

Order Extension and Szpilrajn's Theorem

Peter Zeller and Lukas Stevens

May 26, 2024

We formalize a more general version of Szpilrajn's extension theorem [3], employing the terminology of Bossert and Suzumura [2]. We also formalize Theorem 2.7 of their book. Our extension theorem states that any preorder can be extended to a total preorder while maintaining its structure. The proof of the extension theorem follows the proof presented in the Wikipedia article [1].

1 Definitions

1.1 Symmetric and asymmetric factor of a relation

According to Bossert and Suzumura, every relation can be partitioned into its symmetric and asymmetric factor. The symmetric factor of a relation r contains all pairs $(x, y) \in r$ where $(y, x) \in r$. Conversely, the asymmetric factor contains all pairs where this is not the case. In terms of an order (\leq), the asymmetric factor contains all $(x, y) \in \{(x, y) \mid x \leq y\}$ where $x < y$.

definition *sym-factor* :: 'a rel \Rightarrow 'a rel
where *sym-factor* $r \equiv \{(x, y) \in r. (y, x) \in r\}$

lemma *sym-factor-def'*: *sym-factor* $r = r \cap r^{-1}$
unfolding *sym-factor-def* **by** *fast*

definition *asym-factor* :: 'a rel \Rightarrow 'a rel
where *asym-factor* $r = \{(x, y) \in r. (y, x) \notin r\}$

1.1.1 Properties of the symmetric factor

lemma *sym-factorI[intro]*: $(x, y) \in r \implies (y, x) \in r \implies (x, y) \in \text{sym-factor } r$
unfolding *sym-factor-def* **by** *blast*

lemma *sym-factorE[elim?]*:
assumes $(x, y) \in \text{sym-factor } r$ **obtains** $(x, y) \in r (y, x) \in r$
using *assms[unfolded sym-factor-def]* **by** *blast*

lemma *sym-sym-factor*[simp]: $\text{sym} (\text{sym-factor } r)$
unfolding *sym-factor-def*
by (*auto intro!*: *symI*)

lemma *trans-sym-factor*[simp]: $\text{trans } r \implies \text{trans} (\text{sym-factor } r)$
unfolding *sym-factor-def'* **using** *trans-Int* **by** *force*

lemma *refl-on-sym-factor*[simp]: $\text{refl-on } A \ r \implies \text{refl-on } A \ (\text{sym-factor } r)$
unfolding *sym-factor-def*
by (*auto intro!*: *refl-onI* *dest*: *refl-onD* *refl-onD1*)

lemma *sym-factor-absorb-if-sym*[simp]: $\text{sym } r \implies \text{sym-factor } r = r$
unfolding *sym-factor-def'*
by (*simp add*: *sym-conv-converse-eq*)

lemma *sym-factor-idem*[simp]: $\text{sym-factor} (\text{sym-factor } r) = \text{sym-factor } r$
using *sym-factor-absorb-if-sym*[*OF sym-sym-factor*] .

lemma *sym-factor-reflc*[simp]: $\text{sym-factor} (r^\equiv) = (\text{sym-factor } r)^\equiv$
unfolding *sym-factor-def* **by** *auto*

lemma *sym-factor-Restr*[simp]: $\text{sym-factor} (\text{Restr } r \ A) = \text{Restr} (\text{sym-factor } r) \ A$
unfolding *sym-factor-def* **by** *blast*

In contrast to *asym-factor*, the *sym-factor* is monotone.

lemma *sym-factor-mono*: $r \subseteq s \implies \text{sym-factor } r \subseteq \text{sym-factor } s$
unfolding *sym-factor-def* **by** *auto*

1.1.2 Properties of the asymmetric factor

lemma *asym-factorI*[*intro*]: $(x, y) \in r \implies (y, x) \notin r \implies (x, y) \in \text{asym-factor } r$
unfolding *asym-factor-def* **by** *blast*

lemma *asym-factorE*[*elim?*]:
assumes $(x, y) \in \text{asym-factor } r$ **obtains** $(x, y) \in r$
using *assms* **unfolding** *asym-factor-def* **by** *blast*

lemma *refl-not-in-asym-factor*[simp]: $(x, x) \notin \text{asym-factor } r$
unfolding *asym-factor-def* **by** *blast*

lemma *irrefl-asym-factor*[simp]: $\text{irrefl} (\text{asym-factor } r)$
unfolding *asym-factor-def* *irrefl-def* **by** *fast*

lemma *asym-asym-factor*[simp]: $\text{asym} (\text{asym-factor } r)$
using *irrefl-asym-factor*
by (*auto intro!*: *asymI* *simp*: *asym-factor-def*)

lemma *trans-asym-factor*[simp]: $\text{trans } r \implies \text{trans} (\text{asym-factor } r)$
unfolding *asym-factor-def* *trans-def* **by** *fast*

lemma *asym-if-irrefl-trans*: $irrefl\ r \implies trans\ r \implies asym\ r$
by (*intro asymI*) (*auto simp: irrefl-def trans-def*)

lemma *antisym-if-irrefl-trans*: $irrefl\ r \implies trans\ r \implies antisym\ r$
using *antisym-def asym-if-irrefl-trans* **by** (*auto dest: asymD*)

lemma *asym-factor-asym-rel[simp]*: $asym\ r \implies asym\text{-factor}\ r = r$
unfolding *asym-factor-def*
by (*auto dest: asymD*)

lemma *irrefl-trans-asym-factor-id[simp]*: $irrefl\ r \implies trans\ r \implies asym\text{-factor}\ r = r$
using *asym-factor-asym-rel[OF asym-if-irrefl-trans]* .

lemma *asym-factor-id[simp]*: $asym\text{-factor}\ (asym\text{-factor}\ r) = asym\text{-factor}\ r$
using *asym-factor-asym-rel[OF asym-asym-factor]* .

lemma *asym-factor-rtrancl*: $asym\text{-factor}\ (r^*) = asym\text{-factor}\ (r^+)$
unfolding *asym-factor-def*
by (*auto simp add: rtrancl-eq-or-trancl*)

lemma *asym-factor-Restr[simp]*: $asym\text{-factor}\ (Restr\ r\ A) = Restr\ (asym\text{-factor}\ r)\ A$
unfolding *asym-factor-def* **by** *blast*

lemma *acyclic-asym-factor[simp]*: $acyclic\ r \implies acyclic\ (asym\text{-factor}\ r)$
unfolding *asym-factor-def* **by** (*auto intro: acyclic-subset*)

1.1.3 Relations between symmetric and asymmetric factor

We prove that *sym-factor* and *asym-factor* partition the input relation.

lemma *sym-asym-factor-Un*: $sym\text{-factor}\ r \cup asym\text{-factor}\ r = r$
unfolding *sym-factor-def asym-factor-def* **by** *blast*

lemma *disjnt-sym-asym-factor[simp]*: $disjnt\ (sym\text{-factor}\ r)\ (asym\text{-factor}\ r)$
unfolding *disjnt-def*
unfolding *sym-factor-def asym-factor-def* **by** *blast*

lemma *Field-sym-asym-factor-Un*:
 $Field\ (sym\text{-factor}\ r) \cup Field\ (asym\text{-factor}\ r) = Field\ r$
using *sym-asym-factor-Un Field-Un* **by** *metis*

lemma *asym-factor-tranclE*:
assumes $(a, b) \in (asym\text{-factor}\ r)^+$ **shows** $(a, b) \in r^+$
using *assms sym-asym-factor-Un*
by (*metis UnCI subsetI trancl-mono*)

1.2 Extension of Orders

We use the definition of Bossert and Suzumura for *extends*. The requirement $r \subseteq R$ is obvious. The second requirement *asym-factor* $r \subseteq \text{asym-factor } R$ enforces that the extension R maintains all strict preferences of r (viewing r as a preference relation).

definition *extends* :: 'a rel \Rightarrow 'a rel \Rightarrow bool
where *extends* R $r \equiv r \subseteq R \wedge \text{asym-factor } r \subseteq \text{asym-factor } R$

We define a stronger notion of *extends* where we also demand that *sym-factor* $R \subseteq (\text{sym-factor } r)^=$. This enforces that the extension does not introduce preference cycles between previously unrelated pairs $(x, y) \in R - r$.

definition *strict-extends* :: 'a rel \Rightarrow 'a rel \Rightarrow bool
where *strict-extends* R $r \equiv \text{extends } R$ $r \wedge \text{sym-factor } R \subseteq (\text{sym-factor } r)^=$

lemma *extendsI[intro]*: $r \subseteq R \implies \text{asym-factor } r \subseteq \text{asym-factor } R \implies \text{extends } R$ r
unfolding *extends-def* **by** (*intro conjI*)

lemma *extendsE*:
assumes *extends* R r
obtains $r \subseteq R$ *asym-factor* $r \subseteq \text{asym-factor } R$
using *assms* **unfolding** *extends-def* **by** *blast*

lemma *trancl Subs extends-if-trans*: *extends* $r\text{-ext } r \implies \text{trans } r\text{-ext} \implies r^+ \subseteq r\text{-ext}$
unfolding *extends-def* *asym-factor-def*
by (*metis subrelI trancl-id trancl-mono*)

lemma *extends-if-strict-extends*: *strict-extends* $r\text{-ext } \text{ext} \implies \text{extends } r\text{-ext } \text{ext}$
unfolding *strict-extends-def* **by** *blast*

lemma *strict-extendsI[intro]*:
assumes $r \subseteq R$ *asym-factor* $r \subseteq \text{asym-factor } R$ *sym-factor* $R \subseteq (\text{sym-factor } r)^=$
shows *strict-extends* R r
unfolding *strict-extends-def* **using** *assms* **by** (*intro conjI extendsI*)

lemma *strict-extendsE*:
assumes *strict-extends* R r
obtains $r \subseteq R$ *asym-factor* $r \subseteq \text{asym-factor } R$ *sym-factor* $R \subseteq (\text{sym-factor } r)^=$
using *assms* *extendsE* **unfolding** *strict-extends-def* **by** *blast*

lemma *strict-extends-antisym-Restr*:
assumes *strict-extends* R r
assumes *antisym* (*Restr* r A)
shows *antisym* $((R - r) \cup \text{Restr } r$ A)

proof(*rule antisymI, rule ccontr*)

fix x y **assume** $(x, y) \in (R - r) \cup \text{Restr } r$ A $(y, x) \in (R - r) \cup \text{Restr } r$ A $x \neq y$

```

with  $\langle \text{strict-extends } R \ r \rangle$  have  $(x, y) \in \text{sym-factor } R$ 
  unfolding sym-factor-def by (auto elim!: strict-extendsE)
with assms  $\langle x \neq y \rangle$  have  $(x, y) \in \text{sym-factor } r$ 
  by (auto elim!: strict-extendsE)
then have  $(x, y) \in r \ (y, x) \in r$ 
  unfolding sym-factor-def by simp-all
with  $\langle \text{antisym } (\text{Restr } r \ A) \rangle$   $\langle x \neq y \rangle$   $\langle (y, x) \in R - r \cup \text{Restr } r \ A \rangle$  show False
  using antisymD by fastforce
qed

```

Here we prove that we have no preference cycles between previously unrelated pairs.

```

lemma antisym-Diff-if-strict-extends:
  assumes strict-extends  $R \ r$ 
  shows antisym  $(R - r)$ 
  using strict-extends-antisym-Restr[OF assms, where  $?A=\{\}$ ] by simp

```

```

lemma strict-extends-antisym:
  assumes strict-extends  $R \ r$ 
  assumes antisym  $r$ 
  shows antisym  $R$ 
  using assms strict-extends-antisym-Restr[OF assms(1), where  $?A=UNIV$ ]
  by (auto elim!: strict-extendsE simp: antisym-def)

```

```

lemma strict-extends-if-strict-extends-reflc:
  assumes strict-extends  $r\text{-ext } (r^=)$ 
  shows strict-extends  $r\text{-ext } r$ 
proof(intro strict-extendsI)
  from assms show  $r \subseteq r\text{-ext}$ 
    by (auto elim!: strict-extendsE)

  from assms  $\langle r \subseteq r\text{-ext} \rangle$  show asym-factor  $r \subseteq \text{asym-factor } r\text{-ext}$ 
    unfolding strict-extends-def
    by (auto simp!: asym-factor-def sym-factor-def)

  from assms show sym-factor  $r\text{-ext} \subseteq (\text{sym-factor } r)^=$ 
    by (auto simp!: sym-factor-def strict-extends-def)
qed

```

```

lemma strict-extends-diff-Id:
  assumes irrefl  $r$  trans  $r$ 
  assumes strict-extends  $r\text{-ext } (r^=)$ 
  shows strict-extends  $(r\text{-ext} - \text{Id}) \ r$ 
proof(intro strict-extendsI)
  from assms show  $r \subseteq r\text{-ext} - \text{Id}$ 
    by (auto elim!: strict-extendsE simp!: irrefl-def)

```

```

note antisym-r = antisym-if-irrefl-trans[OF assms(1,2)]
with assms strict-extends-if-strict-extends-reflc show asym-factor  $r \subseteq \text{asym-factor}$ 

```

(r-ext - Id)
unfolding *asym-factor-def*
by (*auto intro: strict-extends-antisym[THEN antisymD] elim: strict-extendsE transE*)

from *assms antisym-r* **show** *sym-factor (r-ext - Id) ⊆ (sym-factor r)⁼*
unfolding *sym-factor-def*
by (*auto intro: strict-extends-antisym[THEN antisymD]*)
qed

Both *extends* and *strict-extends* form a partial order since they are reflexive, transitive, and antisymmetric.

lemma shows
reflp-extends: reflp extends and
transp-extends: transp extends and
antisymp-extends: antisymp extends
unfolding *extends-def reflp-def transp-def antisymp-def*
by *auto*

lemma shows
reflp-strict-extends: reflp strict-extends and
transp-strict-extends: transp strict-extends and
antisymp-strict-extends: antisymp strict-extends
using *reflp-extends transp-extends antisymp-extends*
unfolding *strict-extends-def reflp-def transp-def antisymp-def*
by *auto*

1.3 Missing order definitions

lemma *preorder-onD[dest?]*:
assumes *preorder-on A r*
shows *refl-on A r trans r*
using *assms* **unfolding** *preorder-on-def* **by** *blast+*

lemma *preorder-onI[intro]*: *refl-on A r ⇒ trans r ⇒ preorder-on A r*
unfolding *preorder-on-def* **by** (*intro conjI*)

abbreviation *preorder* ≡ *preorder-on UNIV*

lemma *preorder-rtrancl: preorder (r*)*
by (*intro preorder-onI refl-rtrancl trans-rtrancl*)

definition *total-preorder-on A r* ≡ *preorder-on A r ∧ total-on A r*

abbreviation *total-preorder r* ≡ *total-preorder-on UNIV r*

lemma *total-preorder-onI[intro]*:
refl-on A r ⇒ trans r ⇒ total-on A r ⇒ total-preorder-on A r
unfolding *total-preorder-on-def* **by** (*intro conjI preorder-onI*)

lemma *total-preorder-onD*[*dest?*]:
assumes *total-preorder-on A r*
shows *refl-on A r trans r total-on A r*
using *assms unfolding total-preorder-on-def preorder-on-def* **by** *blast+*

definition *strict-partial-order r* \equiv *trans r* \wedge *irrefl r*

lemma *strict-partial-orderI*[*intro*]:
trans r \implies *irrefl r* \implies *strict-partial-order r*
unfolding *strict-partial-order-def* **by** *blast*

lemma *strict-partial-orderD*[*dest?*]:
assumes *strict-partial-order r*
shows *trans r irrefl r*
using *assms unfolding strict-partial-order-def* **by** *blast+*

lemma *strict-partial-order-acyclic*:
assumes *strict-partial-order r*
shows *acyclic r*
by (*metis acyclic-irrefl assms strict-partial-order-def trancl-id*)

abbreviation *partial-order* \equiv *partial-order-on UNIV*

lemma *partial-order-onI*[*intro*]:
refl-on A r \implies *trans r* \implies *antisym r* \implies *partial-order-on A r*
using *partial-order-on-def* **by** *blast*

lemma *linear-order-onI*[*intro*]:
refl-on A r \implies *trans r* \implies *antisym r* \implies *total-on A r* \implies *linear-order-on A r*
using *linear-order-on-def* **by** *blast*

lemma *linear-order-onD*[*dest?*]:
assumes *linear-order-on A r*
shows *refl-on A r trans r antisym r total-on A r*
using *assms[unfolded linear-order-on-def] partial-order-onD* **by** *blast+*

A typical example is (\subset) on sets:

lemma *strict-partial-order-subset*:
strict-partial-order $\{(x,y). x \subset y\}$
proof
show *trans* $\{(x,y). x \subset y\}$
by (*auto simp add: trans-def*)
show *irrefl* $\{(x,y). x \subset y\}$
by (*simp add: irrefl-def*)
qed

We already have a definition of a strict linear order in *strict-linear-order*.

2 Extending preorders to total preorders

We start by proving that a preorder with two incomparable elements x and y can be strictly extended to a preorder where $x < y$.

lemma *can-extend-preorder*:

assumes *preorder-on* A r
and $y \in A$ $x \in A$ $(y, x) \notin r$

shows

preorder-on A $((\text{insert } (x, y) r)^+)$ *strict-extends* $((\text{insert } (x, y) r)^+)$ r

proof –

note *preorder-onD*[*OF* $\langle \text{preorder-on } A \ r \rangle$]

then have $\text{insert } (x, y) r \subseteq A \times A$

using $\langle y \in A \rangle \langle x \in A \rangle$ *refl-on-domain* **by** *fast*

with $\langle \text{refl-on } A \ r \rangle$ **show** *preorder-on* A $((\text{insert } (x, y) r)^+)$

by (*intro preorder-onI refl-onI trans-trancl*)

(*auto simp: trancl-subset-Sigma intro!: r-into-trancl' dest: refl-onD*)

show *strict-extends* $((\text{insert } (x, y) r)^+)$ r

proof(*intro strict-extendsI*)

from *preorder-onD*(2)[*OF* $\langle \text{preorder-on } A \ r \rangle$] $\langle (y, x) \notin r \rangle$

show *asym-factor* $r \subseteq \text{asym-factor } ((\text{insert } (x, y) r)^+)$

unfolding *asym-factor-def trancl-insert*

using *rtranclD rtrancl-into-trancl1 r-r-into-trancl*

by *fastforce*

from *assms* **have** $(y, x) \notin (\text{insert } (x, y) r)^+$

unfolding *preorder-on-def trancl-insert*

using *refl-onD rtranclD* **by** *fastforce*

with $\langle \text{trans } r \rangle$ **show** *sym-factor* $((\text{insert } (x, y) r)^+) \subseteq (\text{sym-factor } r)^=$

unfolding *trancl-insert sym-factor-def* **by** (*fastforce intro: rtrancl-trans*)

qed *auto*

qed

With this, we can start the proof of our main extension theorem. For this we will use a variant of Zorns Lemma, which only considers nonempty chains:

lemma *Zorns-po-lemma-nonempty*:

assumes *po*: *Partial-order* r

and $u: \bigwedge C. \llbracket C \in \text{Chains } r; C \neq \{\} \rrbracket \implies \exists u \in \text{Field } r. \forall a \in C. (a, u) \in r$

and $r \neq \{\}$

shows $\exists m \in \text{Field } r. \forall a \in \text{Field } r. (m, a) \in r \longrightarrow a = m$

proof –

from $\langle r \neq \{\} \rangle$ **obtain** x **where** $x \in \text{Field } r$

using *FieldI2* **by** *fastforce*

with *assms* **show** *?thesis*

using *Zorns-po-lemma* **by** (*metis empty-iff*)

qed

theorem *strict-extends-preorder-on*:
assumes *preorder-on A base-r*
shows $\exists r. \text{total-preorder-on } A \ r \wedge \text{strict-extends } r \ \text{base-r}$
proof –

We define an order on the set of strict extensions of the base relation *base-r*, where $r \leq s$ iff *strict-extends r base-r* and *strict-extends s r*:

define *order-of-orders* :: $(\text{'a rel}) \text{ rel}$ **where** *order-of-orders* =
Restr $\{(r, s). \text{strict-extends } r \ \text{base-r} \wedge \text{strict-extends } s \ r\} \{r. \text{preorder-on } A \ r\}$

We show that this order consists of those relations that are preorders and that strictly extend the base relation *base-r*

have *Field-order-of-orders*: *Field order-of-orders* =
 $\{r. \text{preorder-on } A \ r \wedge \text{strict-extends } r \ \text{base-r}\}$
using *transp-strict-extends*

proof(*safe*)
fix *r* **assume** *preorder-on A r strict-extends r base-r*
with *reflp-strict-extends* **have**
 $(r, r) \in \{(r, s). \text{strict-extends } r \ \text{base-r} \wedge \text{strict-extends } s \ r\}$
by (*auto elim!*: *reflpE*)
with $\langle \text{preorder-on } A \ r \rangle$ **show** $r \in \text{Field order-of-orders}$
unfolding *order-of-orders-def* **by** (*auto simp*: *Field-def*)
qed (*auto simp*: *order-of-orders-def Field-def elim*: *transpE*)

We now show that this set has a maximum and that any maximum of this set is a total preorder and as thus is one of the extensions we are looking for. We begin by showing the existence of a maximal element using Zorn's lemma.

have $\exists m \in \text{Field order-of-orders}$.
 $\forall a \in \text{Field order-of-orders}. (m, a) \in \text{order-of-orders} \longrightarrow a = m$
proof (*rule Zorns-po-lemma-nonempty*)

Zorn's Lemma requires us to prove that our *order-of-orders* is a nonempty partial order and that every nonempty chain has an upper bound. The partial order property is trivial, since we used *strict-extends* for the relation, which is a partial order as shown above.

from *reflp-strict-extends transp-strict-extends*
have *Refl* $\{(r, s). \text{strict-extends } r \ \text{base-r} \wedge \text{strict-extends } s \ r\}$
unfolding *refl-on-def Field-def* **by** (*auto elim*: *transpE reflpE*)
moreover **have** *trans* $\{(r, s). \text{strict-extends } r \ \text{base-r} \wedge \text{strict-extends } s \ r\}$
using *transp-strict-extends* **by** (*auto elim*: *transpE intro*: *transI*)
moreover **have** *antisym* $\{(r, s). \text{strict-extends } r \ \text{base-r} \wedge \text{strict-extends } s \ r\}$
using *antisym-strict-extends* **by** (*fastforce dest*: *antisymD intro*: *antisymI*)

ultimately **show** *Partial-order order-of-orders*
unfolding *order-of-orders-def order-on-defs*
using *Field-order-of-orders Refl-Restr trans-Restr antisym-Restr*
by *blast*

Also, our order is obviously not empty since it contains $(base-r, base-r)$:

```

have  $(base-r, base-r) \in order-of-orders$ 
  unfolding  $order-of-orders-def$ 
  using  $assms reflp-strict-extends$  by  $(auto dest: reflpD)$ 
thus  $order-of-orders \neq \{\}$  by force

```

Next we show that each chain has an upper bound. For the upper bound we take the union of all relations in the chain.

```

show  $\exists u \in Field\ order-of-orders. \forall a \in C. (a, u) \in order-of-orders$ 
if  $C-def: C \in Chains\ order-of-orders$  and  $C-nonempty: C \neq \{\}$ 
for  $C$ 
proof  $(rule\ bexI[where\ x = \bigcup C])$ 

```

Obviously each element in the chain is a strict extension of $base-r$ by definition and as such it is also a preorder.

```

have  $preorder-r: preorder-on\ A\ r$  and  $extends-r: strict-extends\ r\ base-r$  if  $r \in C$  for  $r$ 
  using  $that\ C-def[unfolded\ order-of-orders-def\ Chains-def]$  by  $blast+$ 

```

Because a chain is partially ordered, the union of the chain is reflexive and transitive.

```

have  $total-sub-C: r \subseteq s \vee s \subseteq r$  if  $r \in C$  and  $s \in C$  for  $r\ s$ 
  using  $C-def\ that$ 
  unfolding  $Chains-def\ order-of-orders-def\ strict-extends-def\ extends-def$ 
  by  $blast$ 

```

```

have  $preorder-UnC: preorder-on\ A\ (\bigcup C)$ 
proof $(intro\ preorder-onI)$ 
  show  $refl-on\ A\ (\bigcup C)$ 
    using  $preorder-onD(1)[OF\ preorder-r]\ C-nonempty$ 
    unfolding  $refl-on-def$  by  $auto$ 

```

```

from  $total-sub-C$  show  $trans\ (\bigcup C)$ 
  using  $chain-subset-trans-Union[unfolded\ chain-subset-def]$ 
  by  $(metis\ preorder-onD(2)[OF\ preorder-r])$ 
qed

```

We show that $\bigcup C$ strictly extends the base relation.

```

have  $strict-extends-UnC: strict-extends\ (\bigcup C)\ base-r$ 
proof $(intro\ strict-extendsI)$ 
  note  $extends-r-unfolded = extends-r[unfolded\ extends-def\ strict-extends-def]$ 

```

```

show  $base-r \subseteq (\bigcup C)$ 
  using  $C-nonempty\ extends-r-unfolded$ 
  by  $blast$ 

```

```

then show  $asym-factor\ base-r \subseteq asym-factor\ (\bigcup C)$ 
  using  $extends-r-unfolded$ 

```

unfolding *asym-factor-def* **by** *auto*

show *sym-factor* $(\bigcup C) \subseteq (\text{sym-factor } \text{base-r})^=$

proof(*safe*)

fix *x y* **assume** $(x, y) \in \text{sym-factor } (\bigcup C)$ $(x, y) \notin \text{sym-factor } \text{base-r}$

then have $(x, y) \in \bigcup C$ $(y, x) \in \bigcup C$

unfolding *sym-factor-def* **by** *blast+*

with *extends-r* **obtain** *c* **where** $c \in C$ $(x, y) \in c$ $(y, x) \in c$

strict-extends *c* *base-r*

using *total-sub-C* **by** *blast*

then have $(x, y) \in \text{sym-factor } c$

unfolding *sym-factor-def* **by** *blast*

with $\langle \text{strict-extends } c \text{ base-r} \rangle$ $\langle (x, y) \notin \text{sym-factor } \text{base-r} \rangle$

show $x = y$

unfolding *strict-extends-def* **by** *blast*

qed

qed

from *preorder-UnC* *strict-extends-UnC* **show** $(\bigcup C) \in \text{Field } \text{order-of-orders}$

unfolding *Field-order-of-orders* **by** *simp*

Lastly, we prove by contradiction that $\bigcup C$ is an upper bound for the chain.

show $\forall a \in C. (a, \bigcup C) \in \text{order-of-orders}$

proof(*rule ccontr*)

presume $\exists a \in C. (a, \bigcup C) \notin \text{order-of-orders}$

then obtain *m* **where** $m \in C$ $(m, \bigcup C) \notin \text{order-of-orders}$

by *blast*

hence *strict-extends-m: strict-extends m base-r preorder-on A m*

using *extends-r preorder-r* **by** *blast+*

with *m* **have** $\neg \text{strict-extends } (\bigcup C) m$

using *preorder-UnC* **unfolding** *order-of-orders-def* **by** *blast*

from *m* **have** $m \subseteq \bigcup C$

by *blast*

moreover

have *sym-factor* $(\bigcup C) \subseteq (\text{sym-factor } m)^=$

proof(*safe*)

fix *a b*

assume $(a, b) \in \text{sym-factor } (\bigcup C)$ $(a, b) \notin \text{sym-factor } m$

then have $(a, b) \in \text{sym-factor } \text{base-r} \vee (a, b) \in \text{Id}$

using *strict-extends-UnC[unfolded strict-extends-def]* **by** *blast*

with $\langle (a, b) \notin \text{sym-factor } m \rangle$ *strict-extends-m(1)* **show** $a = b$

by (*auto elim: strict-extendsE simp: sym-factor-mono[THEN in-mono]*)

qed

ultimately

have $\neg \text{asym-factor } m \subseteq \text{asym-factor } (\bigcup C)$

```

using  $\langle \neg \text{strict-extends } (\bigcup C) m \rangle$  unfolding strict-extends-def extends-def
by blast

then obtain  $x y$  where
 $(x, y) \in m (y, x) \notin m (x, y) \in \text{asym-factor } m (x, y) \notin \text{asym-factor } (\bigcup C)$ 
unfolding asym-factor-def by blast

then obtain  $w$  where  $w \in C (y, x) \in w$ 
unfolding asym-factor-def using  $\langle m \in C \rangle$  by auto

with  $\langle (y, x) \notin m \rangle$  have  $\neg \text{extends } m w$ 
unfolding extends-def by auto
moreover
from  $\langle (x, y) \in m \rangle$  have  $\neg \text{extends } w m$ 
proof(cases  $(x, y) \in w$ )
  case True
    with  $\langle (y, x) \in w \rangle$  have  $(x, y) \notin \text{asym-factor } w$ 
      unfolding asym-factor-def by simp
    with  $\langle (x, y) \in \text{asym-factor } m \rangle$  show  $\neg \text{extends } w m$ 
      unfolding extends-def by auto
  qed (auto simp: extends-def)

ultimately show False
  using  $\langle m \in C \rangle \langle w \in C \rangle$ 
  using C-def[unfolded Chains-def order-of-orders-def strict-extends-def]
  by auto
qed blast
qed
qed

```

Let our maximal element be named *max*:

```

from this obtain  $max$ 
where  $max\text{-field}: max \in \text{Field order-of-orders}$ 
and  $is\text{-max}: \forall a \in \text{Field order-of-orders}. (max, a) \in \text{order-of-orders} \longrightarrow a = max$ 
by auto

```

```

from  $max\text{-field}$  have  $max\text{-extends-base}: \text{preorder-on } A \text{ max strict-extends } max$ 
base-r
using Field-order-of-orders by blast+

```

We still have to show, that *max* is a strict linear order, meaning that it is also a total order:

```

have  $total\text{-on } A \text{ max}$ 
proof
  fix  $x y :: 'a$ 
  assume  $x \neq y x \in A y \in A$ 

  show  $(x, y) \in max \vee (y, x) \in max$ 

```

proof (*rule ccontr*)

Assume that max is not total, and x and y are incomparable. Then we can extend max by setting $x < y$:

presume $(x, y) \notin max$ **and** $(y, x) \notin max$
let $?max' = (insert\ (x, y)\ max)^+$

note $max'-extends-max = can-extend-preorder[OF$
 $\langle preorder-on\ A\ max \rangle \langle y \in A \rangle \langle x \in A \rangle \langle (y, x) \notin max \rangle]$

hence $max'-extends-base: strict-extends\ ?max'\ base-r$
using $\langle strict-extends\ max\ base-r \rangle$ **transp-strict-extends** **by** (*auto elim: transpE*)

The extended relation is greater than max , which is a contradiction.

have $(max, ?max') \in order-of-orders$
using $max'-extends-base\ max'-extends-max\ max-extends-base$
unfolding $order-of-orders-def$ **by** *simp*
thus *False*
using *FieldI2* $\langle (x, y) \notin max \rangle$ **is-max** **by** *fastforce*
qed *simp-all*
qed

with $\langle preorder-on\ A\ max \rangle$ **have** $total-preorder-on\ A\ max$
unfolding $total-preorder-on-def$ **by** *simp*

with $\langle strict-extends\ max\ base-r \rangle$ **show** *?thesis* **by** *blast*
qed

With this extension theorem, we can easily prove Szpilrajn's theorem and its equivalent for partial orders.

corollary *partial-order-extension:*

assumes $partial-order-on\ A\ r$
shows $\exists r-ext. linear-order-on\ A\ r-ext \wedge r \subseteq r-ext$

proof –

from *assms* **strict-extends-preorder-on** **obtain** $r-ext$ **where** $r-ext:$
 $total-preorder-on\ A\ r-ext\ strict-extends\ r-ext\ r$
unfolding $partial-order-on-def$ **by** *blast*

with *assms* **have** $antisym\ r-ext$
unfolding $partial-order-on-def$ **using** $strict-extends-antisym$ **by** *blast*

with *assms* $r-ext$ **have** $linear-order-on\ A\ r-ext \wedge r \subseteq r-ext$
unfolding $total-preorder-on-def\ order-on-defs\ strict-extends-def\ extends-def$
by *blast*
then show *?thesis* **..**

qed

corollary *Szpilrajn:*

assumes *strict-partial-order* r
shows $\exists r\text{-ext. strict-linear-order } r\text{-ext} \wedge r \subseteq r\text{-ext}$
proof –
from *assms* **have** *partial-order* $(r^=)$
by (*auto simp: antisym-if-irrefl-trans strict-partial-order-def*)
from *partial-order-extension*[*OF this*] **obtain** $r\text{-ext}$ **where** *linear-order* $r\text{-ext}$ $(r^=)$
 $\subseteq r\text{-ext}$
by *blast*
with *assms* **have** $r \subseteq r\text{-ext} - Id$ *strict-linear-order* $(r\text{-ext} - Id)$
by (*auto simp: irrefl-def strict-linear-order-on-diff-Id dest: strict-partial-orderD(2)*)
then show *?thesis* **by** *blast*
qed

corollary *acyclic-order-extension*:

assumes *acyclic* r
shows $\exists r\text{-ext. strict-linear-order } r\text{-ext} \wedge r \subseteq r\text{-ext}$
proof –
from *assms* **have** *strict-partial-order* (r^+)
unfolding *strict-partial-order-def* **using** *acyclic-irrefl trans-trancl* **by** *blast*
thus *?thesis*
by (*meson Szpilrajn r-into-trancl' subset-iff*)
qed

3 Consistency

As a weakening of transitivity, Suzumura introduces the notion of consistency which rules out all preference cycles that contain at least one strict preference. Consistency characterises those order relations which can be extended (in terms of *extends*) to a total order relation.

definition *consistent* :: ' a *rel* \Rightarrow *bool*
where *consistent* $r = (\forall (x, y) \in r^+. (y, x) \notin \text{asym-factor } r)$

lemma *consistentI*: $(\bigwedge x y. (x, y) \in r^+ \implies (y, x) \notin \text{asym-factor } r) \implies \text{consistent } r$

unfolding *consistent-def* **by** *blast*

lemma *consistent-if-preorder-on*[*simp*]:

preorder-on A $r \implies \text{consistent } r$

unfolding *preorder-on-def consistent-def asym-factor-def* **by** *auto*

lemma *consistent-asym-factor*[*simp*]: $\text{consistent } r \implies \text{consistent } (\text{asym-factor } r)$

unfolding *consistent-def*

using *asym-factor-tranclE* **by** *fastforce*

lemma *acyclic-asym-factor-if-consistent*[*simp*]: $\text{consistent } r \implies \text{acyclic } (\text{asym-factor } r)$

unfolding *consistent-def acyclic-def*

using *asym-factor-tranclE* **by** (*metis case-prodD trancl.simps*)

lemma *consistent-Restr[simp]*: *consistent* $r \implies$ *consistent* (*Restr* r A)
unfolding *consistent-def asym-factor-def*
using *trancl-mono* **by** *fastforce*

This corresponds to Theorem 2.2 [2].

theorem *trans-if-refl-total-consistent*:
assumes *refl* r *total* r **and** *consistent* r
shows *trans* r

proof

fix x y z **assume** $(x, y) \in r$ $(y, z) \in r$

from $\langle (x, y) \in r \rangle \langle (y, z) \in r \rangle$ **have** $(x, z) \in r^+$
by *simp*

hence $(z, x) \notin$ *asym-factor* r

using \langle *consistent* r \rangle **unfolding** *consistent-def* **by** *blast*

hence $x \neq z \implies (x, z) \in r$

unfolding *asym-factor-def* **using** \langle *total* r \rangle

by (*auto simp: total-on-def*)

then show $(x, z) \in r$

apply (*cases* $x = z$)

using *refl-onD[OF refl r]* **by** *blast+*

qed

lemma *order-extension-if-consistent*:

assumes *consistent* r

obtains *r-ext* **where** *extends* *r-ext* r *total-preorder* *r-ext*

proof –

from *assms* **have** *extends: extends* (r^*) r

unfolding *extends-def consistent-def asym-factor-def*

using *rtranclD* **by** (*fastforce simp: Field-def*)

have *preorder: preorder* (r^*)

unfolding *preorder-on-def* **using** *refl-on-def trans-def* **by** *fastforce*

from *strict-extends-preorder-on[OF preorder]* *extends* **obtain** *r-ext* **where**
total-preorder *r-ext* *extends* *r-ext* r

using *transpE[OF transp-extends]* **unfolding** *strict-extends-def* **by** *blast*

then show *thesis* **using** *that* **by** *blast*

qed

lemma *consistent-if-extends-trans*:

assumes *extends* *r-ext* r *trans* *r-ext*

shows *consistent* r

proof (*rule consistentI, standard*)

fix x y **assume** $*$: $(x, y) \in r^+$ $(y, x) \in$ *asym-factor* r

with *assms* **have** $(x, y) \in$ *r-ext*

using *trancl-subst-extends-if-trans[OF assms]* **by** *blast*

moreover from $*$ *assms* **have** $(x, y) \notin$ *r-ext*

unfolding *extends-def asym-factor-def* **by** *auto*
ultimately show *False* **by** *blast*
qed

With Theorem 2.6 [2], we show that *consistent* characterises the existence of order extensions.

corollary *order-extension-iff-consistent*:
 $(\exists r\text{-ext. extends } r\text{-ext } r \wedge \text{total-preorder } r\text{-ext}) \longleftrightarrow \text{consistent } r$
using *order-extension-if-consistent consistent-if-extends-trans*
by (*metis total-preorder-onD(2)*)

The following theorem corresponds to Theorem 2.7 [2]. Bossert and Suzumura claim that this theorem generalises Szpilrajn's theorem; however, we cannot use the theorem to strictly extend a given order Q . Therefore, it is not strong enough to extend a strict partial order to a strict linear order. It works for total preorders (called orderings by Bossert and Suzumura). Unfortunately, we were not able to generalise the theorem to allow for strict extensions.

theorem *general-order-extension-iff-consistent*:
assumes $\bigwedge x y. \llbracket x \in S; y \in S; x \neq y \rrbracket \implies (x, y) \notin Q^+$
assumes *total-preorder-on S Ord*
shows $(\exists \text{Ext. extends Ext } Q \wedge \text{total-preorder Ext} \wedge \text{Restr Ext } S = \text{Ord})$
 $\longleftrightarrow \text{consistent } Q$ (**is** *?ExExt* \longleftrightarrow -)

proof

assume *?ExExt*
then obtain *Ext* **where**
extends Ext Q
refl Ext trans Ext total Ext
Restr Ext S = Restr Ord S
using *total-preorder-onD* **by** *fast*

show *consistent Q*

proof(*rule consistentI*)

fix $x y$ **assume** $(x, y) \in Q^+$
with $\langle \text{extends Ext } Q \rangle \langle \text{trans Ext} \rangle$ **have** $(x, y) \in \text{Ext}$
unfolding *extends-def* **by** (*metis trancl-id trancl-mono*)
then have $(y, x) \notin \text{asym-factor Ext}$
unfolding *asym-factor-def* **by** *blast*
with $\langle \text{extends Ext } Q \rangle$ **show** $(y, x) \notin \text{asym-factor } Q$
unfolding *extends-def asym-factor-def* **by** *blast*

qed

next

assume *consistent Q*

define Q' **where** $Q' \equiv Q^* \cup \text{Ord} \cup \text{Ord } O \text{ } Q^* \cup Q^* O \text{ } \text{Ord} \cup (Q^* O \text{ } \text{Ord}) O$
 Q^*

have *refl (Q*) trans (Q*) refl-on S Ord trans Ord total-on S Ord*
using *refl-rtrancl trans-rtrancl total-preorder-onD[OF <total-preorder-on S Ord>]*
by - *assumption*

```

have preorder-Q': preorder Q'
proof
  show refl Q'
    unfolding Q'-def refl-on-def by auto

  from ⟨trans (Q*)⟩ ⟨refl-on S Ord⟩ ⟨trans Ord⟩ show trans Q'
    unfolding Q'-def[simplified]
    apply(safe intro!: transI)
    unfolding relcomp.simps
    by (metis assms(1) refl-on-domain rtranclD transD)+
qed

have consistent Q'
  using consistent-if-preorder-on preorder-Q' by blast

have extends Q' Q
proof(rule extendsI)
  have Q ⊆ Restr (Q*) (Field Q)
    by (auto intro: FieldI1 FieldI2)
  then show Q ⊆ Q'
    unfolding Q'-def by blast

  from ⟨consistent Q⟩ have consistentD: (x, y) ∈ Q+ ⇒ (y, x) ∈ Q ⇒ (x, y)
  ∈ Q for x y
    unfolding consistent-def asym-factor-def using rtranclD by fastforce
  have refl-on-domainE: [ (x, y) ∈ Ord; x ∈ S ⇒ y ∈ S ⇒ P ] ⇒ P for x
  y P
    using refl-on-domain[OF ⟨refl-on S Ord⟩] by blast

  show asym-factor Q ⊆ asym-factor Q'
    unfolding Q'-def asym-factor-def Field-def
    apply(safe)
    using assms(1) consistentD refl-on-domainE
    by (metis r-into-rtrancl rtranclD rtrancl-trancl-trancl)+
qed

with strict-extends-preorder-on[OF ⟨preorder Q'⟩]
obtain Ext where Ext: extends Ext Q' extends Ext Q total-preorder Ext
  unfolding strict-extends-def
  by (metis transpE transp-extends)

have not-in-Q': x ∈ S ⇒ y ∈ S ⇒ (x, y) ∉ Ord ⇒ (x, y) ∉ Q' for x y
  using assms(1) unfolding Q'-def
  apply(safe)
  by (metis ⟨refl-on S Ord⟩ refl-on-def refl-on-domain rtranclD)+

have Restr Ext S = Ord
proof

```

```

from  $\langle \text{extends Ext } Q' \rangle$  have  $\text{Ord} \subseteq \text{Ext}$ 
  unfolding  $Q'\text{-def extends-def}$  by auto
with  $\langle \text{refl-on } S \text{ Ord} \rangle$  show  $\text{Ord} \subseteq \text{Restr Ext } S$ 
  using refl-on-domain by fast
next
have  $(x, y) \in \text{Ord}$  if  $x \in S$  and  $y \in S$  and  $(x, y) \in \text{Ext}$  for  $x \ y$ 
proof(rule ccontr)
  assume  $(x, y) \notin \text{Ord}$ 
  with that not-in- $Q'$  have  $(x, y) \notin Q'$ 
    by blast
  with  $\langle \text{refl-on } S \text{ Ord} \rangle$   $\langle \text{total-on } S \text{ Ord} \rangle$   $\langle x \in S \rangle$   $\langle y \in S \rangle$   $\langle (x, y) \notin \text{Ord} \rangle$ 
  have  $(y, x) \in \text{Ord}$ 
    unfolding refl-on-def total-on-def by fast
  hence  $(y, x) \in Q'$ 
    unfolding  $Q'\text{-def}$  by blast
  with  $\langle (x, y) \notin Q' \rangle$   $\langle (y, x) \in Q' \rangle$   $\langle \text{extends Ext } Q' \rangle$ 
  have  $(x, y) \notin \text{Ext}$ 
    unfolding extends-def asym-factor-def by auto
  with  $\langle (x, y) \in \text{Ext} \rangle$  show False by blast
qed
then show  $\text{Restr Ext } S \subseteq \text{Ord}$ 
  by blast
qed

with Ext show ?ExExt by blast
qed

```

4 Strong consistency

We define a stronger version of *consistent* which requires that the relation does not contain hidden preference cycles, i.e. if there is a preference cycle then all the elements in the cycle should already be related (in both directions). In contrast to consistency which characterises relations that can be extended, strong consistency characterises relations that can be extended strictly (cf. *strict-extends*).

definition *strongly-consistent* $r \equiv \text{sym-factor } (r^+) \subseteq \text{sym-factor } (r^-)$

lemma *consistent-if-strongly-consistent*: $\text{strongly-consistent } r \implies \text{consistent } r$
unfolding *strongly-consistent-def consistent-def*
by (*auto simp: sym-factor-def asym-factor-def*)

lemma *strongly-consistentI*: $\text{sym-factor } (r^+) \subseteq \text{sym-factor } (r^-) \implies \text{strongly-consistent } r$
unfolding *strongly-consistent-def* **by** *blast*

lemma *strongly-consistent-if-trans-strict-extension*:
assumes *strict-extends r-ext r*
assumes *trans r-ext*

shows *strongly-consistent* r
proof(*unfold strongly-consistent-def, standard*)
fix x **assume** $x \in \text{sym-factor } (r^+)$
then show $x \in \text{sym-factor } (r^-)$
using *assms trancl-subst-extends-if-trans[OF extends-if-strict-extends]*
by (*metis sym-factor-mono strict-extendsE subsetD sym-factor-refl*)
qed

lemma *strict-order-extension-if-consistent*:
assumes *strongly-consistent* r
obtains $r\text{-ext}$ **where** *strict-extends* $r\text{-ext}$ r *total-preorder* $r\text{-ext}$
proof –
from *assms* **have** *strict-extends* (r^+) r
unfolding *strongly-consistent-def strict-extends-def extends-def asym-factor-def sym-factor-def*
by (*auto simp: Field-def dest: tranclD*)
moreover have *strict-extends* (r^*) (r^+)
unfolding *strict-extends-def extends-def*
by (*auto simp: asym-factor-rtrancl sym-factor-def dest: rtranclD*)
ultimately have *extends: strict-extends* (r^*) r
using *transpE[OF transp-strict-extends]* **by blast**

have *preorder* (r^*)
unfolding *preorder-on-def* **using** *refl-on-def trans-def* **by fastforce**
from *strict-extends-preorder-on[OF this]* *extends* **obtain** $r\text{-ext}$ **where**
total-preorder $r\text{-ext}$ *strict-extends* $r\text{-ext}$ r
using *transpE[OF transp-strict-extends]* **by blast**
then show *thesis* **using** *that* **by blast**
qed

experiment begin

We can instantiate the above theorem to get Szpilrajn’s theorem.

lemma
assumes *strict-partial-order* r
shows $\exists r\text{-ext. } \text{strict-linear-order } r\text{-ext} \wedge r \subseteq r\text{-ext}$
proof –
from *assms[unfolded strict-partial-order-def]* **have** *strongly-consistent* r *antisym*
 r
unfolding *strongly-consistent-def* **by** (*simp-all add: antisym-if-irrefl-trans*)
from *strict-order-extension-if-consistent[OF this(1)]* **obtain** $r\text{-ext}$
where *strict-extends* $r\text{-ext}$ r *total-preorder* $r\text{-ext}$
by blast
with *assms[unfolded strict-partial-order-def]*
have *trans* $(r\text{-ext} - \text{Id})$ *irrefl* $(r\text{-ext} - \text{Id})$ *total* $(r\text{-ext} - \text{Id})$ $r \subseteq (r\text{-ext} - \text{Id})$
using *strict-extends-antisym[OF - ‹antisym r›]*
by (*auto simp: irrefl-def elim: strict-extendsE intro: trans-diff-Id dest: total-preorder-onD*)

```
then show ?thesis
  unfolding strict-linear-order-on-def by blast
qed

end
```

References

- [1] Wikipedia: Szpilrajn extension theorem. https://en.wikipedia.org/wiki/Szpilrajn_extension_theorem. Accessed: 2019-07-27.
- [2] W. Bossert and K. Suzumura. *Consistency, Choice, and Rationality*. Harvard University Press, 2010.
- [3] E. Szpilrajn. Sur l'extension de l'ordre partiel. *Fundamenta Mathematicae*, 16:386–389, 1930.