
begin

definition *multisets* :: $'a$ *set* $\Rightarrow$ $'a$ *multiset set* **where**
multisets A = $\{M.$ *set-mset M* $\subseteq A\}$

lemma *in-multisets-iff*:
$M \in$ *multisets A* $\longleftrightarrow$ *set-mset M* $\subseteq A$
⟨*proof*⟩

lemma *empty-multisets* [*simp*]:
$\{\#\} \in$ *multisets F*
⟨*proof*⟩

lemma *multisets-union* [*simp*]:
$M \in$ *multisets A* $\Longrightarrow N \in$ *multisets A* $\Longrightarrow M + N \in$ *multisets A*
⟨*proof*⟩

definition *mulex1* :: $('a \Rightarrow 'a \Rightarrow$ *bool*$) \Rightarrow$ $'a$ *multiset* $\Rightarrow$ $'a$ *multiset* $\Rightarrow$ *bool* **where**
mulex1 P = $(\lambda M\ N.\ (M,\ N) \in$ *mult1* $\{(x,\ y).\ P\ x\ y\})$

lemma *mulex1-empty* [*iff*]:
mulex1 P M {#} ⟷ *False*
⟨*proof*⟩

lemma *mulex1-add*: *mulex1 P N* (*M0* + {#*a*#}) ⟹
(∃ *M*. *mulex1 P M M0* ∧ *N* = *M* + {#*a*#}) ∨
(∃ *K*. (∀ *b*. *b* ∈# *K* ⟶ *P b a*) ∧ *N* = *M0* + *K*)
⟨*proof*⟩

lemma *mulex1-self-add-right* [*simp*]:
mulex1 P A (*add-mset a A*)
⟨*proof*⟩

lemma *empty-mult1* [*simp*]:
({#}, {#*a*#}) ∈ *mult1 R*
⟨*proof*⟩

lemma *empty-mulex1* [*simp*]:
mulex1 P {#} {#*a*#}
⟨*proof*⟩

definition *mulex-on* :: ('*a* ⇒ '*a* ⇒ *bool*) ⇒ '*a set* ⇒ '*a multiset* ⇒ '*a multiset* ⇒ *bool* **where**
mulex-on P A = (*restrict-to* (*mulex1 P*) (*multisets A*))$^{++}$

abbreviation *mulex* :: ('*a* ⇒ '*a* ⇒ *bool*) ⇒ '*a multiset* ⇒ '*a multiset* ⇒ *bool* **where**
mulex P ≡ *mulex-on P UNIV*

lemma *mulex-on-induct* [*consumes 1*, *case-names base step*, *induct pred*: *mulex-on*]:
assumes *mulex-on P A M N*
and ⋀*M N*. ⟦*M* ∈ *multisets A*; *N* ∈ *multisets A*; *mulex1 P M N*⟧ ⟹ *Q M N*
and ⋀*L M N*. ⟦*mulex-on P A L M*; *Q L M*; *N* ∈ *multisets A*; *mulex1 P M N*⟧
⟹ *Q L N*
shows *Q M N*
⟨*proof*⟩

lemma *mulex-on-self-add-singleton-right* [*simp*]:
assumes *a* ∈ *A* **and** *M* ∈ *multisets A*
shows *mulex-on P A M* (*add-mset a M*)
⟨*proof*⟩

lemma *singleton-multisets* [*iff*]:
{#*x*#} ∈ *multisets A* ⟷ *x* ∈ *A*
⟨*proof*⟩

lemma *union-multisetsD*:
assumes *M* + *N* ∈ *multisets A*
shows *M* ∈ *multisets A* ∧ *N* ∈ *multisets A*

⟨*proof*⟩

lemma *mulex-on-multisetsD* [*dest*]:
assumes *mulex-on P F M N*
shows $M \in$ *multisets F* **and** $N \in$ *multisets F*
⟨*proof*⟩

lemma *union-multisets-iff* [*iff*]:
$M + N \in$ *multisets A* $\longleftrightarrow$ $M \in$ *multisets A* $\wedge$ $N \in$ *multisets A*
⟨*proof*⟩

lemma *add-mset-multisets-iff* [*iff*]:
add-mset a M $\in$ *multisets A* $\longleftrightarrow$ $a \in A \wedge M \in$ *multisets A*
⟨*proof*⟩

lemma *mulex-on-trans*:
mulex-on P A L M $\Longrightarrow$ *mulex-on P A M N* $\Longrightarrow$ *mulex-on P A L N*
⟨*proof*⟩

lemma *transp-on-mulex-on*:
transp-on B (*mulex-on P A*)
⟨*proof*⟩

lemma *mulex-on-add-right* [*simp*]:
assumes *mulex-on P A M N* **and** $a \in A$
shows *mulex-on P A M* (*add-mset a N*)
⟨*proof*⟩

lemma *empty-mulex-on* [*simp*]:
assumes $M \neq \{\#\}$ **and** $M \in$ *multisets A*
shows *mulex-on P A* $\{\#\}$ *M*
⟨*proof*⟩

lemma *mulex-on-self-add-right* [*simp*]:
assumes $M \in$ *multisets A* **and** $K \in$ *multisets A* **and** $K \neq \{\#\}$
shows *mulex-on P A M* $(M + K)$
⟨*proof*⟩

lemma *mult1-singleton* [*iff*]:
$(\{\#x\#\}, \{\#y\#\}) \in$ *mult1 R* $\longleftrightarrow$ $(x, y) \in R$
⟨*proof*⟩

lemma *mulex1-singleton* [*iff*]:
mulex1 P $\{\#x\#\}$ $\{\#y\#\}$ $\longleftrightarrow$ *P x y*
⟨*proof*⟩

lemma *singleton-mulex-onI*:
P x y $\Longrightarrow$ $x \in A \Longrightarrow$ $y \in A \Longrightarrow$ *mulex-on P A* $\{\#x\#\}$ $\{\#y\#\}$
⟨*proof*⟩

lemma *reflclp-mulex-on-add-right* [*simp*]:
assumes $(mulex\text{-}on\ P\ A)^{==}\ M\ N$ **and** $M \in multisets\ A$ **and** $a \in A$
shows $mulex\text{-}on\ P\ A\ M\ (N + \{\#a\#\})$
⟨*proof*⟩

lemma *reflclp-mulex-on-add-right′* [*simp*]:
assumes $(mulex\text{-}on\ P\ A)^{==}\ M\ N$ **and** $M \in multisets\ A$ **and** $a \in A$
shows $mulex\text{-}on\ P\ A\ M\ (\{\#a\#\} + N)$
⟨*proof*⟩

lemma *mulex-on-union-right* [*simp*]:
assumes $mulex\text{-}on\ P\ F\ A\ B$ **and** $K \in multisets\ F$
shows $mulex\text{-}on\ P\ F\ A\ (K + B)$
⟨*proof*⟩

lemma *mulex-on-union-right′* [*simp*]:
assumes $mulex\text{-}on\ P\ F\ A\ B$ **and** $K \in multisets\ F$
shows $mulex\text{-}on\ P\ F\ A\ (B + K)$
⟨*proof*⟩

Adapted from $wf\ ?r \Longrightarrow \forall M.\ M \in Wellfounded.acc\ (mult1\ ?r)$ in *HOL−Library.Multiset*.

lemma *accessible-on-mulex1-multisets*:
assumes *wf*: $wfp\text{-}on\ P\ A$
shows $\forall M{\in}multisets\ A.\ accessible\text{-}on\ (mulex1\ P)\ (multisets\ A)\ M$
⟨*proof*⟩

lemmas *wfp-on-mulex1-multisets* =
accessible-on-mulex1-multisets [*THEN accessible-on-imp-wfp-on*]

lemmas *irreflp-on-mulex1* =
wfp-on-mulex1-multisets [*THEN wfp-on-imp-irreflp-on*]

lemma *wfp-on-mulex-on-multisets*:
assumes $wfp\text{-}on\ P\ A$
shows $wfp\text{-}on\ (mulex\text{-}on\ P\ A)\ (multisets\ A)$
⟨*proof*⟩

lemmas *irreflp-on-mulex-on* =
wfp-on-mulex-on-multisets [*THEN wfp-on-imp-irreflp-on*]

lemma *mulex1-union*:
$mulex1\ P\ M\ N \Longrightarrow mulex1\ P\ (K + M)\ (K + N)$
⟨*proof*⟩

lemma *mulex-on-union*:
assumes $mulex\text{-}on\ P\ A\ M\ N$ **and** $K \in multisets\ A$
shows $mulex\text{-}on\ P\ A\ (K + M)\ (K + N)$
⟨*proof*⟩

lemma *mulex-on-union'*:
assumes *mulex-on P A M N* **and** $K \in$ *multisets A*
shows *mulex-on P A* $(M + K)$ $(N + K)$
⟨*proof*⟩

lemma *mulex-on-add-mset*:
assumes *mulex-on P A M N* **and** $m \in A$
shows *mulex-on P A* (*add-mset m M*) (*add-mset m N*)
⟨*proof*⟩

lemma *union-mulex-on-mono*:
mulex-on P F A C $\Longrightarrow$ *mulex-on P F B D* $\Longrightarrow$ *mulex-on P F* $(A + B)$ $(C + D)$
⟨*proof*⟩

lemma *mulex-on-add-mset'*:
assumes *P m n* **and** $m \in A$ **and** $n \in A$ **and** $M \in$ *multisets A*
shows *mulex-on P A* (*add-mset m M*) (*add-mset n M*)
⟨*proof*⟩

lemma *mulex-on-add-mset-mono*:
assumes *P m n* **and** $m \in A$ **and** $n \in A$ **and** *mulex-on P A M N*
shows *mulex-on P A* (*add-mset m M*) (*add-mset n N*)
⟨*proof*⟩

lemma *union-mulex-on-mono1*:
$A \in$ *multisets F* $\Longrightarrow$ (*mulex-on P F*)$^{==}$ *A C* $\Longrightarrow$ *mulex-on P F B D* $\Longrightarrow$
mulex-on P F $(A + B)$ $(C + D)$
⟨*proof*⟩

lemma *union-mulex-on-mono2*:
$B \in$ *multisets F* $\Longrightarrow$ *mulex-on P F A C* $\Longrightarrow$ (*mulex-on P F*)$^{==}$ *B D* $\Longrightarrow$
mulex-on P F $(A + B)$ $(C + D)$
⟨*proof*⟩

lemma *mult1-mono*:
assumes $\bigwedge x\ y.\ [\![x \in A;\ y \in A;\ (x,\ y) \in R]\!] \Longrightarrow (x,\ y) \in S$
and $M \in$ *multisets A*
and $N \in$ *multisets A*
and $(M,\ N) \in$ *mult1 R*
shows $(M,\ N) \in$ *mult1 S*
⟨*proof*⟩

lemma *mulex1-mono*:
assumes $\bigwedge x\ y.\ [\![x \in A;\ y \in A;\ P\ x\ y]\!] \Longrightarrow Q\ x\ y$
and $M \in$ *multisets A*
and $N \in$ *multisets A*
and *mulex1 P M N*
shows *mulex1 Q M N*

⟨*proof*⟩

lemma *mulex-on-mono*:
assumes ∗: $\bigwedge x\ y.\ [\![x \in A;\ y \in A;\ P\ x\ y]\!] \Longrightarrow Q\ x\ y$
and *mulex-on P A M N*
shows *mulex-on Q A M N*
⟨*proof*⟩

lemma *mult1-reflcl*:
assumes $(M,\ N) \in mult1\ R$
shows $(M,\ N) \in mult1\ (R^{=})$
⟨*proof*⟩

lemma *mulex1-reflclp*:
assumes *mulex1 P M N*
shows *mulex1* $(P^{==})$ *M N*
⟨*proof*⟩

lemma *mulex-on-reflclp*:
assumes *mulex-on P A M N*
shows *mulex-on* $(P^{==})$ *A M N*
⟨*proof*⟩

lemma *surj-on-multisets-mset*:
$\forall\, M{\in}multisets\ A.\ \exists\, xs{\in}lists\ A.\ M = mset\ xs$
⟨*proof*⟩

lemma *image-mset-lists* [*simp*]:
mset ‘ lists A = *multisets A*
⟨*proof*⟩

lemma *multisets-UNIV* [*simp*]: *multisets UNIV* = *UNIV*
⟨*proof*⟩

lemma *non-empty-multiset-induct* [*consumes 1*, *case-names singleton add*]:
assumes $M \neq \{\#\}$
and $\bigwedge x.\ P\ \{\#x\#\}$
and $\bigwedge x\ M.\ P\ M \Longrightarrow P\ (add\text{-}mset\ x\ M)$
shows *P M*
⟨*proof*⟩

lemma *mulex-on-all-strict*:
assumes $X \neq \{\#\}$
assumes $X \in multisets\ A$ **and** $Y \in multisets\ A$
and ∗: $\forall\, y.\ y \in\# Y \longrightarrow (\exists\, x.\ x \in\# X \wedge P\ y\ x)$
shows *mulex-on P A Y X*
⟨*proof*⟩

The following lemma shows that the textbook definition (e.g., “Term Rewrit-

ing and All That") is the same as the one used below.

lemma *diff-set-Ex-iff*:
$X \neq \{\#\} \wedge X \subseteq\# M \wedge N = (M - X) + Y \longleftrightarrow X \neq \{\#\} \wedge (\exists Z.\ M = Z + X \wedge N = Z + Y)$
⟨*proof*⟩

Show that *mulex-on* is equivalent to the textbook definition of multiset-extension for transitive base orders.

lemma *mulex-on-alt-def*:
assumes *trans*: *transp-on A P*
shows *mulex-on P A M N* $\longleftrightarrow$ $M \in$ *multisets A* $\wedge$ $N \in$ *multisets A* $\wedge$ $(\exists X\ Y\ Z.$
$X \neq \{\#\} \wedge N = Z + X \wedge M = Z + Y \wedge (\forall y.\ y \in\#\ Y \longrightarrow (\exists x.\ x \in\#\ X \wedge P\ y\ x)))$
(**is** *?P M N* $\longleftrightarrow$ *?Q M N*)
⟨*proof*⟩

end

12 Multiset Extension Preserves Well-Quasi-Orders

theory *Wqo-Multiset*
imports
Multiset-Extension
Well-Quasi-Orders
begin

lemma *list-emb-imp-reflclp-mulex-on*:
assumes *xs* $\in$ *lists A* **and** *ys* $\in$ *lists A*
and *list-emb P xs ys*
shows $(mulex\text{-}on\ P\ A)^{==}$ *(mset xs) (mset ys)*
⟨*proof*⟩

The (reflexive closure of the) multiset extension of an almost-full relation is almost-full.

lemma *almost-full-on-multisets*:
assumes *almost-full-on P A*
shows *almost-full-on* $(mulex\text{-}on\ P\ A)^{==}$ *(multisets A)*
⟨*proof*⟩

lemma *wqo-on-multisets*:
assumes *wqo-on P A*
shows *wqo-on* $(mulex\text{-}on\ P\ A)^{==}$ *(multisets A)*
⟨*proof*⟩

end

References

[1] C. S. J. A. Nash-Williams. On well-quasi-ordering finite trees. *Proceedings of the Cambridge Philosophical Society*, 59(4):833–835, 1963. doi:10.1017/S0305004100003844.